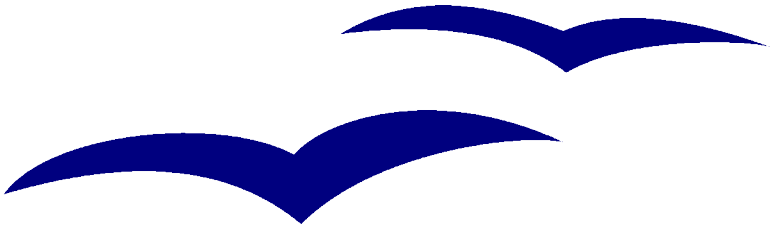
Aprendiendo OOo Basic



Autor :	*Mauricio Baeza Servín*
Correo :	public (NO-SPAM) elmau PUNTO net
Fecha :	15 de Septiembre del 2007
Licencia :	GNU Free Documentation License, v1.3 o posterior
Ultima modificación :	27 de febrero de 2019
Para ayudar :	Apoya este proyecto

Revisiones (gracias):

03-Julio-207 - Gonzalo Fernandez de Cordoba Martos

Índice de contenido

1 Introducción

"Quien recibe una idea de mí, recibe instrucción sin disminuir la mía; igual que quien enciende su vela con la mía, recibe luz sin que yo quede a oscuras"
Thomas Jefferson

¿Así que quieres aprender a programar en OOo Basic? Bien, pues yo también así que ya tenemos un interés común y eso es un buen comienzo. No se si llegaste al lugar indicado, ya me lo contaras. Sabe de antemano que soy un programador autodidacta que reconoce que tiene vicios de los cuales tal vez no me doy cuenta, espero y deseo, que algún buen samaritano que se encuentre con estas notas me ayude a identificarlos. Estos apuntes no tienen otro propósito que compartir mi experiencia al ir experimentando con este maravilloso programa llamado OpenOffice.org y por supuesto, con su lenguaje de macros OOo Basic, una experiencia sumamente placentera. Me decidí a aventurarme a escribir, por la falta de documentación en español para programar con OOo Basic, a base de paciencia y disciplina he ido profundizado en el dominio del lenguaje, creo que puedo ayudarte a que tu curva de aprendizaje no sea tan pronunciada como la mía, si se cumple este segundo y sencillo propósito, el tiempo y esfuerzo invertido, bien habrán valido la pena, así que, como dicen en mi pueblo -sobre advertencia no hay engaño-, adelante, empecemos...

La versión de OpenOffice.org que utilizo para escribir estas notas y mostrarte los ejemplos de programación, normalmente es la ultima estable (3.2.1 en este momento) descargada directamente desde http://es.openoffice.org, esto es por que algunas distribuciones GNU/Linux, compilan sus propias versiones, casi todas “deberían” de trabajar de forma satisfactoria, pero te recomiendo bajar e instalar la oficial para que nos entendamos mejor. Daré por sentado que esta versión es también con la que trabajas, no te garantizo que los ejemplos mostrados aquí funcionen con otras versiones, de hecho no te garantizo que funcionen con ninguna, ojo, no me malinterpretes, es tan alta la cantidad de equipos, sistemas y configuraciones diferentes, que es cuestión software es difícil garantizar algo y si lo dudas, leete alguna de las muchas licencias de software que hay en el mercado, incluyendo por supuesto, las de software libre. Lo que si te puedo asegurar, es que cada uno de los ejemplos que te muestro los he probado más de una vez antes de darlos por buenos, así que en general espero no tengas problemas con ellos.

También es importante que sepas, que trabajo sobre GNU/Linux en varias distribuciones (por ahora) ArchLinux (http://archlinux-es.org), aunque continuamente y más seguido de lo que te imaginas cambio de distribución, pues aun no puedo abandonar esa manía de probar y experimentar (y a veces suicidarte), tanto las nuevas distribuciones como las nuevas versiones de las ya existentes, por lo que las interfaces podrían variar un poco con respecto al sistema operativo (S.O.) que usas, por ejemplo Mac/OS o Windows u otro, aunque espero que esto no sea un problema pues en si lo único que cambia es la decoración de las ventanas, lo demás, “debería” ser igual en la mayoría de los casos, no obstante no esta de más que al enviar una consulta especifiques la versión de OOo que usas y el S.O. donde trabajas, aunque claro, como no habría de recomendartelo, que esperas para probar alguna de las decenas de distribuciones GNU/Linux existentes, seguro que alguna se adapta a tu gusto y forma de trabajar, por lo menos, no dejes de intentarlo.

Este documento esta en constante crecimiento, puedes verificar la ultima versión en: www.universolibre.org, como eres libre de distribuirlo, si te parece que lo merece, te invito a que hables de el y lo difundas, y si no te gusta, pues no seas chismoso.

Para comentarios, notificación de errores, sugerencias, colaboraciones y dudas puedes usar las siguientes alternativas:

- Presionar la tecla **F1** ayuda mucho, antes de escribir para dudas, asegúrate que:
  - No esta resuelta en estas notas
  - No esta resuelta en la ayuda
  - No se ha comentado en las listas de correo o foros
  - ¿Ya realizaste una consulta en tu buscador favorito?
  - Y lo más importante, que hayas tratado de resolverlo por ti mismo

- De preferencia, plantea tus dudas en las listas o foros y no a mi correo, estoy en la mejor disposición de ayudarte, pero recuerda que tu duda, tal vez otros la tengan, y si se publica en alguna lista o foro, la posible respuesta también le podrá servir a mas de uno, si aun así, deseas escribirme, procura ser lo mas claro posible en tu exposición y aunque procuro contestar siempre, no te garantizo una respuesta inmediata pues a parte de escribir estas notas, tengo otras varias actividades que tal vez no sean de tu interés, pero que consumen un poco de mi tiempo, agradezco tu comprensión al respecto.

- Inscribete a cualquiera de las siguientes listas de correo, de preferencia a las dos, pues en las dos participo constantemente.

Para enviar mensajes	*Para subscribirse*
users@es.openoffice.org	users-subscribe@es.openoffice.org
oooes@correolibre.net	http://www.correolibre.net/mailman/listinfo/oooes_correolibre.net
Procuro participar en el foro oficial de macros: Foro de Macros en Español	

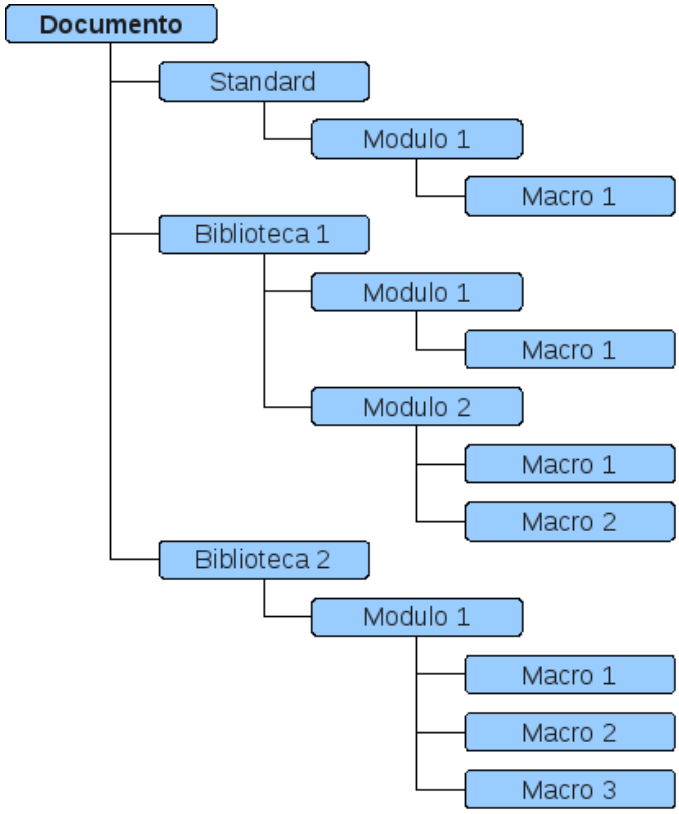
- Estas notas son posibles gracias a la generosidad de la gran comunidad OpenOffice.org en todo el mundo y por supuesto a la comunidad mundial del software libre, es decir, a las ***personas*** que “todos los días” desarrollan, usan y difunden el software libre.

Dedico este trabajo a las personas que me aman
sin su amor, no seria lo que soy...

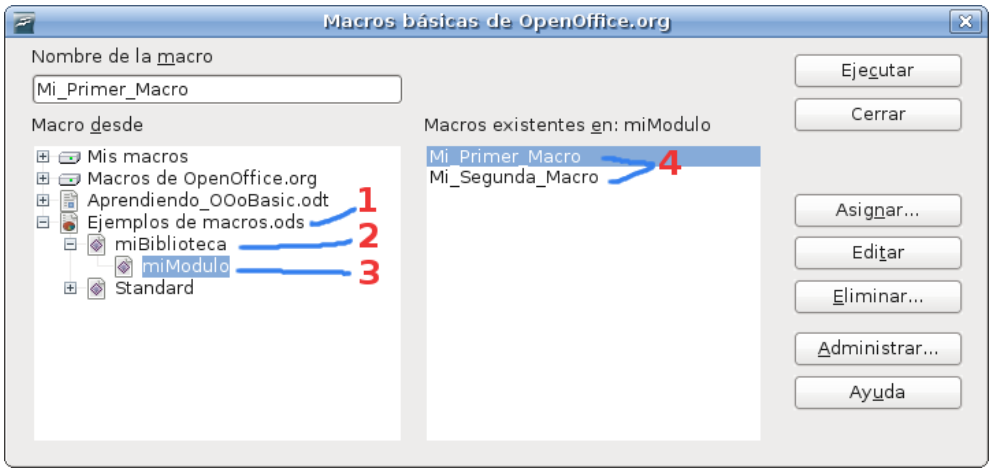
2 Conceptos iniciales

Como soy una persona floja, tal vez pienses que no me gusta trabajar y estarás en lo correcto, por ello, cuando descubrí lo que se podía hacer con los *lenguajes de programación*, comprendí que esto es lo que necesitaba (y lo que me gustaba), los lenguajes de programación nos permiten hacer que la computadora haga por nosotros, esas tareas complejas, repetitivas y tediosas (también las divertidas), en estos apuntes, trataremos de aprender como hacer en OpenOffice.org, esas tareas complejas, repetitivas y tediosas con uno de sus varios lenguajes con los que se puede programar en él, me refiero claro esta, al OOo Basic. Las instrucciones que usamos para decirle al programa qué hacer, como, cuando, el orden y la secuencia en que queremos que lo haga, las llamaremos *MACROS*, estas macros podemos hacerlas de forma "semiautomática" con la *Grabadora de macros* (por ahora sólo presente en Writer y Calc) o escribirlas nosotros desde cero (podemos crearlas desde todas las aplicaciones de OpenOffice.org), a mi criterio, a esta grabadora todavía le falta madurar un poco, por lo cual, crearemos nuestras macros desde cero, veras que no es tan complicado y si, muy divertido.

Lo primero que tienes que aprender (y recordar) es que las macros se guardan en **módulos**, estos a su vez se guardan y organizan en **bibliotecas**, las cuales, están contenidas dentro de documentos, con lo cual tendremos el siguiente diagrama, por cierto, hecho en Draw.



Todos los documentos, tienen una biblioteca especial predeterminada llamada **Standard**, a esta biblioteca le puedes agregar y eliminar módulos, pero no puedes eliminarla, mas adelante veremos y aprenderemos que los módulos también, además de macros, pueden contener "*funciones*", y los documentos también pueden contener "*diálogos*". En la siguiente imagen puedes ver un documento (1) con una biblioteca (2), un modulo (3) y dos macros (4).

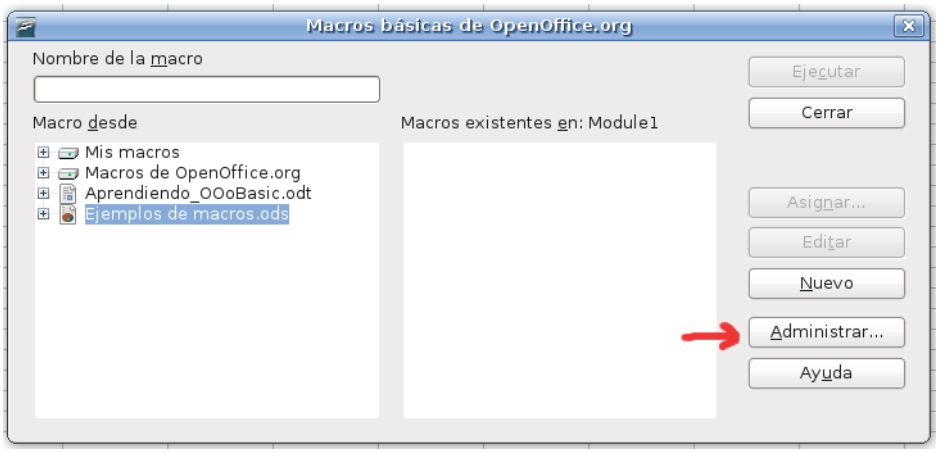


Empecemos con lo divertido, crearemos el archivo que te muestro en la imagen anterior, tal como esta y el cual nos servirá para guardar las macros que vayamos creando:

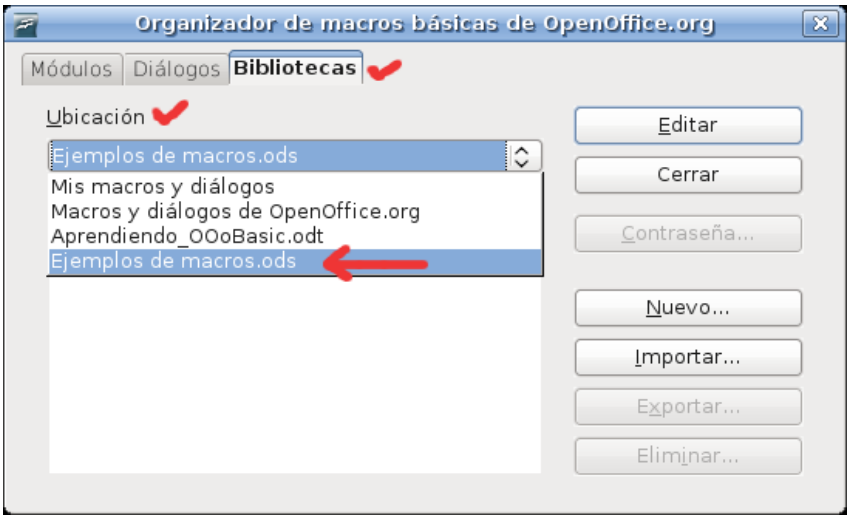
Abre Calc (puedes usar cualquier otra aplicación de OpenOffice.org) y guarda el documento nuevo en el lugar y con el nombre que consideres, para nuestros fines usaremos el nombre "Ejemplos de macros.ods".

Ve al menú *Herramientas | Macros | Organizar macros | OpenOffice.org Basic...*

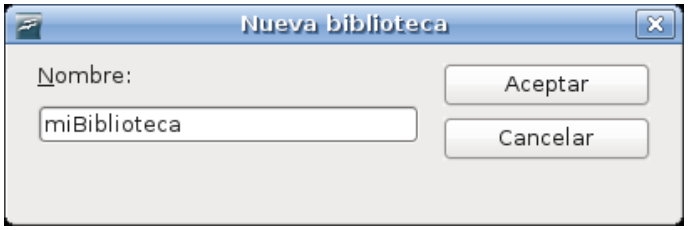
Te mostrara el siguiente cuadro de dialogo, presiona el botón de comando ***Administrar***



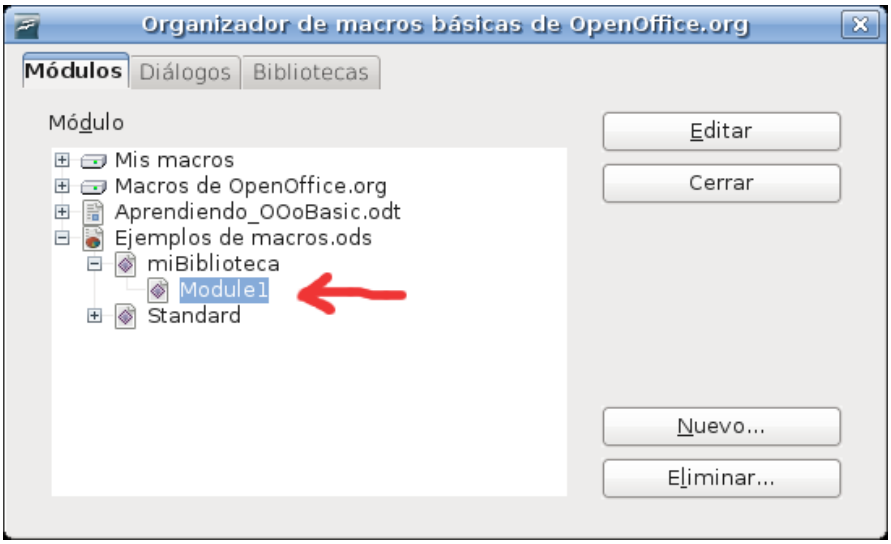
Al mostrarte el siguiente cuadro de dialogo, da un clic en la ficha ***Bibliotecas,*** asegurate de que dentro del cuadro de lista desplegable ***Ubicación***, este seleccionado el documento al que le anexaremos la nueva biblioteca, en nuestro caso, el documento de Calc, *Ejemplos de macros.ods*



Inmediatamente después, presiona el botón de comando ***Nuevo***, tras lo cual te mostrara un cuadro de dialogo, solicitando el nombre de la nueva biblioteca, la cual llamaremos *miBiblioteca*.

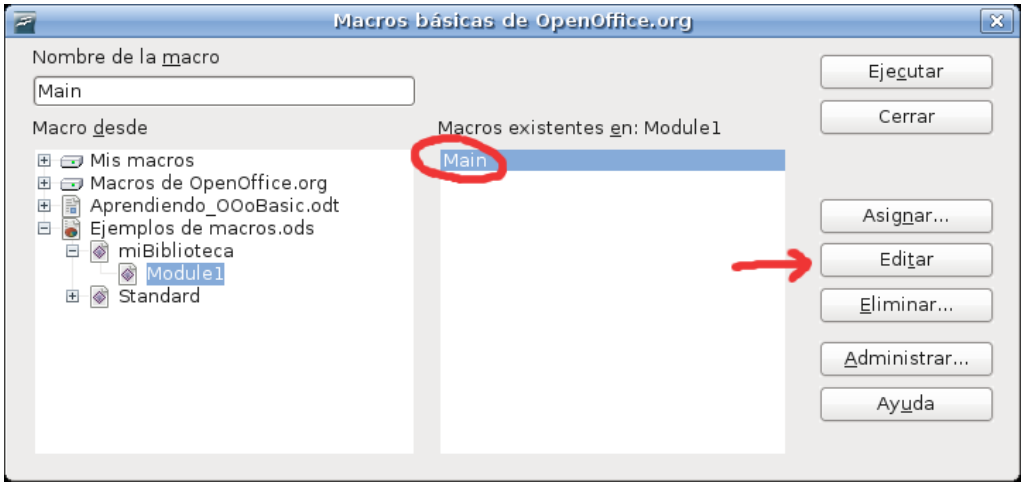


Después de dar clic en ***Aceptar***, notaras que la nueva biblioteca se agrega a la lista, ahora, regresa a la ficha ***Módulos*** y observMia que automáticamente a la nueva biblioteca se le anexó un modulo nuevo llamado *Module1*, si no lo ves, tal ves tengas qué dar clic en el botón expandir representado por un signo de suma (+) a la izquierda del nombre del documento. Da un clic en el botón de comando ***Cerrar*** para regresar al anterior cuadro de dialogo.

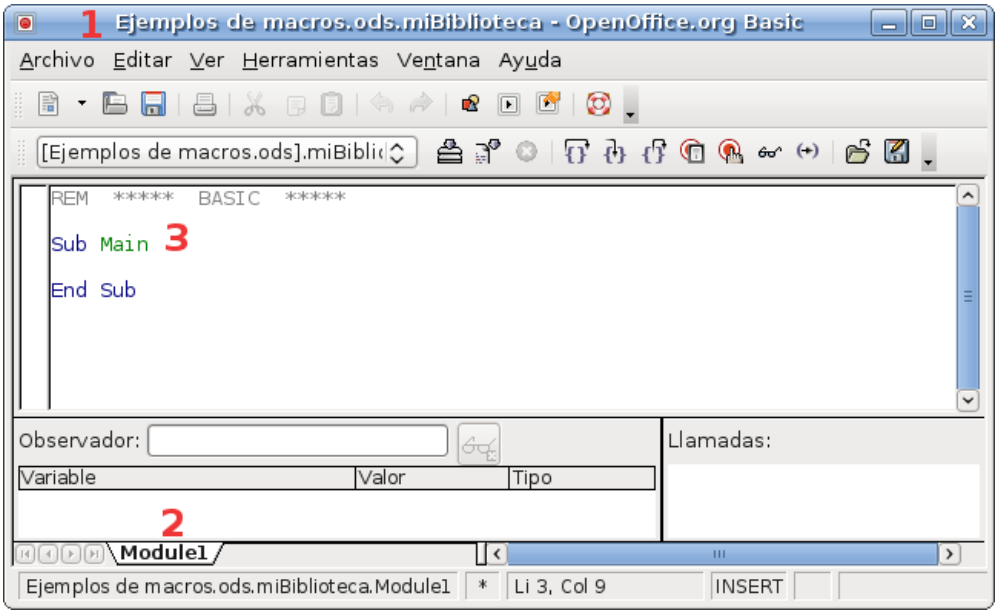


Ahora, despliega los elementos de nuestro documento, esto lo logras con el icono del símbolo mas (+) que esta a la izquierda del nombre del documento, lo que veas tiene que ser igual o muy parecido a la siguiente ilustración, observa que también en la lista de macros, aparece ya una con el nombre de Main (elipse roja), del mismo modo que al agregar una nueva biblioteca esta

aparece con un nuevo modulo llamado de forma predeterminada ***Module1***, cuando agregamos un nuevo modulo, este crea de forma predeterminada, una macro nueva y vacía con el nombre de ***Main***, un nombre que te resultara familiar si alguna vez programaste en C. Estos nombres vienen predeterminados, pero se pueden cambiar cuando quieras, como lo haremos a continuación, para lo cual, da un clic en el botón de comando ***Editar***.



Al dar clic en ***Editar***, te abrirá una nueva ventana, la cual será el centro de trabajo de "todo" lo que hagamos en OOo Basic, esta nueva ventana de hecho es una nueva aplicación, una aplicación muy especial, pues ella nos permitirá escribir todas las macros que nos permita nuestro tiempo e imaginación, esta aplicación, recibe el nombre de Entorno de Desarrollo Integrado (IDE por sus siglas en ingles) y es muy "***importante que te familiarices con él***" lo mas posible, lo cual, afortunadamente se da como consecuencia intrínseca de programar en OOo Basic. Observa como nos muestra el nombre del documento en donde nos encontramos, así como el nombre de la biblioteca en la que estamos trabajando (1), nos muestra, el nombre del modulo activo actualmente (2), en caso de que hubiese más, nos los mostraría en fichas contiguas a este, por ultimo, observa la macro mencionada en el inciso anterior, Main (3)

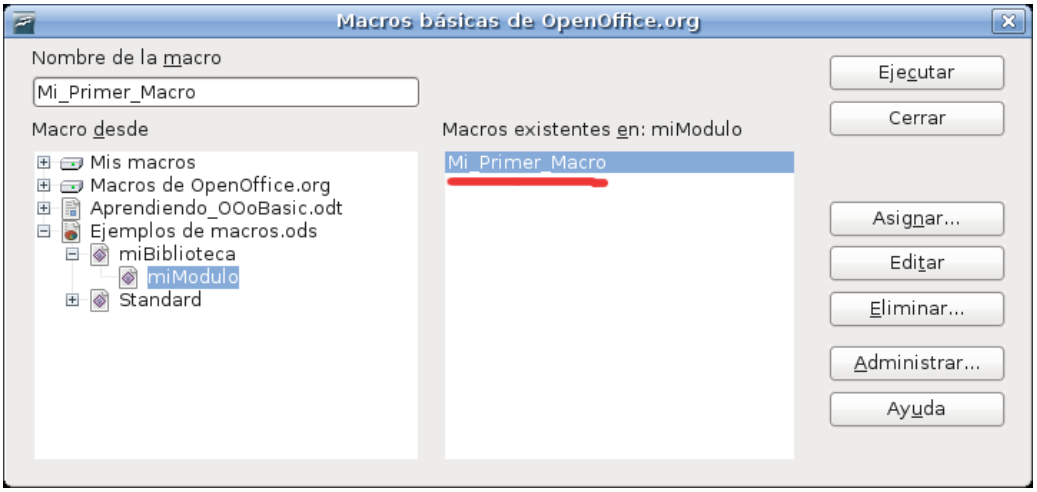


Como siguiente paso, cambiemos el nombre del modulo, para ello, da un clic con el botón secundario de tu ratón (normalmente el botón secundario es el derecho) sobre la ficha que muestra el nombre actual (2), te mostrara un menú contextual entre cuyas opciones veras ***Cambiar nombre***, da un clic en ella y el cursor de escritura (el que parpadea) se posicionara al

final del nombre actual del modulo, ahora, puedes cambiarlo por el que quieras, para nuestro ejercicio le llamaremos **miModulo**, al terminar presiona la tecla {**Enter**} para aceptar el cambio o simplemente activa el área de escritura de código (3) con un clic, esta sección en realidad es un editor de textos, pero un editor con muchas características especiales que iremos conociendo poco a poco, para terminar nuestro primer ejercicio, da un doble clic a la palabra **Main**, con lo cual lograremos seleccionar la palabra completa, puedes reemplazar este nombre por el que quieras (bueno, casi por el que quieras, más adelante, veremos las restricciones al respecto), para continuar con nuestro ejercicio escribiremos ***Mi_Primer_Macro***.

Por ultimo, ve al menú *Archivo | Guardar,* o da un clic en el conocido icono de *Guardar*, o presiona la combinación de teclas CRTL+G, como en la mayor parte de los programas de computo, hay mas de una manera de hacer la misma acción, usa, la que mejor te convenga o guste y guarda tu documento, ya sea con el nombre propuesto o con el que consideres correcto.

En esta misma ventana, ve al menú *Herramientas | Macros | Organizar macros | OpenOffice.org Basic...* y comprueba que lo que tienes es bastante similar a la imagen siguiente, que, salvo un pequeño cambio (que ya notaste) quedo igual al que propusimos al inicio de estas notas.



Los restantes botones de comando presentes en este cuadro de diálogo, aprenderemos a usarlos más adelante.

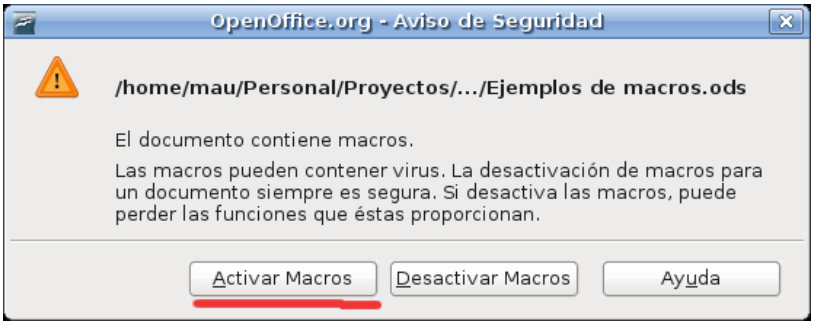
¿Te gusto el curso?, espero que si, eso es todo, fin y hasta pronto. No, no es cierto, con esto cubrimos el 0.001% de lo que pretendo, conste que sólo lo pretendo. Para finalizar este capitulo y esta noche (ya me canse), siento mucho decirte y confesarte, que soy un profesor que deja mucha tarea, así que tu tarea es:

- Practicar mucho
- Agregar y eliminar bibliotecas a diferentes documentos de OpenOffice.org.
- Practicar mucho
- Agregar y eliminar módulos a diferentes bibliotecas.
- Practicar mucho
- Observar como esta estructurada nuestra primer macro y tratar de hacer mas, como consejo, una macro no puede ir dentro de otra.
- Practicar mucho
- Los cuadro de dialogo vistos hasta ahora, tienen algunos controles mas, trata de investigar para que sirven o cual es su función.
- Practicar mucho
- Envíame tus comentarios, impresiones, quejas y dudas, como un servicio agregado, también puedes enviarme tus problemas existenciales.
- !Ah¡ y practica mucho...

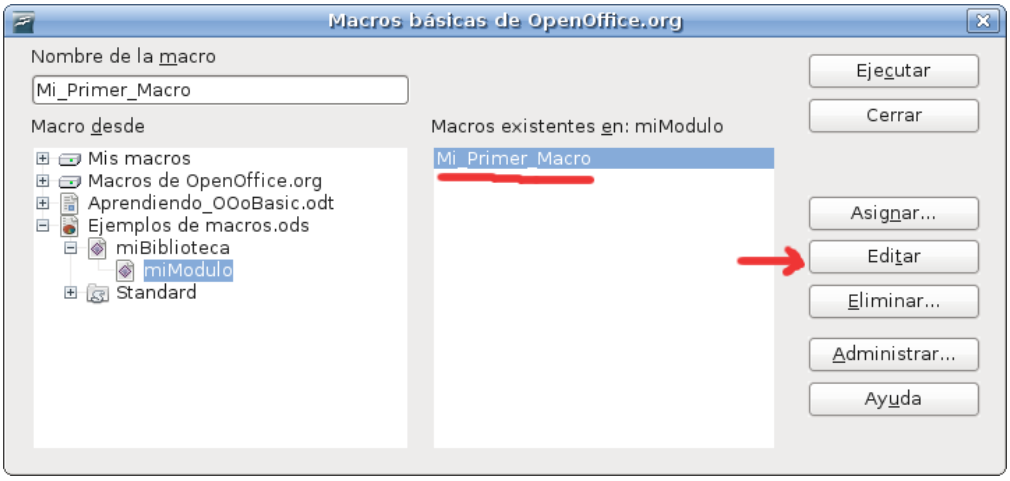
3 Mi primer macro

Ahora que ya sabes donde crearemos nuestras macros, empecemos con ellas...

Abre tu archivo de trabajo, si has seguido estas notas, nuestro archivo, es un documento de Calc nombrado ***Ejemplos* de *macros.ods.*** De forma predeterminada, cuando abres un archivo de OpenOffice.org que contiene macros, te mostrara el siguiente aviso de seguridad, que creo, es bastante claro.



Por supuesto, para poder trabajar correctamente tienes que habilitar las macros, más adelante aprenderemos a evitar este mensaje para los archivos que sabemos son seguros. Puede ser que no te muestre este mensaje y te abra el archivo directamente con las macros deshabilitadas, si este es tu caso, tienes que cambiar el nivel de seguridad como te muestro en 11.1 Seguridad en macros. Ve al menú *Herramientas | Macros | Organizar macros | OpenOffice.org Basic...*y navega hasta tu documento, tu biblioteca y tu modulo, inmediatamente después de seleccionar la macro que lleva por nombre *Mi_Primer_Macro*, presiona el botón de comando ***Editar***. Esta serie de acciones, es la forma que usaremos para acceder a la edición de cualquier macro vista en estas notas, por lo cual, de aquí en adelante, daré por hecho, que ya lo dominas.

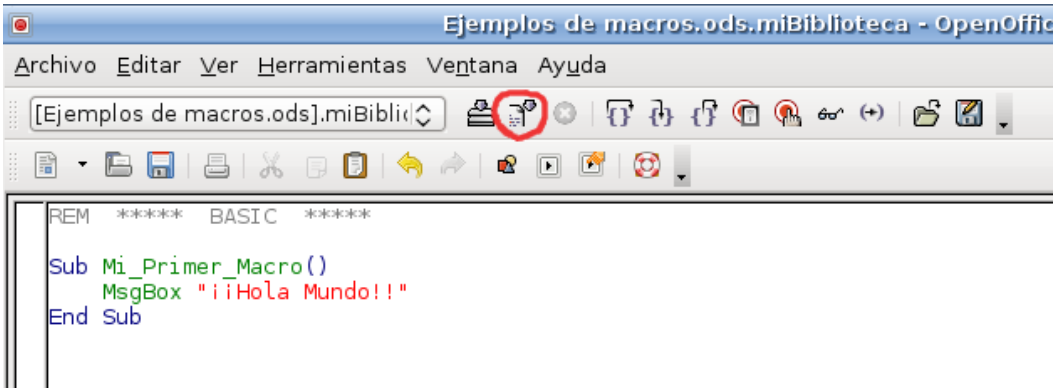


Con la acción anterior, te abrirá el Entorno de desarrollo (de aquí en adelante ***IDE***) y este es el punto interesante y el más divertido, escribir código, es decir, decirle al programa que haga lo que queramos, cuando queramos y como queramos, y para no salirnos de los "santos cánones" de la programación, empezaremos con un clásico...

Y para lograrlo, agrega la siguiente linea a la macro, asegurate de escribirla o copiarla tal cual:

```
MsgBox "¡¡Hola Mundo!!"
```

De modo que quede de la siguiente manera:



Ahora, para "ejecutar" la macro, es decir, que realice su trabajo, presiona el botón de ejecución (circulo rojo en la imagen anterior) o presiona la tecla {**F5**} y listo, tendrás que ver el cuadro de mensaje que te mostré lineas arriba. ¡Felicidades!, ya eres todo un programador, pero espera, no eres todavía un programador completo, pues ya sabes el "como", sólo te falta el saber el "por qué", para lograr lo que decía mi entrañable profesor de topografía:

El hombre teórico sabe "por qué",
el hombre practico sabe el "cómo",
lo ideal es saber "cómo" y "por qué"...

Para entender el "¿por qué?", analicemos la linea que acabamos de agregarle a nuestra macro:

```
MsgBox "¡¡Hola Mundo!!"
```

La primer palabra, es una "***instrucción***" de OOo Basic, es decir, una palabra especial, que el lenguaje, en conjunto con el ***IDE***, reconoce de modo diferente y "realiza", "hace", "ejecuta", una tarea especifica, en el caso de esta "***palabra clave o reservada***" (llamada así, por que “sólo” OOo Basic puede hacer uso de ella, recuerda esto bien, **sólo OOo Basic la puede usar**) `MsgBox`, muestra un "***mensaje***" en un cuadro de dialogo, ¿qué mensaje?, como habrás notado (por que de aquí en adelante tendrás que ser muy observador), el mensaje que muestra es el que "nosotros" le indicamos delante de ella, y como es una "cadena" de texto, lo tenemos que hacer entre comillas y separada de la instrucción por un espacio, ¿sucederá lo mismo con números?, ¿y con fechas o algún otro valor?...

```
MsgBox 1234567890
MsgBox 15/01/1974
MsgBox 1dfgdfh245
```

Realiza todos los experimentos que quieras y me cuentas qué sucede. Observa la primer linea de la macro que acabas de hacer, y que, si has seguido al pie de la letra, no tendrías que haber modificado:

```
REM  *****  BASIC  *****
```

La primer palabra , también es una palabra clave de OOo Basic, y sencillamente lo que "hace", es decirle al programa (nuestra macro), que todo lo haya después de esta palabra, es un comentario, es decir, el lenguaje se lo saltara olímpicamente sin hacerle caso, comentar las lineas de código es una practica sumamente útil, sobre todo, cuando las macros van creciendo de tamaño y complejidad, así que, asume esta recomendación, "casi", como una regla, -siempre comenta tus lineas de código-, por ahí leí en algún foro que un código bien escrito y profesional no necesitaba comentarios, pero como nosotros somos “novatos” haremos caso omiso de ese comentario, pues como dijo alguna vez Ling Yu Tan -me gusta la gente amateur-. Así que ve al código de nuestra macro y comenta la linea que le agregaste, de modo que te quede, así:

```
REM Muestra un mensaje en un cuadro de dialogo
MsgBox "¡¡Hola Mundo!!"
```

Los comentarios, también puedes establecerlos con una comilla simple (')

```
'Muestra un mensaje en un cuadro de dialogo
MsgBox "¡¡Hola Mundo!!"
```

Además, podemos agregarlos, al final de la linea de código:

```
MsgBox "¡¡Hola Mundo!!" REM Muestra un mensaje...dialogo
MsgBox "¡¡Hola Mundo!!" 'Muestra un mensaje... de dialogo
```

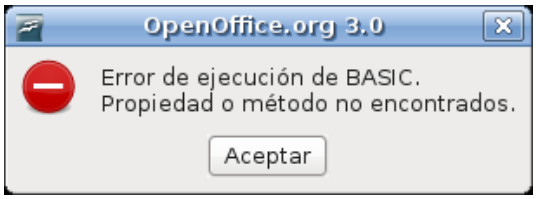
Reitero, procura comentar la mayor parte de tu código, tu memoria te lo agradecerá, así como todos los programadores con los que compartas tu código, si decides que sea software libre, como te recomiendo que sea.

Ahora veamos la linea donde inicia nuestra macro:

```
Sub Mi_Primer_Macro
```

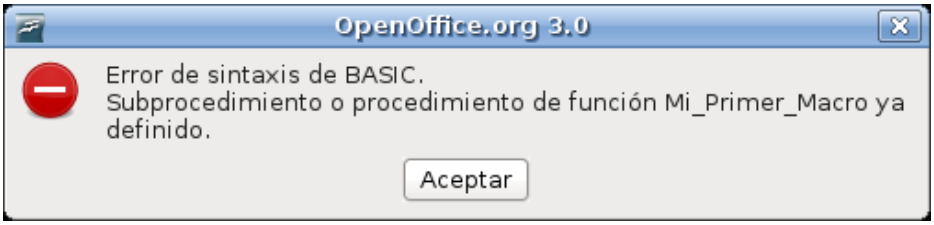
Como se que eres observador, ya habrás deducido que la primer palabra, si, efectivamente, es una palabra clave de OOo Basic, la cual nos sirve para definir una "subrutina", como también se les conoce a las macros, para identificarla, le asignamos un nombre, en nuestro caso le pusimos: `Mi_Primer_Macro`, nombre que:

•NO puede llevar espacios, si lo haces, te mostrara un mensaje, que por ahora, no analizaremos, pero recuerdalo para más adelante.



•NO puede iniciar con un número, si lo hace, te mostrara el siguiente mensaje de error:

•NO debe repetirse el nombre de la macro, dentro del modulo en que se declara, o sea, no puede haber dos que se llamen igual, si tratas de hacer lo anterior, al ejecutar, te mostrara el siguiente mensaje de error:



Observa la primer linea de los mensajes de las dos imágenes anteriores, las dos son iguales -**Error de sintaxis de BASIC**-, aquí, la palabra importante es; "sintaxis", que, si nos atenemos a lo que dice la Doña Real Academia Española, quiere decir:

> *sintaxis*
>
> (Del lat. *syntaxis,* y este del gr. σύνταξις, de συντάσσειν, coordinar).
>
> **1.** f. *Gram.* Parte de la gramática que enseña a coordinar y unir las palabras para formar las oraciones y expresar conceptos.
>
> **2.** f. *Inform.* Conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje de programación.
>
>

De estas definiciones, la que nos interesa es la numero dos, trata de recordarla -*Conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje de programación*-, pues en las siguientes páginas, trataremos de aprender e implementar este ***conjunto de reglas***, las cuales, te confesaré, no son tantas, ni tan complejas y como seguro habrás notado, el ***IDE*** nos informa muchas veces cuando implementamos mal, este conjunto de reglas, es decir, cuando comúnmente hacemos mal uso de la “sintaxis” de OOo Basic.

Para finalizar este capitulo, analicemos la ultima y muy sencilla linea de nuestra macro:

```
End Sub
```

Si, las dos son palabras claves del lenguaje y también, creo, son bastante explicitas, nos indican el final de nuestro "programa", "subrutina" o "macro", como las llamaremos de aquí en adelante.

Entonces, cada línea de código y, hasta donde se encuentre un salto de línea, el lenguaje lo reconocerá como una **sentencia** de OOo Basic, puedes tener varias sentencias en una sola línea, si las separas con dos puntos, como en:

```
Sub VariasSentencias()
    MsgBox "Estas es una sentencia" : MsgBox "Esta es otra sentencia"
End Sub
```

En sentencias largas, puedes usar el carácter de continuación de línea, que es el guión bajo (_), como en el siguiente ejemplo, pero recuerda que sigue siendo la misma sentencia:

```
Sub LineasLargas()
    MsgBox "Estas es una linea larga de código, se puede dividir " & _
    "usando el guión bajo, como en este ejemplo", 0 ,"Ejemplo " & _
    "de linea larga"
End Sub
```

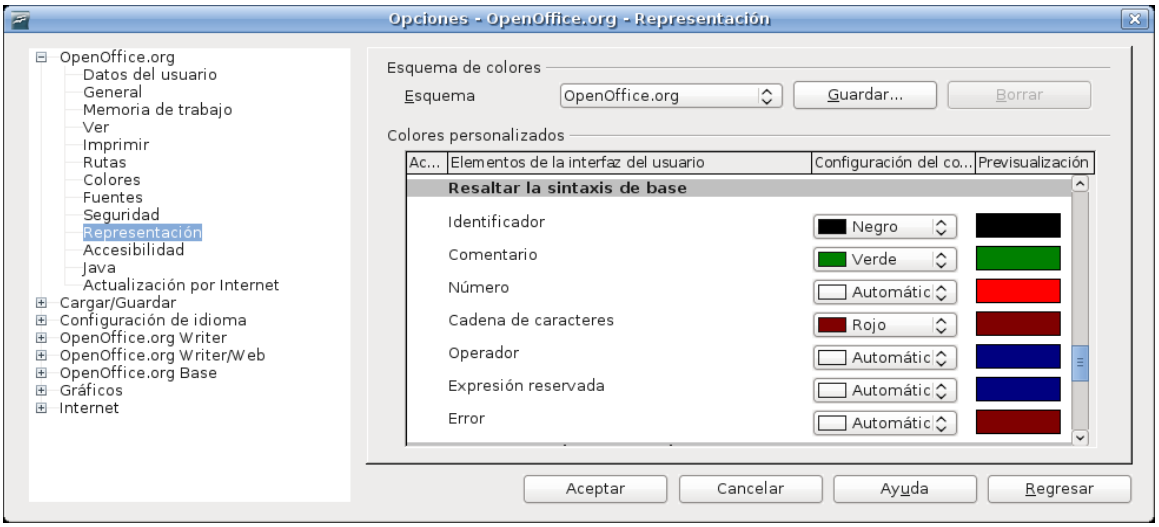
Y ahora si, para terminar este capitulo, observa que el *IDE* nos muestra diferentes colores en las palabras, esto es muy útil cuando se esta programando, pero por puro gusto y también, lo confieso, como una herencia del VBA (aunque también pesa el hecho de tener una macro ya hecha que convierte estos colores), uso la siguiente combinación de colores:

- Azul para las palabras claves
- Verde para los comentarios
- Rojo para los números
- Rojo obscuro para las cadenas de texto
- Negro para todo lo demás

Aquí un ejemplo de como se mostraría el código con esta combinación de colores:

```
Option Explicit    Rem Pruebas de comentario

Sub Pruebas(ByVal Pal As String)
' comentario
Dim intNum As Integer
     intNum = 10 + 10
     MsgBox "Prueba"    'comentario
End Sub
```

Para configurar estos u otros colores que sean de tu agrado, ve al menú *Herramientas | Opciones...* en *OpenOffice.org* escoge *Representación*, para que quede de la siguiente manera



Recuerda que esto es sólo una recomendación, tu, con el uso y la practica, decidirás si es así como te gusta, si no, no dudes ni un momento en establecer los colores que más te agraden, gusten o simplemente te den la gana que para eso es la libertad.

Esta es la de a de veras, ahora si, para terminar este capitulo, tienes que recordar que para OOo Basic, es indistinto el uso de MAYUSCULAS o minúsculas, o una combinación de ambas, es decir, nuestra macro, "debería" de funcionar igual con:

•minúsculas

```
sub mi_primer_macro
     msgbox "¡¡Hola Mundo!!"
end sub
```

•MAYUSCULAS

```
SUB MI_PRIMER_MACRO
     MSGBOX "¡¡Hola Mundo!!"
END SUB
```

•Combinación de ambas, como nuestro original, que es la forma mas común y la forma que yo usare a todo lo largo de estas notas.

```
Sub Mi_Primer_Macro
     MsgBox "¡¡Hola Mundo!!"
End Sub
```

El *IDE*, no hace una autocorrección de las palabras, no te iguala las palabras a mayúsculas o minúsculas (todavía), pero esto, te ayuda mucho a mejorar tu mecanografía, a ser mejor observador, así que:

- Practica mucho.
- Investiga mas acerca de la instrucción MsgBox, presionar {**F1**} "ayuda" mucho.
- Practica mucho.
- Envíame tus comentarios, aportes y dudas.
- Practica mucho.

4 El lenguaje OOo Basic

OOo Basic, forma parte de la familia de lenguajes Basic, comparte con ellos la mayor parte de su sintaxis, su facilidad y sencillez, no entraremos aquí en disquisiciones filosóficas de su historia o diferencia con otros "sabores" Basic, me limitare a mostrarte el uso de su sintaxis lo mas claramente que me sea posible, procurando que los ejemplos sean tu principal fuente de consulta. El *IDE*, se encarga de revisar la correcta "***sintaxis***" (recordando que sintaxis es un "conjunto de reglas") de cada línea que nosotros escribamos, por ello, a OOo Basic se le denomina un "lenguaje interpretado", porqué revisa la sintaxis, cada vez qué "ejecutamos" la macro.

Ahora ya conoces donde inicia y donde termina una macro, como nombrarla y como ejecutarla, esto es importante qué lo domines bien, pues qué de aquí en adelante, daré por entendido qué así es, por qué cada nueva instrucción qué aprendamos, la tienes qué probar "dentro" de una macro, así qué para qué no haya excusas, repasemos una vez mas esto...

La estructura básica para una macro es.

```
Sub Nombre_de_la_Macro

   REM Aquí van todas las instrucciones qué queramos

End Sub
```

Por supuesto existen más variantes, por ahora, para nuestros fines, esta es más qué suficiente, vamos a escribir nuestra segunda macro, aquí esta.

```
Sub Mi_Segunda_Macro
   'Algo más interesante
   MsgBox "¡Hola Nena!"
End Sub
```

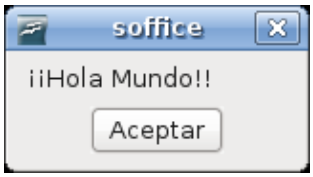
Y ya encarrilados, agrega una tercer macro.

```
Sub MiTercerMacro
   'Lo qué quieras
   MsgBox "OOo Basic es fácil y divertido"
End Sub
```

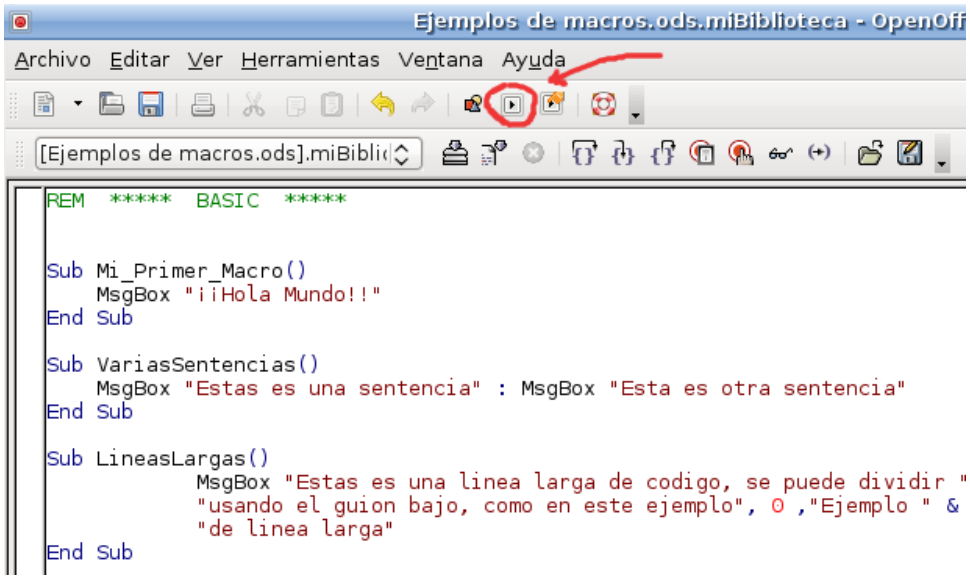
Observaciones (¿recuerdas qué íbamos a ser observadores?, ser buen observador es una excelente virtud en un programador).

En la tercer macro, en el nombre, ya no use guiones para separar las palabras, ¿es correcto?, respuesta, si, es correcto, uno de los requisitos es qué el nombre "no" lleve espacios, así qué esta bien escrito ***MiTercerMacro***, en la qué usamos la técnica denominada de "camello", creo qué es bastante obvio ¿por qué?, esta es la forma en qué de aquí en adelante, escribiré todo mi código.

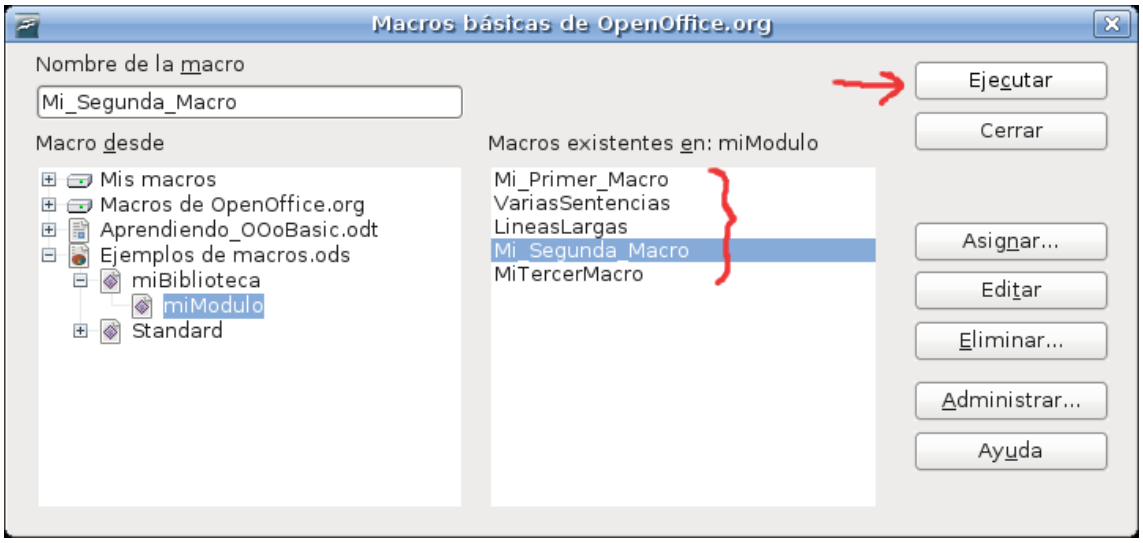
Supongo qué has ido probando cada macro, ¿verdad?, si no es así, ahora es el momento, presiona {**F5**} y observa muy bien qué sucede, ¿ya?, es muy sencilla la explicación, ya la habrás deducido. El IDE "sólo", ejecuta la primer macro que este en el módulo activo, así que si tienes las tres macros que hemos hecho, sólo se ejecutara la que se llama ***Mi_Primer_Macro***. Claro, si tienes como primer macro alguna otra, esta será la que se ejecute. Y cada vez que trates de ejecutar solamente te mostrara.



En algunas distribuciones Linux, he notado que la macro que se ejecuta al presionar **F5**, es la macro donde este el cursor de escritura en ese momento, esto me parece bien, pero todavía no esta implementado en la distribución oficial. Para ejecutar la macro que quieras, ve a la barra de herramientas y da un clic en el icono ***Seleccionar macro***.



Del cuadro de dialogo que te muestra, escoges la macro que desees y presiona el botón ***Ejecutar***.



Como diría un buen amigo -ansina sí-, ahora sí vemos lo que queremos ver:

Y la tercera también:

Otra técnica para ejecutar la macro que sea, es crear una primer macro que se llame por ejemplo: **Pruebas**, y, desde esta, "llamar" a las otras, así quedarían nuestras macros:

```
REM *****  BASIC  *****

Sub Pruebas
   Mi_Primer_Macro
End Sub

Sub Mi_Primer_Macro
   'Todo un clásico
   MsgBox "¡Hola Mundo!"
End Sub

Sub Mi_Segunda_Macro
   'Algo mas interesante
   MsgBox "¡Hola Nena!"
End Sub

Sub MiTercerMacro
   'Lo que quieras
   MsgBox "OOo Basic es fácil y divertido"
End Sub
```

Y esto, es algo que tendrás que dominar muy bien, "llamar" a una macro desde otra, observa como es tan sencillo como escribir el nombre correcto de la macro que deseamos "llamar"...

```
Sub Pruebas
   Mi_Primer_Macro
End Sub
```

Para llamar a la segunda macro seria así:

```
Sub Pruebas
   Mi_Segunda_Macro
End Sub
```

Tu haces la tercera. Lo interesante, es que puedes "llamar" a más de una macro, desde otra, por ejemplo.

```
Sub Pruebas
```

```
    Mi_Primer_Macro
    Mi_Segunda_Macro
End Sub
```

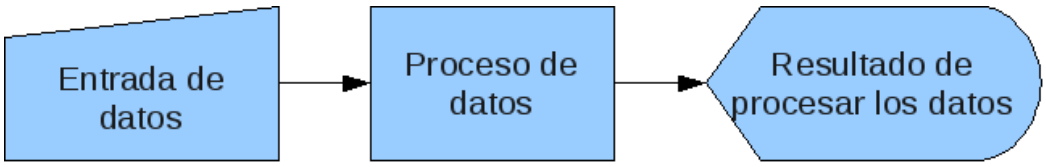
Otra forma de "llamar" a otras macros, es usando la instrucción **Call**, de la siguiente manera, el resultado es el mismo:

```
Sub Pruebas
    Call Mi_Primer_Macro
    Call Mi_Segunda_Macro
End Sub
```

Se que tendríamos que escribir el nombre de la macro a llamar para probar la que deseamos cada vez que la necesitamos, pero recuerda que sólo es de prueba, además, más adelante, cuando aprendamos más opciones de como declarar y usar macros, veras que este ultimo método de pruebas es mejor que el primero y que esto nos sirve como afirmación de lo que decía el Sr. Descartes -todo lo complejo, puede dividirse en partes simples-

4.1 Declarando y usando variables

Cada vez que programes tus *macros*, recuerda el siguiente sencillo diagrama:



Si analizas cada programa de computo que usas, veras que en mayor o menor medida, cumple el flujo anterior, cada uno de estos tres pasos los podemos dividir en muchos más, pero esa división tu mismo la irás notando y desglosando conforme profundices en este o cualquier otro lenguaje, veamos un sencillo ejemplo:

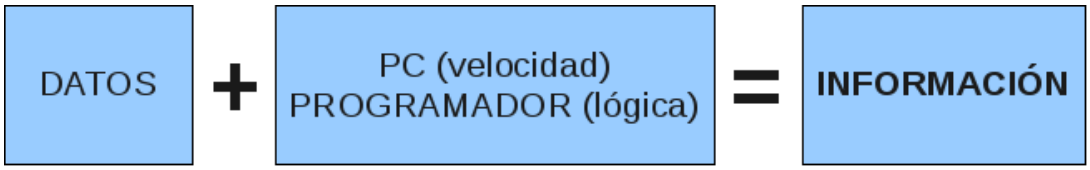
- Solicitamos que el usuario capture su nombre completo (Entrada)
- Contamos el número de letras (o palabras) que contiene (Proceso)
- Mostramos el resultado obtenido (Visualización)

Ahora, que pasa si además del nombre solicitamos que capture su fecha de nacimiento, tal vez podríamos obtener además del número de letras o palabras:

- El día de la semana en que nació
- Cuantos días lleva vivo
- Cuantos años tiene
- Cuantas vocales tiene contiene el nombre
- ¿Se te ocurren otra “información” que puedas obtener?

Observa como con sólo dos datos, podemos obtener mucha “información”, esta sencilla ecuación hace la diferencia entre un programa eficiente, de uno que no lo es, los buenos programas, manipulan eficientemente los datos para darnos “información” y entre el flujo de los

datos a la “información”, existen dos entes que son determinantes para la calidad de la información obtenida, uno es la computadora, la cual, normalmente sólo provee de “velocidad” y tu, el programador, que pone toda su inteligencia, unos mas otros menos, al servicio del proceso de convertir los datos en “información”, que como sabes, es la que permite tomar las decisiones.



No pienses que estoy desvariando, todo este “rollo” lleva un fin, veamos cual es. De acuerdo a los dos diagramas anteriores, el inicio de toda “macro” (o programa), es obtener los datos, ya sea traerlos de algún lado o solicitarlos al usuario. Recuerda que primero hay que saber caminar para después correr, así que no desesperes si ves que vamos algo lento, pero estos temas iniciales son determinantes para comprender todo lo que venga después. Entonces, retomando el sencillo ejemplo con el que empezamos este capitulo, vamos a solicitarle al usuario que nos proporcione su nombre, lo cual lograremos con la “función” **`InputBox`**, ojo con la palabra “función” ya que es importante, creo que casi cualquier persona que usa computadoras, sabe que una función, “normalmente”, retorna un valor, esto es importante, nos regresa un valor, que, en el caso de nuestra macro, tendremos que poner en algún lugar para poder tenerla disponible y para poder manipularla, el primer lugar donde tenemos que aprender a guardar datos, aunque sea de forma temporal, es en memoria, por supuesto, no en nuestra memoria, sino en la de la computadora, en la llamada memoria RAM, para ello tendremos que darle un nombre a una porción de dicha memoria, a esta porción que estamos nombrando, le llamaremos “**variable**”, otra vez, recuerdalo bien “**variable**”, algunos textos también la nombran como “marcador”, pero lo más común es llamarlo “**variable**”, precisamente por que su valor o contenido puede variar durante los procesos que haga nuestra macro, resumiendo, los datos que obtengamos los guardamos “temporalmente” en variables, con lo cual tenemos esta primer sencilla ecuación, que iremos ampliando según vayamos avanzando en nuestro curso.

Variable = Dato

Veamos un ejemplo de esto, mostrando el nombre de una novia que tuve.

```
Sub MostrarNombre
   'Asignamos un valor a una variable
   Nombre = "Nicole Kidman"
   'Mostramos el valor de la variable
   MsgBox Nombre
End Sub
```

Observa como en la linea.

```
Nombre = "Nicole Kidman"
```

Estamos asignando el “dato” **“Nicole Kidman”**, a la “variable” **Nombre,** el nombre de las variables, al igual que el nombre de las macros, no pueden llevar espacios y no pueden empezar con un número, además de no poder llevar caracteres especiales, veamos algunos ejemplos de nombres de variables:

- Direccion - Correcto
- Telefono2 - Correcto
- FechaDeNacimiento - Correcto
- Correo electronico - Incorrecto (tiene un espacio)

- 3Fax - Incorrecto (inicia con número)
- Áéíóú - Incorrecto (no se permiten caracteres especiales)
- Calle,Numero - Incorrecto (no se permiten comas ni puntos)

Como notaste, simplemente con usar el nombre correcto de una variable, OOo Basic hace uso de ella, esta forma de usar variables se conoce como, “declaración implícita”, porque al asignarle un valor por primera vez a una variable, el lenguaje reserva el espacio en memoria para su uso, pero (dicen que siempre hay un pero), esta forma de usar las variables, no es nada recomendable, conforme tus macros se hacen más grandes y complejas, el hacer uso de las variable de esta forma se vuelve (y no exagero) una verdadera locura, en contraste, es más eficiente y mucho más claro si le dices al lenguaje, desde el inicio de tu macro, que variables son las que usaras para que reserve la memoria necesaria, esta segunda forma de usar las variables se denomina, “declaración explícita”, de esta forma, nosotros somos los que le indicamos al lenguaje, que variables son las que usaremos, hay una palabra clave para ello, se llama `Dim` y su uso es muy sencillo, aquí el ejemplo.

```
Sub DeclarandoVariables
'Declaramos cuatro variables
Dim Direccion
Dim Telefono2
Dim FechaNacimiento
Dim Edad
   'Mas código
End Sub
```

De esta forma, con la instrucción `Dim`, le estamos diciendo -oye mi hermano, apartame un cachito de memoria que se llame **`Direccion`** para que yo la use-, y efectivamente, el lenguaje lo hará para nosotros. Observa que el nombre de las variables inmediatamente nos da una idea acerca del dato que guardaremos en ella, en el caso de la variable **`Direccion`**, normalmente guardara un texto, en la variable **`FechaNacimiento`** una fecha y en la variable **`Edad`** un número, de la forma como están declaradas las variables, se puede guardar cualquier “tipo” de dato (texto, numero, fecha, etc) en ella, por eso, a este tipo de variables se le conoce como “`variant`”, veamoslo con un ejemplo.

```
Sub VariablesTipoVariant
Dim UnDato

   UnDato = "Esto es un texto"       'Guardamos un texto
   MsgBox UnDato
   UnDato = 12345678                 'Guardamos un numero
   MsgBox UnDato
   UnDato = "15/01/1974"             'Guardamos una fecha
   MsgBox UnDato
End Sub
```

La recomendación es que, en vez de usar variables tipo “Variant”, le indiquemos explicitamente al lenguaje qué tipo de datos guardaremos en una variable, para ello, se usa la palabra clave `As`, junto con `Dim` y el tipo de variable, por ejemplo.

```
Sub DeclaracionExplicitaDeVariables
Dim Direccion As String  'Variable tipo texto
Dim Edad As Integer      'Variable tipo numero entero
Dim Fecha As Date        'Variable tipo fecha

   Direccion = "Parque San Andres N° 14"
   Edad = 32
   MsgBox Direccion
```

```
    MsgBox Edad

End Sub
```

Un error frecuente en programadores noveles, es declarar variables del siguiente modo.

```
Dim Nombre, Paterno, Materno As String
```

Aparentemente, las tres variables anteriores son de tipo texto (String), pero no es así, sólo la variable Materno es de este tipo, Nombre y Paterno son de tipo variante (Variant), para que las tres sean de tipo texto (String) tienen que declararse del siguiente modo.

```
Dim Nombre As String, Paterno As String, Materno As String
```

Aquí, la lista completa del tipo de variables que puedes usar en OOo Basic:

Tipo	*Descripción*	*Valores que puede contener*
`String`	Texto	Hasta 65,535 caracteres
`Byte`	Entero	Enteros entre 0 y 255
`Integer`	Entero	Enteros entre -32768 y 32767
`Long`	Entero largo	Enteros entre -2147483648 y 2147483647
`Single`	Simples	Precisión simple y coma flotante (3,402823 x 10E38 a 1,401298 x 10E-45).
`Double`	Dobles	Precisión doble y coma flotante (1,79769313486232 x 10E308 a 4,94065645841247 x 10E-324)
`Currency`	Moneda	Máximo de 15 números antes del signo decimal y cuatro después de él. Almacena valores entre -922337203685477.5808 y +922337203685477.5807
`Boolean`	Lógica	sólo puede contener valores True (verdadero) o False (falso)
`Date`	Fecha	Desde el 01-mar-200 al 31-dic-9999
`Variant`	Variante	Cualquier valor, tipo de declaración predeterminado
`Object`	Objeto	Para hacer referencias a objetos

Es importante que recuerdes que si sobrepasas estos valores, se generará un error en tiempo de ejecución que se llama "*desbordamiento*" el cual que aprenderemos a controlar más adelante.

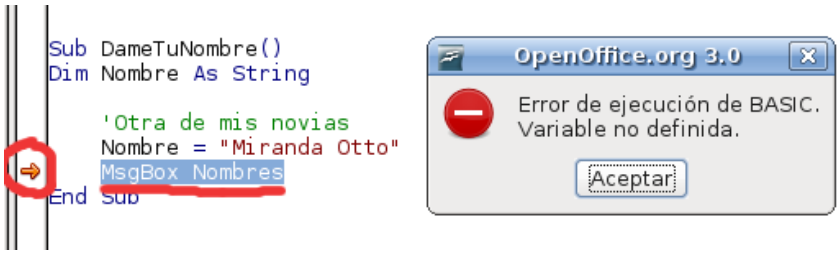
Veamos este nuevo ejemplo de una macro, asegúrate de transcribirla tal cual o mejor aún, puedes copiarla.

```
Sub DameTuNombre
Dim Nombre As String

   'Otra de mis novias
   Nombre = "Miranda Otto"
   MsgBox Nombres
End Sub
```

¿Qué pasa al ejecutarla?, efectivamente, no nos muestra nada en el cuadro de mensaje, esto es porque, para el lenguaje, la variable **Nombre**, es diferente de la variable **Nombres** y a la variable **Nombres** no le hemos asignado ningún valor todavía, a pesar de que en este ejemplo es bastante evidente nuestro error (una pequeña "**s**" de diferencia), cuando se programa y haces uso de unas cuantas decenas de variables declaradas, esto no es tan evidente a simple vista, para minimizar este tipo de errores, existe una instrucción que nos ayuda a ello, `Option Explicit`, que nos obliga a declarar con la instrucción `Dim`, cualquier variable antes de usarla, esta instrucción, cuando se use, debe hacerse como primer linea de cualquier modulo (sólo una vez por modulo), de modo que nuestra anterior macro quedara así.

```
Option Explicit

Sub DameTuNombre
Dim Nombre As String

   'Otra de mis novias
   Nombre = "Miranda Otto"
   MsgBox Nombres
End Sub
```

Vuelve a ejecutar esta macro y observa qué pasa, así es, nos muestra el siguiente mensaje.



Como ves el mensaje es muy claro: **Variable no definida**, es decir, no hemos usado la instrucción `Dim` para declararla, además, observa en la imagen (el subrayado rojo), que el *IDE* nos indica con una flecha la línea donde ocurrió el error, así que, otra recomendación que tienes que tomar como regla -siempre usa **Option Explicit** al programar-, usando esta instrucción te volverás un poco menos loco cuando busques ¿por qué? no funciona tu macro, mas adelante veremos qué otras herramientas implementa el *IDE* para ayudarnos con la localización de errores y la depuración de nuestras macros.

El inicio de este tema, mencionábamos que una macro (o programa), empieza con la solicitud de algún dato al usuario (generalmente), también veíamos que este dato o datos se guardan temporalmente en variables (generalmente), la instrucción básica para ello era.

Variable = Dato

y como ejemplo mostramos.

```
Sub MostrarNombre
   'Asignamos un valor a una variable
   Nombre = "Nicole Kidman"
   'Mostramos el valor de la variable
   MsgBox Nombre
End Sub
```

Lo cual funciona, pero no es muy versátil que digamos, es decir, siempre nos mostrará el mismo nombre, que aunque bello, creo que terminará por cansarnos. La segunda

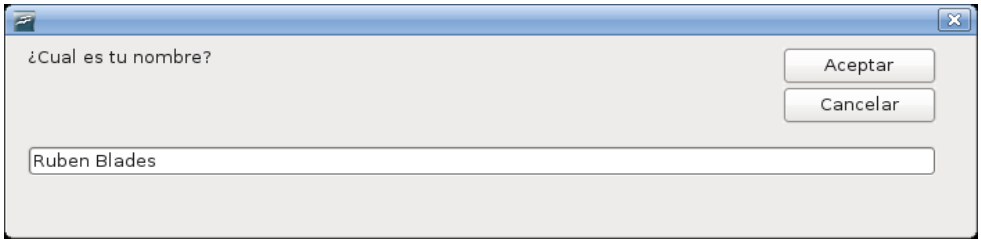
variante que veremos para asignarle un valor a una variable y que servirá como preámbulo para los siguiente capítulos es la siguiente.

Variable = Función

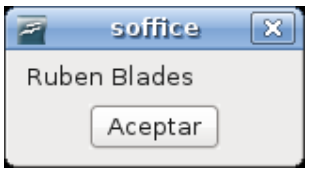
En donde **Función**, es cualquier instrucción que nos “retorne” un valor, veamos un ejemplo de la función *InputBox*.

```
Sub MiNombre
Dim Nombre As String

   Nombre = InputBox("¿Cual es tu nombre?")
   MsgBox Nombre
End Sub
```

Al ejecutar, te tendrá que mostrar el siguiente cuadro de dialogo:



Y al escribir tu nombre y dar clic en Aceptar, te mostrará lo que hayas escrito.



Para cerrar el círculo, y comprobar que sí funciona el primer diagrama de este capítulo, “**hagamos algo**” con el dato, veamos cuántas letras tiene, esto lo logramos con la función **Len**, que nos dice cuántos caracteres tiene una cadena, entonces, nuestra macro completa quedaría así:

```
Sub MiNombre
Dim Nombre As String
Dim NumeroDeLetras As Integer

   Nombre = InputBox( "¿Cual es tu nombre?" )    'Entrada de datos
   NumeroDeLetras = Len( Nombre )                'Proceso de datos
   MsgBox NumeroDeLetras                         'Visualización de resultado
End Sub
```

Observa cómo se cumple el flujo del diagrama mostrado al inicio de este capítulo, sé que es un gran simplificación pero nos es bastante útil para nuestros propósitos.

Ni modo, no se me olvida dejarte tarea:

- Practica mucho
- Investiga mas de la función InputBox.
- Practica mucho

4.2 Instrucciones y funciones en OOo Basic

¿Recuerdas que mencionamos que las palabras claves o reservadas de OOo Basic, se llaman así porque sólo OOo Basic puede hacer uso de ellas?, ¿verdad que lo recuerdas?, por ello, no puedes crear una macro que se llame `MsgBox` u `Option`, tampoco puedes declarar variables con alguna de estas palabras. Estas palabras clave, las podemos dividir en dos grandes grupos; las instrucciones y las funciones, que, como primera aproximación a una definición útil, diremos que:

Las instrucciones "hacen" algo
Las funciones "devuelven" algo

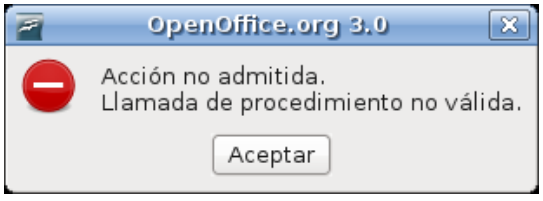
La diferencia entre una y otra, es crucial para el entendimiento y desarrollo óptimo de los siguientes capítulos. Usaremos una instrucción que ya conoces, MsgBox, para mostrarte las diferencias entre las instrucciones y las funciones, pues esta palabra clave tiene la particularidad de poder usarse como una u otra dependiendo de nuestras necesidades. Veamos la siguiente linea de código.

```
MsgBox "Estoy aprendiendo OOo Basic"
```

Estamos usando MsgBox como una instrucción, porque está "haciendo" algo, ¿qué?, nos muestra un mensaje en pantalla, ¿qué muestra?, el texto, de aquí en adelante "cadena de texto", que le indicamos entre comillas, esta cadena de texto, recibe el nombre de "parámetro" o "argumento", recuérdalo bien, recibe el nombre de "argumento". ¿Has probado a intentar ejecutar esta instrucción sin ningún argumento?.

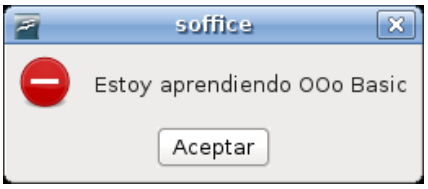
```
MsgBox
```

Efectivamente, el *IDE* nos muestra el siguiente error.



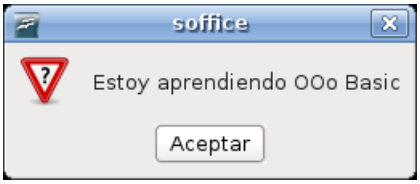
Y nos señalará con la flecha que ya conoces, la linea del error, en este caso, la linea con la instrucción `MsgBox`, con lo cual se deduce fácilmente que el argumento **Mensaje**, que es de tipo "cadena de texto" (ojo con esto, los argumentos, al igual que las variables, son de un "*tipo de dato*"), es necesario o requerido, es decir, no trabajará mientras no lo incorporemos a la linea de código, ¿necesita algún otro?, no, con este argumento es más que suficiente para que trabaje, pero eso sí, puede llevar más argumentos, por ejemplo, prueba la siguiente línea de código y observa el resultado.

```
    MsgBox "Estoy aprendiendo OOo Basic", 16
```

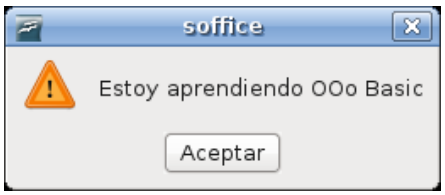


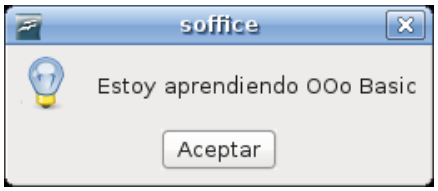
Y ahora esta:

```
MsgBox "Estoy aprendiendo OOo Basic", 32
```



Entonces, si quieres que te muestre un icono con la señal de “stop”, debes de agregarle un segundo argumento que se llama **Tipo**, el cual, es un número entero, para el primer caso un 16 y para el icono de interrogación ponemos un 32, existen otro dos iconos que puedes mostrar, el icono de exclamación (48) y el icono de información (64), que te muestro aquí:

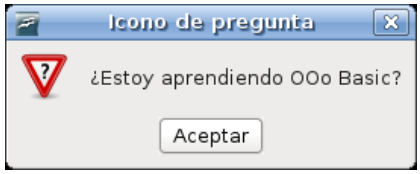




La instrucción MsgBox, tiene un tercer argumento opcional que se llama **Título**, el cual permite personalizar el titulo del cuadro de dialogo, que como notas en las cuatro imágenes anteriores, todas dicen **soffice**, compruébalo, nunca me creas completamente hasta no haberlo comprobado por ti mismo, para hacerlo, prueba las siguientes líneas de código.

```
MsgBox "Estoy aprendiendo OOo Basic", 16, "Icono de Stop"
MsgBox "Estoy aprendiendo OOo Basic", 32, "Icono de pregunta"
MsgBox "Estoy aprendiendo OOo Basic", 48, "Icono de exclamación"
MsgBox "Estoy aprendiendo OOo Basic", 64, "Icono de información"
```

Y observa el resultado, en este caso, sólo de la segunda línea:



Creo que todavía mantengo tu confianza, ¿verdad?, espero que sí. Sigamos. Estos son los tres argumentos que puede usar la instrucción MsgBox, Mensaje, Tipo y Título, por lo cual, la sintaxis completa y correcta de esta instrucción es:

MsgBox Mensaje As String, [Tipo As Integer], [Titulo As String]

Lo importante aquí es que *aprendas a distinguir la estructura o sintaxis de una instrucción*, observa cómo los *argumentos* se separan por comas y los argumentos opcionales se encierran entre corchetes, además, y muy importante, observa que se especifica el *tipo de dato* que se espera se use en el argumento, esto es muy importante, pasarle a un argumento un tipo de dato que **no** es el que espera, puede producir errores o resultados imprevisibles.

Regresemos al argumento Tipo, y veamos que también podemos establecer los botones de comando que queramos se muestren en el cuadro de dialogo, aquí está la lista completa de estas opciones:

Argumento Tipo	*Muestra los botones:*
0	Mostrar sólo el botón Aceptar.
1	Mostrar los botones Aceptar y Cancelar.
2	Muestre los botones Cancelar, Repetir y Ignorar.
3	Mostrar los botones Sí, No y Cancelar.
4	Mostrar los botones Sí y No.
5	Mostrar los botones Repetir y Cancelar.

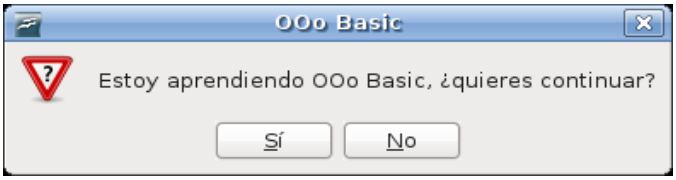
Y una muestra.

```
MsgBox "Estoy aprendiendo OOo Basic", 2, "Instrucción MsgBox"
```

Cuando quieras mostrar una combinación de botones e iconos, simplemente establece el argumento *Tipo*, como una suma de los valores que quieres mostrar, las dos lineas siguientes son equivalentes:

```
MsgBox "Estoy aprendiendo OOo Basic, ¿quieres continuar?", 4 + 32, "OOo Basic"
MsgBox "Estoy aprendiendo OOo Basic, ¿quieres continuar?", 36, "OOo Basic"
```

Que nos mostrará.



Y apuesto a que ya te hiciste la pregunta interesante, ¿cómo sabemos qué botón es el que presionó el usuario?, la respuesta es sencilla, usaremos `MsgBox` pero ahora, como función, para ello tienes que recordar que: una función devuelve un valor, el cual, para empezar, guardaremos en una variable, segundo; las funciones, al devolver un valor, también es de un tipo especifico (String, Integer, etc), por lo cual, es muy importante que la variable donde guardemos el valor sea del "**mismo tipo**" del valor que esperamos nos devuelva la función. En las funciones, los argumentos que lleve la función, van siempre entre paréntesis y, como van asignadas a una

variable, necesitamos usar el signo igual, como resultado, la estructura completa de la **función** MsgBox seria:

```
Variable As Integer = MsgBox (Mensaje As String, [Tipo As Integer], [Titulo As String])
```

Recordando, ya que es muy importante, que la variable debe ser declarada del *mismo tipo* del valor que esperamos devuelva la función, en este caso en especifico, la función MsgBox, devuelve un entero (Integer), por lo que la variable donde guardes el valor de retorno debes de declararla como tipo entera (Integer) y así para cada valor de retorno de cada función que uses, por supuesto, en la documentación de OOo Basic, normalmente viene especificado este valor de retorno para cada función, o en su defecto, espero me alcancen las ganas de mostrarte todas las funciones que usa OOo Basic, espero que sí. Lo realmente importante, es que aprendas a consultar esta documentación, es decir, viendo estructuras como las que te he mostrado, debes ser capaz de hacer uso de cualquier función o instrucción, saber qué argumentos necesita y de qué tipo, así como saber qué tipo de valor de retorno tendrás, en el caso de las funciones.

Veamos el ejemplo anterior, completo, ya con su variable de retorno.

```
Sub InstruccionesYFunciones()
Dim Respuesta As Integer

   Respuesta = MsgBox ( "Estoy aprendiendo OOo Basic, ¿quieres continuar?", 36, "OOo Basic" )
   MsgBox Respuesta
End Sub
```

Por supuesto, por ahora, no verás mucha diferencia, te mostrará el mismo cuadro de diálogo, pero la gran diferencia, es que en la variable Respuesta, tendremos el valor del botón que el usuario haya presionado, ¿como saber cual? y actuar en consecuencia, es tema de otro capítulo, así que sigue leyendo, por ahora, sólo te paso la lista completa de los valores que puede retornar la función MsgBox:

Valor de retorno	*El usuario presionó el botón*
1	Aceptar
2	Cancelar
3	Cancelar (si argumento Tipo = 2)
4	Repetir
5	Ignorar
6	Sí
7	No

No creas que está mal la lista, el botón Cancelar, puede regresar un valor 2 o 3, dependiendo de con qué otros botones se muestre.

Entre más conozcas y domines las instrucciones y funciones de OOo Basic, más rápido y productivo serás, pero recuerda que no es necesario aprenderse de memoria todo esto, lo verdaderamente importante es que "**aprendas a reconocer la sintaxis y estructura de cualquier instrucción o función**", con esto, el resto es sólo consultar la documentación del lenguaje, la ayuda del programa o libros como este, que sólo son referencia, la lógica, lo importante y relevante al programar, eso, sólo se logra con la práctica diaria, para hacer real lo que decía Don Aristóteles, -somos lo que hacemos día con día, de modo que la excelencia no es un acto sino un hábito-

Más adelante aprenderemos a crear nuestras propias instrucciones que aquí llamaremos “subrutinas” y también nuestras propias funciones, pero para ello, no debes tener duda alguna de cómo se usan las que incorpora OOo Basic.

Y un nuevo capítulo ha llegado a su fin, no olvides de:

•Practicar mucho.
•Prueba a combinar los diferentes valores del argumento **Tipo.**
•Practicar mucho.
•No debes tener dudas, este tema es muy importante para los siguientes capítulos.

4.3 Constantes – Siempre lo mismo

Hemos visto cómo guardar un dato temporalmente en una variable, pero habrá ocasiones, que un valor será constante durante toda la ejecución de una macro, el uso de constantes es muy sencillo y nos evita tener que estar recordando valores específicos, otro valor agregado de las constantes, es que, si esta llega a cambiar, sólo tenemos que cambiar su valor una vez en la sección donde la declaramos.

Para declarar una constante, usamos la palabra clave `Const` de la siguiente manera:

```
Const I = 100
```

Aunque lo más recomendable, es también establecer el tipo de constante:

```
Const PI As Single = 3.1416
```

Es una práctica común en los programadores, declarar las constantes con MAYÚSCULAS, aunque esto no es una regla, te encontrarás muy seguido con esta forma de declarar constantes. Un buen ejemplo del uso de constantes, es sustituir el uso de los enteros en el argumento *Tipo* de la instrucción `MsgBox`, visto en el capitulo anterior, veamos como:

```
Const ICONO_PREGUNTA As Integer = 32
Const BOTONES_SINO As Integer = 4

MsgBox "Estoy aprendiendo OOo Basic", ICONO_PREGUNTA + BOTONES_SINO, "OOo Basic"
```

Para nombrar a las constantes, aplican las mismas reglas vistas para nombrar a las macros y a las variables, por lo que no las repetiremos aquí. Dependiendo del tamaño de tu proyecto, es una buena práctica de programación, usar un modulo único para declarar variables y constantes, pero a fin de cuentas tu gusto y criterio son los que mandan.

4.4 Bifurcaciones – Tomando decisiones

¿Te has puesto a pensar cuántas decisiones tomas al día?, la vida esta hecha de decisiones, o eso dicen, y en las macros no nos salvamos de ellas, casi siempre hay que estar decidiendo algo y de acuerdo a esta decisión tomamos uno u otro camino. Afortunadamente en OOo Basic es mucho más sencillo que en la vida, veamos como hacerlo.

Un pregunta importante: ¿Quieres ser mi novia?, si dice que "no" ¿que hago?, pero que tal que dice que "sí", preguntémosle.

```
Option Explicit

Sub Novios
Dim Respuesta As Integer
   Respuesta = MsgBox( "¿Quieres ser mi novia?", 32 + 4, "Novios" )
   MsgBox Respuesta
End Sub
```

Copia y ejecuta la macro anterior y observa como únicamente estamos obteniendo la información del botón que selecciono el usuario: un 6 para el botón "Sí" y un 7 para el botón "No", de acuerdo a los valores mostrados en la sección 2. Ahora, actuaremos en consecuencia, dependiendo de la respuesta, para ello te presento una nueva instrucción que te acompañara (esta sí) para el resto de tu programación. Señoras y señores, con Ustedes, la instrucción **If...Then...Else**. Veamos su uso.

```
Sub Novios2
Dim Respuesta As Integer
   'La pregunta importante
   Respuesta = MsgBox( "¿Quieres ser mi novia?", 32 + 4, "Novios" )
   'Evaluamos la respuesta, recuerda, un 7 es igual No
   If Respuesta = 7 Then
         MsgBox "De lo que te pierdes"
   End If
End Sub
```

Observa que sólo logramos saber cuando se presionó el botón de "No", pero cuando se presiona el botón "Si", simplemente no lo sabemos, para solventar esto, modifica la macro anterior o crea una nueva con el siguiente código.

```
Sub Novios3
Dim Respuesta As Integer
   'La pregunta importante
   Respuesta = MsgBox( "¿Quieres ser mi novia?", 32 + 4, "Novios" )
   'Evaluamos la respuesta, recuerda, un 7 es igual No
   If Respuesta = 7 Then
         MsgBox "De lo que te pierdes"
   Else
         'Cualquier otro valor diferente del que se evalúa en la condición
         MsgBox "Que inteligente eres"
   End If
End Sub
```

Notarás al ejecutar, que ahora sí sabemos cuando se presionó cada uno de los botones de nuestro cuadro de diálogo. Debes de poner atención a algo importante, la instrucción **Else**, significa "cualquier" otro valor o caso o condición, es decir, en nuestro ejemplo, si el valor de la variable ***Respuesta*** era 7, la ejecución continúa en la línea inmediata, si la variable tiene "cualquier" otro valor o condición, entonces hace lo segundo, aquí un ejemplo.

```
Sub DiaDeLaSemana
Dim Dia As Integer
```

```
    'Solicitar un numero digamos entre 1 y 10
    Dia = InputBox( "¿Que día quieres saber (1 a 10)?" )
    If Dia = 1 Then
         MsgBox "Este día es Domingo"
    Else
         MsgBox "Este día NO es Domingo"
    End If

End Sub
```

Observa cómo "sólamente" con el valor de 1, es como se nos informa que es Domingo, cualquier otro valor nos muestra el segundo mensaje. Si quisiéramos saber qué día de la semana corresponde a otro valor, usaremos una variante de la instrucción **If...Then...Else**, veamos como.

```
Sub DiaDeLaSemana2
Dim Dia As Integer

   'Solicitar un numero digamos entre 1 y 10
   Dia = InputBox( "¿Qué día quieres saber (1 a 10)?" )
   If Dia = 1 Then
        MsgBox "Este día es Domingo"
   ElseIf Dia = 2 Then
        MsgBox "Este día es Lunes"
   Else
        MsgBox "NO sé qué día es"
   End If

End Sub
```

De esta manera, podemos hacer tantas comparaciones como queramos, [TAREA] completa la macro para que nos dé todos los días de la semana y en caso de otro valor nos informe que no es un día de la semana. Como pista, puedes usar tantos **ElseIf** como quieras. Resumiendo: la estructura general de la instrucción **If...Then...Else** es.

```
    If CONDICIÓN Then
         'Código si condición es verdadera
    [ElseIf CONDICIÓN_2 Then]
         'Código si condición 2 es verdadera
    [Else]
         'Cualquier otro valor
    End If
```

Recuerda que las opciones que están entre corchetes son "opcionales", no necesariamente tienes que usarlas, pero verás que casi siempre así será. En ***CONDICIÓN***, puede ir "cualquier" instrucción susceptible de ser evaluada como verdadera o falsa y puede ser tan sencilla como las vistas hasta ahora o compleja como el siguiente ejemplo, que por ahora, no te importe como está construida, sino tan sólo quiero mostrarte un caso "diferente" de condición, pero verás que más adelante aprenderemos a hacer.

```
ElseIf bolHayComillas = False And (strPalabra = "'" Or UCase(strPalabra) = "REM") Then

ElseIf EsPalabraClave(strPalabra) And bolHayComillas = False Then
```

Recuerda, l*a condición es cualquier valor susceptible de evaluarse como verdadero (True) o falso (False)*, cuando necesites evaluar más de una condición, tienes que relacionarlas con operadores **And** (y) y **Or** (o) que veremos más adelante.

Volviendo al ejemplo de los días de la semana, en ocasiones, sobre todo cuando se compara la misma variable varias veces, es posible usar otra instrucción de bifurcación, ahora, aprenderás a usar **Select...Case**, veamos un ejemplo.

```
Sub DiaDeLaSemana3
Dim Dia As Integer

   Dia = InputBox( "¿Qué día quieres saber (1 a 10)?" )
   Select Case Dia
         Case 1
               MsgBox "Este día es Domingo"
         Case 2
               MsgBox "Este día es Lunes"
         Case 3
               MsgBox "Este día es Martes"
         Case 4
               MsgBox "Este día es Miércoles"
         Case Else
               MsgBox "NO se que día es"
   End Select

End Sub
```

Observa qué fácil es esta instrucción. Además, puede usar varios valores a evaluar separándolos con una coma o rangos de valores junto con la palabra clave **To**, por ejemplo.

```
Sub DiaDeLaSemana4
Dim Dia As Integer

   Dia = InputBox( "¿Que día quieres saber (1 a 10)?" )
   Select Case Dia
         'Aquí evaluamos si es sábado o domingo
         Case 1, 7
               MsgBox "Por fin a descansar"
         'Y aquí si es entre semana, de Lunes a Viernes
         Case 2 To 6
               MsgBox "Qué horror, día de trabajo"
         Case Else
               MsgBox "NO sé qué día es"
   End Select

End Sub
```

¿Verdad que es fácil? Al igual que con la instrucción **If...Then...Else**, las condiciones a evaluar puede ser muy sencilla o compleja, pero las dos, son el pan nuestro de cada día en la programación, así que tenlas siempre a la mano.

4.5 Bucles – Repitelo otra vez

En ocasiones (de hecho, muy seguido) es necesario realizar el mismo paso una y otra vez, es decir, necesitamos repetir bloques de código las veces que queramos o cuando (o hasta) que se cumpla una condición específica. Para ello, vamos a conocer dos nuevas estructuras para hacer bucles o repeticiones, estas son: **For...Next** y **Do...Loop**, veamos ejemplos de cada una, te acuerdas cuando te portabas mal y la maestra te dejaba escribir cien veces -

Tengo que portarme bien-, si en eso entonces hubiese sabido programar (y conste que cuando iba a la primaria ya existían las computadoras) le hubiese entregado a la maestra el siguiente código.

```
Option Explicit

Sub PortarseBien
Dim Contador As Integer

   'Inicia el contador con los límites especificados
   For Contador = 1 To 2
         'Este es el código que se ejecuta el número de veces
         'especificado con los límites en nuestro caso 2 veces
         MsgBox "Tengo que portarme bien"
   Next

End Sub
```

Sé que esta muy claro, pero mejor lo explicamos para que no haya ninguna duda y ningún pretexto de como aplicarlo más adelante. Para hacer uso de la estructura For...Next, es necesaria una variable de apoyo, para nuestro caso, escogimos la variable Contador, con frecuencia te encontrarás que algunos programadores usan la letra I, J y subsecuentes como contadores, pero tú eres libres de elegir y usar la variable que más te plazca, por ejemplo, la letra M es muy bonita, la B también es muy linda. Observa que sólo te mostrará el mensaje dos veces, que son las indicadas por el limite inferior y superior del contador especificadas con 1 To 2, no te puse cien por que tendrías que dar cien clics para terminar la macro, pero para establecer valores más grandes, aprovecharemos para ver como “sumar” cadenas de texto, en programación a la “suma” de cadenas de texto se le conoce como “concatenar” y no es otra cosa que pegar cadenas de texto una con otra, por ejemplo.

```
Sub TuNombre
Dim Nombre As String
Dim Apellido1 As String
Dim Apellido2 As String
Dim NombreCompleto As String

   Nombre = InputBox( "¿Cual es tu Nombre?")
   Apellido1 = InputBox( "¿Cual es tu primer Apellido?")
   Apellido2 = InputBox( "¿Y el segundo?")
   NombreCompleto = Nombre + Apellido1 + Apellido2

   MsgBox "Tu nombre completo es: " + NombreCompleto

End Sub
```

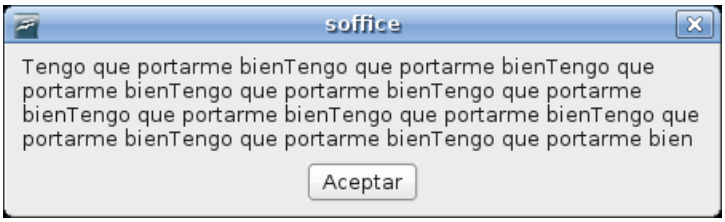
Observa cómo solicitamos el nombre y apellido y al final lo mostramos en una sola línea de texto, si bien funciona con el operador +, es más común usar el símbolo de ampersan (&) como concatenador de cadenas de texto.

```
Sub TuNombre2
Dim Nombre As String
Dim Apellido1 As String
Dim Apellido2 As String
Dim NombreCompleto As String

   Nombre = InputBox( "¿Cual es tu Nombre?")
   Apellido1 = InputBox( "¿Cual es tu primer Apellido?")
   Apellido2 = InputBox( "¿Y el segundo?")
   NombreCompleto = Nombre & Apellido1 & Apellido2

   MsgBox "Tu nombre completo es: " & NombreCompleto
```

```
End Sub
```

Por supuesto habrás notado que el nombre completo sale todo pegado, te dejo de tarea que te quede muy presentable y bonito. Ahora si, podemos hacer nuestro bucle más grande, probemos primero con 10.

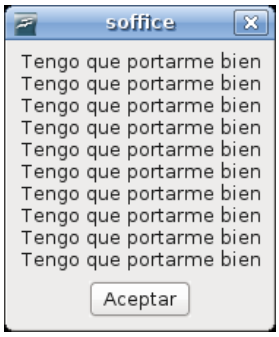
```
Sub PortarseBien2
Dim Contador As Integer
Dim Texto As String

   For Contador = 1 To 10
         Texto = Texto & "Tengo que portarme bien"
   Next
   MsgBox Texto

End Sub
```



Rápido pero no nos gusta todo el texto junto, ¿verdad?. Para mejorarlo un poco, le agregaremos un saldo de línea al final de cada una, así.

```
Sub PortarseBien3
Dim Contador As Integer
Dim Texto As String

   For Contador = 1 To 10
         Texto = Texto & "Tengo que portarme bien" & Chr(13)
   Next
   MsgBox Texto

End Sub
```

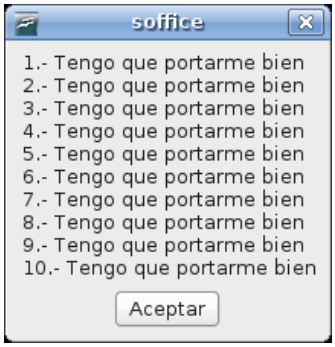
¿Recuerdas cómo decía mi amigo? -ansina sí-.



¿Y qué te parecería si lo mejoramos agregándole el número a cada línea?:

```
Sub PortarseBien4
Dim Contador As Integer
```

```
Dim Texto As String

   For Contador = 1 To 10
        Texto = Texto & Contador & ".- Tengo que portarme bien" & Chr(13)
   Next
   MsgBox Texto

End Sub
```

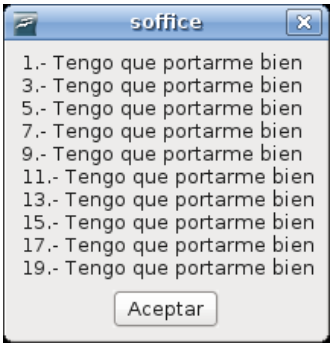


Mucho mejor, ¿no crees?. Los limites inferior y superior no necesariamente tienen que ser positivos:

```
   For Contador = -5 To 5
```

Tampoco tienen por que ir de uno en uno, cuando podemos ir de dos en dos, de tres en tres o en el rango que necesitemos.

```
Sub PortarseBien5
Dim Contador As Integer
Dim Texto As String

   For Contador = 1 To 20 Step 2
        Texto = Texto & Contador & ".- Tengo que portarme bien" & Chr(13)
   Next
   MsgBox Texto

End Sub
```



Esto se está poniendo divertido. Por supuesto observaste que le agregamos un argumento a la instrucción, en este caso **Step**, para indicarle qué salto queremos y este salto no necesariamente tiene que ser ni entero, ni positivo, ojo, mucho ojo con esto.

```
   For Contador = 100 To 10 Step -10
```

Y aquí te lo dejo para que hagas todas las pruebas que quieras y nos las cuentes en el foro o lista de correo. La siguiente macro tiene un pequeño ***error***, tu tarea es indicarme cuál es, en el capítulo de variables puedes encontrar ayuda.

```
Sub PortarseBien6
Dim Contador As Byte
Dim Texto As String

   For Contador = 1 To 256 Step 1
         Texto = Texto & Contador & ".- Tengo que portarme bien" & Chr(13)
   Next
   MsgBox Texto

End Sub
```

Como último comentario de esta instrucción, puedes usar después de la palabra clave **Next**, el mismo nombre de la variable que este usando como contador, por ejemplo.

```
   For Contador = 10 To 100 Step 5
         'Aquí va código
   Next Contador
```

Esto es sumamente útil cuando tienes varios bucles de repetición anidados, muchas veces el uso de la sangría es suficiente para distinguir unos de otros, pero en otras ocasiones, sobre todo cuando hay muchas líneas de código entre ellos, es un poco complejo saber dónde termina uno, claro que el IDE te indica si falta alguno, pero no está de más como ayuda, a fin de cuentas, de nuevo, tú y solo tú, decidirás si lo usas o no, que de eso se trata la libertad.

```
Sub BuclesAnidados
Dim co1 As Integer, co2 As Integer

   For co1 = 1 To 5
         For co2 = 1 To 10

         Next co2
   Next co1

End Sub
```

Ahora sí, de “a deveras” para finalizar, recuerda bien la estructura de esta instrucción.

```
   For Variable = Inicio To Fin [Step Salto]
         'codigo
         [Exit For]
   Next [Variable]
```

Y recuerda que lo que esta entre corchetes es opcional. Observa que debajo de la linea de comentario, hay una instrucción que no hemos visto, se trata de: **Exit For**, esta nos sirve para salir anticipadamente de un ciclo For...Next, es decir en el momento que quieras, veamos un ejemplo divertido de su uso.

```
Option Explicit

Sub Usando_Exit_For()
Dim co1 As Integer
```

```
Dim sLetra As String
Dim sVocales As String
Dim sConsonantes As String

   For co1 = 1 To 100
         sLetra = Chr( Rnd() * 25 + 65 )
         Select Case sLetra
               Case "A", "E", "I", "O", "U"
                     sVocales = sVocales & " " & sLetra
               Case "S"
                     Exit For
               Case Else
                     sConsonantes = sConsonantes & " " & sLetra
         End Select
   Next
   MsgBox "El ciclo termino en: " & co1 & Chr(13) & Chr(13) & _
               "Vocales: " & sVocales & Chr(13) & _
               "Consonantes: " & sConsonantes, 0, "Ejemplo Exit For"

End Sub
```

La función Chr(valor) nos devuelve el carácter ASCII correspondiente ha dicho valor, la letra A tiene un valor de 65 y la letra Z tiene un valor de 90, por ello es que con la línea

```
sLetra = Chr( Rnd() * 25 + 65 )
```

Obtenemos un valor aleatorio entre 65 y 90 y Chr nos devuelve dicha letra, después evaluamos si es una vocal, si es la letra S o si es una consonante, solo en el caso de que la letra devuelta sea una S (de Salida), el ciclo For...Next se interrumpe y sale con **Exit For**, al final, mostramos el valor del contador para saber si terminó el ciclo o salió antes, es mis pruebas curiosamente salió varias veces en 1 y lo máximo fue 98, es decir, nunca termino el ciclo, ya me contarás cómo te va a ti con tus pruebas. Al tipo de ciclo usado en For..Next, algunos autores le llama “determinado” pues sabemos dónde empieza y dónde termina, hay otro tipo de ciclos llamados “indeterminados”, pues dependen de una o unas condiciones para empezar o terminar el ciclo, veamos cuales son.

La instrucción **Do...Loop**, también nos sirve para hacer repeticiones, pero en vez de usar un contador, se usa una condición para saber cuándo terminar un bucle e incluso cuándo empieza o tal vez nunca empezar. Como creo que es más claro con ejemplos, a ellos nos atenemos.

```
Sub DameTuNombre
Dim Nombre As String

   Do
         Nombre = InputBox( "¿Cuál es tu Nombre?")
   Loop While Nombre = ""

End Sub
```

Analicemos línea por línea la macro anterior.

```
Sub DameTuNombre                'Iniciamos la macro

Dim Nombre As String            'Declaramos una variable de texto

Do                              'Iniciamos el bucle

Nombre = InputBox( "¿Cuál es tu Nombre?") 'Solicitamos un valor, en este caso un nombre
```

```
Loop While Nombre = ""   'Continuamos el bucle "mientras" la condición sea VERDADERA

End Sub                        'Finalizamos la macro
```

Habrás notado que de esta forma, forzamos al usuario a escribir un valor en el cuadro de dialogo, si bien es posible, recuerda que es deseable, proporcionar casi siempre una forma de cancelar o interrumpir el bucle, para probar una primer alternativa para ello, nos apoyaremos en una estructura ya vista.

```
Sub DameTuNombre2
Dim Nombre As String
Dim Respuesta As Integer

   Do
         Nombre = InputBox( "¿Cuál es tu Nombre?" )
         If Nombre = "" Then
               Respuesta = MsgBox( "Al parecer no escribiste nada, ¿realmente quieres salir?", 32 + 4,
"Salir" )
               'Recuerda que un 6 significa que el usuario presionó el botón SI
               If Respuesta = 6 Then
                     'Establecemos la variable con cualquier cosa para que termine el bucle
                     Nombre = "cualquier cosa"
               End If
         End If
   Loop While Nombre = ""

End Sub
```

Recuerda siempre de ir probando el código. Parece que funciona bien la macro anterior, pero hay una simple forma de "engañarla", prueba a introducir puros espacios y veras que sale inmediatamente, para subsanar este inconveniente, haremos uso de una función integrada en OOo Basic que nos sirve para eliminar los espacios vacíos al inicio y al final de una cadena, te presento a la instrucción **Trim**, que se usa de la siguiente forma.

```
Sub DameTuNombre3
Dim Nombre As String
Dim Respuesta As Integer

   Do
         Nombre = InputBox( "¿Cuál es tu Nombre?" )
         Nombre = Trim( Nombre )
         If Nombre = "" Then
               Respuesta = MsgBox( "Al parecer no escribiste nada, ¿realmente quieres salir?", 32 + 4,
"Salir" )
               'Recuerda que un 6 significa que el usuario presionó el botón SI
               If Respuesta = 6 Then
                     'Establecemos la variable con cualquier cosa para que termine el bucle
                     Nombre = "cualquier cosa"
               End If
         End If
   Loop While Nombre = ""

End Sub
```

Ahora sí, funciona mejor, ¿se te ocurren más alternativas por "prever" o mejorar?, lo siento, esa es tu tarea. Para continuar, veremos una variante de esta estructura que nos permite invertir la lógica de la condición, veamos como.

```
Sub DameTuNombre4
```

```
Dim Nombre As String

   Do
         Nombre = InputBox( "¿Cuál es tu Nombre?")
   Loop Until Nombre <> ""

End Sub
```

Se muy atento, observa cómo ahora le decimos "**hasta**" (con la instrucción **Until**, en negritas) que la variable Nombre, sea "diferente" de vacía, es decir, contenga algo. También, nota que la condición tiene que seguir siendo verdadera para que el bucle termine, lo que hicimos fue cambiar un argumento y el operador de la condición. Ahora, la macro completa, con **validación** y todo, ojo, recuerda esta palabra muy bien, **validación**, es el pan nuestro de cada día al programar.

```
Sub DameTuNombre5
Dim Nombre As String
Dim Respuesta As Integer

   Do
         Nombre = InputBox( "¿Cuál es tu Nombre?")
         Nombre = Trim( Nombre )
         If Nombre = "" Then
               Respuesta = MsgBox( "Al parecer no escribiste nada, ¿realmente quieres salir?", 32 + 4,
"Salir" )
               'Recuerda que un 6 significa que el usuario presionó el botón SI
               If Respuesta = 6 Then
                     'Establecemos la variable con cualquier cosa para que termine el bucle
                     Nombre = "cualquier cosa"
               End If
         End If
   Loop Until Nombre <> ""

End Sub
```

Tal vez, con justa razón, te preguntarás, ¿por qué dos formas de hacer lo mismo?. Recuerda que no todos tenemos el mismo modo de "razonar", a los fundamentalistas de cualquier extremo y color se les reconoce fácilmente por su "pensamiento" uniforme y negación a ver otro o al otro. En casi todo y más en la programación, hay más de una forma de hacer las cosas, es casi seguro que unas se adaptaran rápidamente a tu forma de razonar, otras no, pero al haber varias maneras de resolver el mismo problema, la cuestión se enriquece, así que la próxima vez que estés pronto a prejuzgar, detente un momento y trata de entender la forma "diferente" de razonar del otro, tal vez, sea un buen momento para ampliar tus horizontes.

Y hablando de variedad, observa cómo las dos variantes de la estructura vista, ejecuta el código entre **Do..Loop**, al menos una vez. Habrá ocasiones en que necesitemos o queramos que no se ejecute ni una sola vez, ¿por ejemplo?, imagina que para entrar en el bucle, necesitamos forzosamente que se cumpla una condición. Para hacer esto, simplemente pasamos la instrucción al inicio de la estructura, de la siguiente forma.

```
Sub DameTuNombre6
Dim Nombre As String
Dim Lista As String

   'Solicitamos un dato
   Nombre = InputBox( "Dame un Nombre")
   'Le quitamos los espacios sobrantes al inicio y al final
   Nombre = Trim( Nombre )
   'Iniciamos el bucle, si la variable es diferente de vacía
   Do While Nombre <> ""
         'Formamos una lista de los nombres introducidos
```

```
        Lista = Lista & " - " & Nombre
        'Solicitamos otro nombre
        Nombre = Trim( InputBox( "Dame otro Nombre") )
    Loop
    'Si la lista contiene algo lo mostramos
    If Lista <> "" Then
        MsgBox Lista
    End If

End Sub
```

Fácil, ¿verdad?. Aquí nota las dos formas en que hacemos uso de la instrucción **InputBox**, la primera de forma directa y la segunda de forma anidada. Anidar funciones, te debe de resultar muy familiar si eres un usuario medio/avanzado de cualquier hoja de calculo, conocimiento que te resultará muy útil en programación ya que es muy recurrida. Y sí, pues hay tarea, la cual consiste en que logres el mismo propósito, pero con la palabra clave **Until**, en vez de **While**.

Para finalizar este tema de los bucles, te mostraré la instrucción para finalizar anticipadamente un ciclo Do...Loop, estas palabras clave son: **Exit Do** y se usan de la siguiente manera.

```
Option Explicit

Sub Usando_Exit_Do()
Dim sClave As String
Dim sTmp As String

   sClave = "mellon"
   Do
        sTmp = Trim(InputBox("Habla amigo y entra"))
        If sTmp = "mellon" Then
            MsgBox "Y se abren las puertas de Moria"
            Exit Do
        End If
   Loop Until sTmp = "salir"

End Sub
```

Observa cómo tenemos dos formar de terminar el bucle, una, la que ya conoces en la condición final, y la otra, si el usuario escribe correctamente la palabra clave y salimos con Exit Do. **Ten cuidado con los ciclos indeterminados**, una condición mal planteada te puede dejar dentro de un ciclo infinito.

Me creerás que con los elementos vistos hasta ahora, ya se pueden hacer muchas cosas. Empieza a pensar en cómo automatizarías esos pasos tan repetitivos y tediosos que a veces haces. Para ello, ten presente siempre en tu mente el diagrama que vimos en el capitulo 4.1, “casi” todo lo que se hace en programación responde aproximadamente bien a él, como dice el dicho -la práctica hace al maestro- y sólo la práctica diaria te dará el dominio de la herramienta para obtener de ella el mejor provecho, **no te engañes**, con sólo leer estas notas o cualquiera otra documentación que encuentres o adquieras (y te lo digo por experiencia) no es suficiente para que resuelvas tus problemas informáticos, mucho menos, que te hagas llamar programador, para terminar este capitulo, recuerda lo que decía un hombre que practicaba y experimentaba mucho.

"El genio es 1% de inspiración y 99% de transpiración"
Thomas Alva Edison

4.6 Matrices – Juntos pero no revueltos

En el cual veremos cómo manipular muchos datos en una sola variable, las matrices pueden ser de mucha utilidad o convertirse en un verdadero galimatías si no se manejan correctamente, es importante que aprendas a utilizarlas, pues muchas estructuras de OpenOffice.org vienen implementadas en matrices.

La sintaxis más sencilla para una matriz es la siguiente:

```
Dim Nombre_Matriz( Tamaño As Integer ) As Tipo
```

En donde para ***Nombre_Matriz*** tiene que cumplir las mismas condiciones que para nombrar las macros y las variables vistas anteriormente. Tamaño se refiere al número de elementos que contendrá la matriz, en este caso, el indicarte que sea un tipo **Integer** (Entero) es más indicativo que restrictivo, he hecho la prueba con valores más grande que un **Integer** (Entero), pero no me imagino una macro que use tantos elementos y si lo hay, tal vez quiera decir que es hora de pasar a usar bases de datos. No he encontrado documentación al respecto del limite de este valor, en mis pruebas estoy casi seguro que esta condicionado más por el limite de la memoria RAM (random access memory) del equipo donde se ejecute que de otra causa. Te dejo a tu criterio, experiencia y pruebas, el limite de elementos a usar.

Te resultará muy fácil pensar en las matrices con una lista de valores en su presentación más sencilla, o en una tabla de filas y columnas en las mas compleja, por ejemplo, los días de la semana, una simple lista sería así:

- Domingo
- Lunes
- Martes
- Miércoles
- Jueves
- Viernes
- Sábado

Para guardar esto en una matriz, sería así.

```
Sub Ejemplo_Matrices1()
Dim miSemana(6) As String

   miSemana(0) = "Domingo"
   miSemana(1) = "Lunes"
   miSemana(2) = "Martes"
   miSemana(3) = "Miércoles"
   miSemana(4) = "Jueves"
   miSemana(5) = "Viernes"
   miSemana(6) = "Sábado"
   MsgBox miSemana( 3 )

End Sub
```

Y observa que con un simple bucle, podemos acceder a toda la matriz.

```
Sub Ejemplo_Matrices2()
Dim miSemana(6) As String
```

```
Dim co1 As Integer

   miSemana(0) = "Domingo"
   miSemana(1) = "Lunes"
   miSemana(2) = "Martes"
   miSemana(3) = "Miércoles"
   miSemana(4) = "Jueves"
   miSemana(5) = "Viernes"
   miSemana(6) = "Sábado"

   For co1 = 0 To 6
        MsgBox miSemana( co1 ), 64, "Toda la semana"
   Next

End Sub
```

Nota que la matriz empieza en 0 (cero), que es la forma predeterminada, podemos forzar a que las matrices empiecen en 1 (uno), usando la palabra clave **Option Base**, de la siguiente manera.

```
Option Explicit
Option Base 1

Sub Ejemplo_Matrices3()
Dim miSemana(7) As String
Dim co1 As Integer

   miSemana(1) = "Domingo"
   miSemana(2) = "Lunes"
   miSemana(3) = "Martes"
   miSemana(4) = "Miércoles"
   miSemana(5) = "Jueves"
   miSemana(6) = "Viernes"
   miSemana(7) = "Sábado"

   For co1 = 1 To 7
        MsgBox miSemana( co1 ), 64, "Toda la semana"
   Next

End Sub
```

Option Base “no” puede ir dentro de una macro, tiene que ir al inicio de un modulo en el área de declaración de variables. La mayoría de las estructuras de OpenOffice.org inician en 0 (cero), así que usaremos este valor de inicio como predeterminado en todo el libro. Si te es necesario un inicio o un fin diferente pues usar la siguiente variante para la declaración de matrices.

Dim Nombre_Matriz(Inicio As Tipo To Fin As Tipo) As Tipo

En donde Inicio puede ser incluso un valor negativo como en los ejemplos siguientes.

```
Sub Ejemplo_Matrices4()
Dim misNumeros(5 To 14) As Integer
Dim misColores(-5 To 4) As String

   misNumeros(5) = 123
   misNumeros(6) = 345
   misNumeros(7) = 567
   misNumeros(8) = 890
   misNumeros(9) = 135
   misNumeros(10) = 246
```

```
    misNumeros(11) = 147
    misNumeros(12) = 258
    misNumeros(13) = 369
    misNumeros(14) = 951

    misColores(-5) = "Azul"
    misColores(-4) = "Verde"
    misColores(-3) = "Morado"
    misColores(-2) = "Rojo"
    misColores(-1) = "Blanco"
    misColores(0) = "Rosa"
    misColores(1) = "Violeta"
    misColores(2) = "Gris"
    misColores(3) = "Negro"
    misColores(4) = "Oro"

    MsgBox misNumeros( 9 ), 64, "Números"
    MsgBox misColores( 0 ), 64, "Colores"

End Sub
```

Observa como en los dos casos, las matrices tienen los mismos diez elementos, en una guardamos números y en la otra, texto, pero muy bien podemos combinar datos si declaramos la matriz como **Variant**, como en el siguiente ejemplo.

```
Sub Ejemplo_Matrices5
Dim misDatos(5) As Variant
Dim co1 As Integer

   misDatos( 0 ) = "Mauricio Baeza"
   misDatos( 1 ) = 1974
   misDatos( 2 ) = "Miguel Angel 64"
   misDatos( 3 ) = "Mexico"
   misDatos( 4 ) = "D.F."
   misDatos( 5 ) = 37000
   For co1 = 0 To 5
        MsgBox misDatos( co1 ), 64, "Mis datos"
   Next

End Sub
```

Notarás que en los casos vistos hasta ahora, cuando queremos recorrer una matriz con un bucle, hemos establecido los valores superior e inferior de la matriz declarada previamente, en muchos casos, no conocerás los valores de inicio y fin de una matriz, en estos casos, OOo Basic, cuenta con dos funciones muy útiles para conocer estos valores.

```
Sub Ejemplo_Matrices6
Dim misDatos() As Integer
Dim co1 As Integer

   misDatos() = Array( 87,58,26,35,98,51,26,58,12,48,35,16 )
   MsgBox "Limite Inferior = " & Str( LBound( misDatos() ))
   MsgBox "Limite Superior = " & Str( UBound( misDatos() ))
   For co1 = LBound( misDatos() ) To UBound( misDatos() )
        MsgBox misDatos( co1 ), 64, "Mis datos"
   Next

End Sub
```

En este ejemplo aprendemos cuatro nuevas funciones de OOo Basic, **Array**, que nos permite crear una matriz introduciendo directamente los valores que contendrá, cada uno

separado por una coma, **LBound** que nos devuelve el límite inferior de una matriz y **UBound** que nos devuelve el límite superior. Usando estas dos últimas funciones, ni siquiera tenemos que enterarnos de los límites de una matriz, y por último la función **Str** que convierte en cadena de texto (String) el argumento pasado.

Puedes declarar una matriz de un determinado tamaño y usar **Array** para llenar la matriz sin que necesariamente correspondan el número de elementos, como en los siguiente ejemplos.

```
Sub Ejemplo_Matrices7
Dim misDatos1(5) As Variant
Dim misDatos2(5) As Variant
Dim misDatos3(3) As Variant
Dim co1 As Integer

   'Llenamos la primer matriz con el número exacto de elementos declarados
   misDatos1() = Array( "Lizet", 30, "Hola", 45, "Prueba", 15 )
   MsgBox "Limite Inferior = " & Str( LBound( misdatos1() )) & Chr( 13 ) & _
         "Limite Superior = " & Str( UBound( misdatos1() ))

   'Con menos elementos
   misDatos2() = Array( "Paola", 25, "Hola" )
   MsgBox "Limite Inferior = " & Str( LBound( misdatos2() )) & Chr( 13 ) & _
         "Limite Superior = " & Str( UBound( misdatos2() ))

   'Con mas elementos
   misDatos3() = Array( "Mariana", 27, "Hola", 18, "Prueba" )
   MsgBox "Limite Inferior = " & Str( LBound( misdatos3() )) & Chr( 13 ) & _
         "Limite Superior = " & Str( UBound( misdatos3() ))

End Sub
```

Al observar los valores inferior y superior que nos devuelve cada matriz, claramente notamos que la matriz se “redimensiona” con el número de elementos que tenga la función Array, no importándole el valor con el que hayamos declarado la matriz. Cambiar de tamaño o de cantidad de elementos que puede contener una matriz de forma dinámica durante la ejecución de una macro, es una tarea habitual de programación, por ello existen varias alternativas para lograr este propósito. OOo Basic cuenta con una instrucción especifica para lograr esto, se llama **ReDim** y se usa de la siguiente manera.

```
Sub Ejemplo_Matrices8
Dim misAmigos(2) As String
Dim co1 As Integer

   misAmigos(0) = "Edgar" : misAmigos(1) = "Gloria" : misAmigos(2) = "Toñito"
   For co1 = LBound( misAmigos() ) To UBound( misAmigos() )
         MsgBox Str( co1 ) & " - " & misAmigos( co1 ), 64, "Mis amigos"
   Next
   Redim misAmigos(4)
   misAmigos(3) = "Lidia": misAmigos(4) = "Anita"
   For co1 = LBound( misAmigos() ) To UBound( misAmigos() )
         MsgBox Str( co1 ) & " - " & misAmigos( co1 ), 64, "Mis amigos"
   Next

End Sub
```

Sé que eres muy observador y ya notaste que en el segundo bucle, sólo nos muestra los valores de los índices 3 y 4, esto es por que al redimensionar la matriz con **ReDim**, esta, borra los valores que hayamos introducido previamente en la matriz, por supuesto que en ocasiones desearemos mantener los valores que existan, esto se logra agregando otra palabra clave.

```
Sub Ejemplo_Matrices9
Dim misAmigos(2) As String
Dim col As Integer

   misAmigos(0) = "Edgar": misAmigos(1) = "Gloria" : misAmigos(2) = "Toñito"
   For col = LBound( misAmigos() ) To UBound( misAmigos() )
        MsgBox Str( col ) & " - " & misAmigos( col ), 64, "Mis amigos"
   Next
   ReDim Preserve misAmigos(4)
   misAmigos(3) = "Lidia": misAmigos(4) = "Anita"
   For col = LBound( misAmigos() ) To UBound( misAmigos() )
        MsgBox Str( col ) & " - " & misAmigos( col ), 64, "Mis amigos"
   Next

End Sub
```

Ahora sí, todo se mantiene con la palabra clave **Preserve.** Pero **ReDim** no sólo sirve para aumentar elementos, también sirve para disminuirlos, pero tiene el inconveniente de que ni aun usando **Preserve**, te mantiene los valores previos, como lo demuestra el siguiente ejemplo.

```
Sub Ejemplo_Matrices10
Dim misNumeros(9) As Integer
Dim col As Integer

   misNumeros() = Array( 1,2,3,4,5,6,7,8,9,10 )
   For col = LBound( misNumeros() ) To UBound( misNumeros() )
        MsgBox misNumeros( col ), 64, "Mis números"
   Next
   ReDim Preserve misNumeros(4)
   For col = LBound( misNumeros() ) To UBound( misNumeros() )
        MsgBox misNumeros( col ), 64, "Mis números"
   Next

End Sub
```

Una primer solución es la propuesta siguiente.

```
Sub Ejemplo_Matrices11
Dim misNumeros(9) As Integer
Dim mTmp() As String
Dim col As Integer

   'Llenamos la matriz con 10 números
   misNumeros() = Array( 1,2,3,4,5,6,7,8,9,10 )
   'Redimensionamos la matriz temporal
   Redim mTmp(4)
   'Pasamos los valores a la matriz temporal
   For col = LBound( mTmp() ) To UBound( mTmp() )
        mTmp( col ) = misNumeros( col )
   Next
   'Redimensionamos la matriz original
   Redim misNumeros(4)
   'Copiamos loa valores temporales
   misNumeros() = mTmp()
   'Verificamos que estén los datos
   For col = LBound( misNumeros() ) To UBound( misNumeros() )
        MsgBox misNumeros( col ), 64, "Mis números"
   Next
   'Borramos la memoria usada por la matriz temporal
   Erase mTmp

End Sub
```

¿Se te ocurre alguna otra?, seguro que sí. Como es una tarea habitual (el disminuir de tamaño una matriz y desear mantener los valores restantes), es una tarea idónea para convertirla en una subrutina o si lo deseas en una función como se ve en el tema 4.8 Funciones y subrutinas - Divide y vencerás. Aquí las dos formas y tú decides cuál usar.

Como una subrutina, en donde le pasamos la matriz a redimensionar y el nuevo tamaño que tendrá, si es mayor solo redimensiona, si es menor, copia los valores a mantener y redimensiona, si es igual la deja tal cual.

```
Sub RedimencionarMatriz( Matriz() As Variant, ByVal Tamano As Integer)
Dim mTmp() As Variant
Dim co1 As Integer

   If Tamano > UBound( Matriz() ) Then
         ReDim Preserve Matriz( Tamano )
   ElseIf Tamano < UBound( Matriz() ) Then
         ReDim mTmp( Tamano )
         For co1 = LBound( mTmp() ) To UBound( mTmp() )
               mTmp( co1 ) = Matriz( co1 )
         Next
         Redim Matriz( Tamano )
         Matriz() = mTmp()
         Erase mTmp
   End If

End Sub
```

Como una función, que hace exactamente lo mismo, excepto que devuelve el valor en vez de manipular el parámetro pasado.

```
Function FuncionRedimencionarMatriz( Matriz() As Variant, ByVal Tamano As Integer) As Variant
Dim mTmp() As Variant
Dim co1 As Integer

   If Tamano > UBound( Matriz() ) Then
         ReDim Preserve Matriz( Tamano )
   ElseIf Tamano < UBound( Matriz() ) Then
         ReDim mTmp( Tamano )
         For co1 = LBound( mTmp() ) To UBound( mTmp() )
               mTmp( co1 ) = Matriz( co1 )
         Next
         Redim Matriz( Tamano )
         Matriz() = mTmp()
         Erase mTmp
   End If
   FuncionRedimencionarMatriz = Matriz()

End Function
```

Pero para que no quede duda, veamos su uso, tanto como subrutina y como función.

```
'Usándola como subrutina
Sub Ejemplo_Matrices12
Dim misNumeros(9) As Variant
Dim mTmp() As String
Dim co1 As Integer

   misNumeros() = Array( 1,2,3,4,5,6,7,8,9,10 )
   Call RedimencionarMatriz( misNumeros(), 15)
   misNumeros(12) = 12
```

```
    MsgBox misNumeros(12)
    Call RedimencionarMatriz( misNumeros(), 5)
    'Verificamos que estén los datos
    For co1 = LBound( misNumeros() ) To UBound( misNumeros() )
         MsgBox misNumeros( co1 ), 64, "Mis números"
    Next

End Sub

'Usándola como función
Sub Ejemplo_Matrices13
Dim misNumeros(9) As Variant
Dim mTmp() As String
Dim co1 As Integer

    'Llenamos la matriz
    misNumeros() = Array( 1,2,3,4,5,6,7,8,9,10 )
    'Llamamos a la función, observa el paso de argumentos, crecemos la matriz
    misNumeros() = FuncionRedimencionarMatriz( misNumeros(), 15 )
    'Asignamos un valor al indice 12
    misNumeros(12) = 12
    'Verificamos que lo haya guardado
    MsgBox misNumeros(12)
    'Llamamos de nuevo a la función, esta vez la disminuimos
    misNumeros() = FuncionRedimencionarMatriz( misNumeros(), 5 )
    'Verificamos que estén los datos
    For co1 = LBound( misNumeros() ) To UBound( misNumeros() )
         MsgBox misNumeros( co1 ), 64, "Mis números"
    Next

End Sub
```

OOo Basic es muy noble en el manejo de las matrices, observa que sencillo es copiar una matriz en otra

MatrizDestino() = MatrizOrigen()

También, observa el uso de la palabra clave **Erase** para borrar de memoria las matrices dinámicas que ya no usemos, si bien ahora la mayoría de las computadoras disponen de muchos megas de RAM, procura mantener el control de la cantidad de memoria que usas en tus macros.

Al copiar matrices de este modo, tienes que saber que pasa algo curioso con ellas, estas, quedan vinculadas como se demuestra con la siguiente macro de ejemplo.

```
Sub CopiarMatrices()
Dim mDatos1()
Dim mDatos2()

    mDatos1 = Array(0,1,2,3,4,5)
    'Copio las matrices
    mDatos2 = mDatos1
    'Muestro el segundo valor de la segunda matriz
    MsgBox mDatos2(1)
    'Modifico este valor
    mDatos2(1)= "B"
    'Muestro el segundo valor de la primer matriz
    MsgBox mDatos1(1)
    'Vuelvo a modificar este valor en esta matriz
    mDatos1(1)= "C"
    'Muestro el valor en la otra matriz
    MsgBox mDatos2(1)
```

```
End Sub
```

Desconozco si esto es una característica o un error, pero tienes que considerarlo pues en algunos algoritmos, los valores de las matrices se usan varias veces (sobre todo en ciclos) por lo que puedes llegar a obtener resultados erróneos si no consideras esta forma de trabajar las matrices por parte de OOo Basic.

Hasta ahora, hemos visto la declaración y el uso de matrices de una sola dimensión, es posible declarar y usar matrices de más de una dimensión veamos como se hace.

La sintaxis para declarar matrices multidimensionales es la siguiente:

Dim Nombre_Matriz(Tamaño As Integer, Tamaño As Integer) As Tipo

Observa como ahora, le indicamos dos tamaños separados por una coma, un ejemplo es más ilustrativo.

```
Sub Matrices_Multidimension1
Dim mDatos( 2, 2 ) As String

   'Llenamos los datos
   mDatos( 0, 0 ) = "0-0"
   mDatos( 0, 1 ) = "0-1"
   mDatos( 0, 2 ) = "0-2"

   mDatos( 1, 0 ) = "1-0"
   mDatos( 1, 1 ) = "1-1"
   mDatos( 1, 2 ) = "1-2"

   mDatos( 2, 0 ) = "2-0"
   mDatos( 2, 1 ) = "2-1"
   mDatos( 2, 2 ) = "2-2"

   'Mostramos algunos datos
   MsgBox mDatos( 0, 0 )
   MsgBox mDatos( 1, 1 )
   MsgBox mDatos( 2, 2 )

End Sub
```

Piensa en las matrices de dos dimensiones como en una hoja de calculo formada por filas y columnas, la matriz anterior quedaría así.

	0	1	2
0	0-0	1-0	2-0
1	0-1	1-1	2-1
2	0-2	1-2	2-2

Por supuesto puedes declarar y usar matrices de más de dos dimensiones ¿hasta cuántas?, parece ser que el límite está determinado una vez más por la cantidad de memoria RAM de que disponga la computadora donde se ejecute la macro, en lo personal, matrices con más de tres dimensiones se me hacen complicadas de manipular, no obstante, creo que es más por mis límites y flojera que no lo he comprobado, así que te invito a que hagas tus pruebas con muchas dimensiones y las compartas conmigo.

Otro ejemplo divertido, llenamos una matriz de 10 x 10 elementos y la llenamos con valores aleatorios de entre 1 y 100.

```
Sub Matrices_Multidimension2
Dim mNumeros( 9, 9 ) As Integer
Dim co1 As Integer, co2 As Integer

   'Recuerda que por default los índices de las matrices empieza en cero
   For co1 = 0 To 9
         For co2 = 0 To 9
               mNumeros( co1, co2 ) = Rnd() * 100 + 1
         Next
   Next
   'Comprobamos un índice cualquiera
   MsgBox mNumeros( 4, 4 )

End Sub
```

Observa el uso de la función **Rnd()**, que nos devuelve un número aleatorio entre 0 y 1 que al ser multiplicado por el valor superior que nos interesa y sumarle el valor inferior que nos interesa, nos da dicho número aleatorio entre estos dos y sin querer tienen ahora el “algoritmo” para que devuelvas un número aleatorio entre dos valores dados.

Con las matrices de dos dimensiones podemos simular el uso de una pequeña base de datos o el uso de una hoja de calculo donde guardamos una serie de datos en columnas que se denominan “campos” y filas que se denominan “registros”, veamos un sencillo ejemplo.

```
Sub Matrices_Multidimension3
Dim mTelefonos( 2, 1 ) As String

   mTelefonos( 0, 0 ) = "Gloria"
   mTelefonos( 0, 1 ) = "12345678"

   mTelefonos( 1, 0 ) = "Antonio"
   mTelefonos( 1, 1 ) = "87654321"

   mTelefonos( 2, 0 ) = "Lidia"
   mTelefonos( 2, 1 ) = "32458924"

   MsgBox "El teléfono de " & mTelefonos( 2, 0 ) & " es " & mTelefonos( 2, 1 )

End Sub
```

Pero lo interesante y divertido es darle la oportunidad al usuario de ir capturando estos datos e ir creciendo la matriz según las necesidades de este, veamos como.

```
Sub Matrices_Multidimension4
Dim mDirectorio( 0, 1 ) As String
Dim Nombre As String
Dim Telefono As String
Dim iContinuar As Integer
Dim co1 As Integer
Dim sTmp As String

   Do
         'Solicitamos el nombre, observa el uso de la función Trim para quitar espacios sobrantes
         Nombre = Trim( InputBox( "Escribe un nombre", "Nombre" ) )
         Telefono = Trim( InputBox( "Ahora su teléfono", "Teléfono" ) )
         'Redimensionamos la matriz, pero OJO, solo la primer dimensión
         Redim Preserve mDirectorio( co1, 1 )
         'Guardamos los datos en el nuevo indice
```

```
        mDirectorio( col, 0 ) = Nombre
        mDirectorio( col, 1 ) =
        'Vamos construyendo nuestro directorio
        sTmp = sTmp & "El teléfono de " & mDirectorio( col, 0 ) & " es " & mDirectorio( col, 1 ) &
Chr(13)
        'Incrementamos nuestro contador de registros
        col = col + 1
        'Preguntamos si desea continuar
        iContinuar = MsgBox( "¿Deseas capturar mas datos?", 4 + 32, "Continuar" )
   Loop While iContinuar = 6

   'Mostramos nuestro directorio
   MsgBox sTmp

End Sub
```

Observa atentamente que hacemos un poco de trampa al ir guardando los valores introducidos por el usuario en un variable temporal (sTmp), con lo cual, no estamos muy seguros de que efectivamente los datos estén siendo guardados dentro de la matriz, para corroborarlo, de tarea tienes que modificar la macro para que:

- Te muestre tu directorio completo, llenando la variable sTmp “después” de salir del bucle, para resolver esto tienes que hacer uso de la función UBound vista más arriba, pero con una pequeña variante, le tienes que indicar de que dimensión quieres saber su limite superior, como en el ejemplo siguiente:

  MsgBox Ubound(mDirectorio, 1)

  En donde nos mostrara el índice superior de la primer dimensión y así sucesivamente, si le estableces un número superior al de dimensiones que tiene la matriz, te dará un error.

- Obliga al usuario a introducir un nombre y teléfono, es decir, que no estén vacíos estos campos
- Por ultimo, ingéniatelas para poder darle la oportunidad al usuario de borrar un registro, es decir, tienes que encontrar la manera de preguntarle al usuario si quiere borrar un registro, y por supuesto, borrarlo efectivamente.

Sí, sí, reconozco que este ultimo punto no esta nada fácil cuando uno va empezando, pero esta dentro de lo posible y confío en tus capacidades, así que, a trabajar.

Otra variante del uso de las matrices, es tener matrices de matrices, es decir, tener dentro de una matriz otra matriz, algunos autores a estas les llaman matrices escalares, veamos su uso.

```
Sub Matrices_Matrices1
Dim mDatos(2) As Variant
Dim mTmp As Variant

   mDatos(0) = Array("Perro","Gato","Oso","Tiburón","Burro")
   mDatos(1) = Array("Cedro","Pino","Caoba","Fresno")
   mDatos(2) = Array("Cobre","Plata","Manganeso","Azufre","Potasio","Fierro")

   mTmp = mDatos(0)
   MsgBox mTmp(0)
   mTmp = mDatos(1)
   MsgBox mTmp(1)
   mTmp = mDatos(2)
   MsgBox mTmp(2)
```

```
End Sub
```

Observa el uso de una variable temporal (mTmp) para asignar la matriz interna y así poder acceder a sus valores, algunas funciones y estructuras de OpenOffice.org que veremos más adelante, están implementadas de estar forma, es decir, como una matriz dentro de otra, de ahí la importancia de que las conozcas, pero ya lo notaste, su uso es muy sencillo, veamos un ejemplo divertido de los que me gustan.

```
Sub Matrices_Matrices2
Dim mDatos(3) As Variant
Dim mTmp As Variant
Dim co1 As Integer
Dim sCaracter As String
Dim sContra As String

   'Llenamos los datos, observa como tenemos cuatro grupos, letras minúsculas, letras mayúsculas,
números y caracteres especiales
   mDatos(0) = Array( "a", "b" ,"c" ,"d" ,"e" ,"f" ,"g" ,"h" ,"i" ,"j" ,"k" ,"l"
,"m","n","ñ","o","p","k","r","s","t","u","v","w","x","y","z")
   mDatos(1) = Array( "A" ,"B" ,"C" ,"D" ,"E" ,"F" ,"G" ,"H" ,"I" ,"J" ,"K" ,"L"
,"M","N","Ñ","O","P","Q","R","S","T","U","V","W","X","Y","Z")
   mDatos(2) = Array("1","2","3","4","5","6","7","8","9","0")
   mDatos(3) = Array( "\" ,"|" ,"!" ,"·" ,"$" ,"%" ,"&" ,"/" ,"(" ,")" ,"=" ,"?"
,"¿","¡","+","-","_",".",":",",",";","<",">","}","{","]","[")

   'Nuestra contraseña sera de 10 caracteres
   For co1 = 1 To 10
         'Seleccionamos aleatoriamente, UNO de los cuatro grupos
         mTmp = mDatos( CInt( Rnd() * 3 ) )
         'Seleccionamos aleatoriamente, UN elemento del grupo, observa como usamos la función Rnd y la
multiplicamos por el índice superior del grupo seleccionado
         sCaracter = mTmp( CInt( Rnd() * UBound(mTmp) ) )
         'Vamos juntando los caracteres de la contraseña
         sContra = sContra & sCaracter
   Next
   'Mostramos la contraseña
   MsgBox "Tu contraseña es: " & sContra

End Sub
```

Analízala y verás que no es tan complejo como parece y sí, lo siento, tienes tarea, modifica la macro de modo que le pregunte al usuario de cuántos caracteres quiere su contraseña, ponle un rango mínimo y máximo que pueda escoger, digamos entre 5 y 50 caracteres

Una segunda forma de acceder a los valores de una matriz de matrices, es usando un doble índice como nos muestra el siguiente ejemplo.

```
Sub Matrices_Matrices3()
Dim mDatos(1)
Dim co1 As Integer, co2 As Integer

   mDatos(0) = Array(1,2,3,4,5,6,7,8,9,0)
   mDatos(1) = Array(11,12,13,14,15,16,17,18,19,10)

   For co1 = 0 To 1
         For co2 = 0 To 9
               MsgBox mDatos (co1) (co2)
         Next
   Next

End Sub
```

Es muy importante que siempre valides los rangos de las matrices (UBound y LBound) cuando accedes por medio de sus índices, si el índice no existe te dará un error en tiempo de ejecución. También es importante que notes la diferencia entre una matriz multidimencional y una matriz de matrices.

Y sin querer tenemos un nuevo capitulo completo, bueno, casi completo, a estas alturas ya te habrás dado cuenta de la cantidad de variantes que se pueden tener con unas cuantas instrucciones de OOo Basic por lo cual, dar por completo un tema es un poco aventurado, pero para fines prácticos, sí, este capitulo, lo damos por concluido, queda a tu curiosidad e imaginación buscarle sus aplicaciones prácticas y las no tan prácticas, que a veces, son las más divertidas, aquellas que no sirven para nada. ¡Feliz programación!

4.7 Tipos personalizados

Este tipo de variables nos permite crear una estructura de datos llamada registro con variable de diferentes tipos, usamos las palabras clave Type y End Type, veamos un ejemplo.

```
Option Explicit

'Nombre del registro
Type Contacto
   'Contenido del registro
   Nombre As String
   Edad As Integer
   Ingreso As Date
End Type


Sub TiposPersonalizados1()
Dim oContacto As Object

   'Creamos un objeto del tipo Contacto
   oContacto = CreateObject( "Contacto" )
   'Llenamos sus datos
   With oContacto
         .Nombre = "Lizette Avila"
         .Edad = 35
         .Ingreso = DateSerial(2005,1,15)
   End With
   'Mostramos sus datos
   MsgBox oContacto.Nombre & " - " & oContacto.Edad & " años - ingreso el " & oContacto.Ingreso

End Sub
```

Los tipos personalizados no puedes declararlos dentro de macros, hay que hacerlo en la sección de declaraciones de un módulo. Otra forma de usarlos es la siguiente.

```
Type Producto
   Cantidad As Single
   Descripcion As String
   Precio As Single
   Importe As Double
End Type

Sub TiposPersonalizados2()
```

```
Dim oProducto As New Producto

   With oProducto
         .Cantidad = 2.5
         .Descripcion = "Horas de servicio"
         .Precio = 200
         .Importe = .Cantidad * .Precio
   End With

   MsgBox oProducto.Cantidad & Chr(10) & _
               oProducto.Descripcion & Chr(10) & _
               oProducto.Precio & Chr(10) & _
               oProducto.Importe
End Sub
```

Por último, puedes declarar matrices que contengan tipos personalizados.

```
Type Direccion
   Calle As String
   Numero As Integer
   Colonia As String
End Type

Sub TiposPersonalizados3()
Dim oDireccion() As New Direccion
Dim sCalle As String
Dim iNumero As Integer
Dim sColonia As String
Dim bSalir As Boolean
Dim co1 As Integer

   Do
         'Solicitamos los datos
         sCalle = InputBox( "Calle" )
         iNumero = Val(InputBox( "Número" ))
         sColonia = InputBox( "Colonia" )
         'Si falta cualquier dato salimos
         If sCalle = "" Or iNumero = 0 Or sColonia = "" Then
               bSalir = True
         Else
               'Redimensionamos la matriz
               Redim Preserve oDireccion( co1 )
               'Vaciamos los datos
               With oDireccion(co1)
                     .Calle = sCalle
                     .Numero = iNumero
                     .Colonia = sColonia
               End With
               co1 = co1 + 1
         End If
   Loop Until bSalir

   'Mostramos los datos capturados
   For co1 = LBound( oDireccion ) To UBound( oDireccion )
         MsgBox oDireccion(co1).Calle & Chr(10) & _
               oDireccion(co1).Numero & Chr(10) & _
               oDireccion(co1).Colonia
   Next

End Sub
```

4.8 Ámbito de variables – Ahora me ves, ahora no

Como todo en esta vida, las variables también se crean y fenecen, unas duran más otras menos, pero todas lo hacen, la buena noticia es que nosotros decidimos cuándo y cuánto, lo cual aprenderemos en este capítulo. El lugar donde declares una variable y cómo lo hagas, determinara su "visibilidad" y tiempo de vida, entendiendo por tiempo de vida, el tiempo que está disponible para poder acceder y manipular su contenido, a esta visibilidad y tiempo de vida se le conoce como ámbito de la variable, el nivel más bajo y básico ya lo conoces pues, si has seguido estos apuntes, los has venido usando desde el inicio de estas notas, veamos cual es.

4.8.1 Ámbito Local

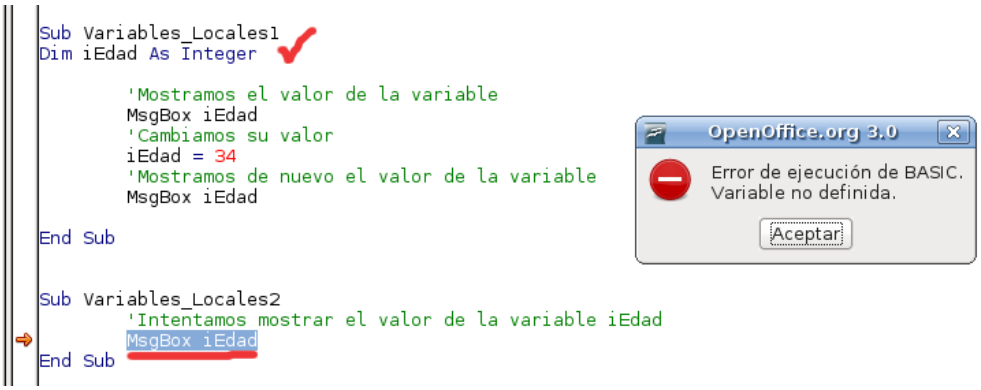
Estas variables son las que se declararan dentro del cuerpo de una macro o función y se crean al invocar a esta y se destruyen al finalizar, como en:

```
Option Explicit

Sub Variables_Locales1
Dim iEdad As Integer

   'Mostramos el valor de la variable
   MsgBox iEdad
   'Cambiamos su valor
   iEdad = 34
   'Mostramos de nuevo el valor de la variable
   MsgBox iEdad

End Sub
```

Ejecuta varias veces la macro anterior para que observes cómo "siempre" el valor inicial de la variable es cero, para comprobar que efectivamente, sólo esta macro puede "ver" a la variable iEdad, crea una segunda macro desde donde intentemos usar esta variable.

```
Sub Variables_Locales2
   'Intentamos mostrar el valor de la variable iEdad
   MsgBox iEdad
End Sub
```

Por supuesto, el IDE de sólo te mostrará un error, si tienes la precaución de usar la palabra clave **Option Explicit**, lo cual, te recomiendo que nunca olvides de usar, te evitará muchos dolores de cabeza. Si no estas usando Option Explicit, la macro anterior no te dará un error "visible", pero siempre te mostrará un cero como resultado de mostrar la variable. Entonces, si ejecutas la segunda macro, "después" de agregar las palabras clave Option Explicit a nivel de módulo,te tiene que dar un error como el de la siguiente imagen.



Observa la flecha en el margen izquierdo que nos indica la línea donde se produjo el error y observa también el mensaje que es muy claro, no hemos definido la variable para "esa macro" **dentro** de esa macro, en resumen, ***todas la variables declaradas dentro de macros, tienen ámbito local***. Veamos un ejemplo más, ojo, tienes que ejecutar la macro que se llama Variables_Locales3.

```
Option Explicit

Sub Variables_Locales3
Dim sTmp As String

   sTmp = "Aquí no soy nadie"
   Call Soy_Filosofo
   Call Soy_Divertido
   MsgBox sTmp

End Sub

Sub Soy_Filosofo
Dim sTmp As String

   sTmp = "Ahora soy un filosofo"
   MsgBox sTmp

End Sub

Sub Soy_Divertido
Dim sTmp As String

   sTmp = "Ahora soy divertido"
   MsgBox sTmp

End Sub
```

Tienes que ser muy observador para que notes cómo la variable toma el valor asignado en cada macro, cada una es diferente y se inicializa y termina con la macro donde se declara. Copia las dos macros siguientes y ejecuta la que se llama ***Variables_Locales4*** varias veces.

```
Option Explicit

Sub Variables_Locales4

   Call Contar()
```

```
End Sub

Sub Contar
Dim iContador As Integer

   iContador = iContador + 1
   MsgBox iContador

End Sub
```

Siempre lo mismo, ¿verdad?, es decir, siempre te muestra el valor 1, ¿que pasará si llamamos a la macro varias veces como en el ejemplo siguiente?.

```
Option Explicit

Sub Variables_Locales4

   Call Contar()
   Call Contar()
   Call Contar()

End Sub
```

Sigue igual, ¿verdad?, vamos a hacer un pequeño cambio a la macro ***Contar***, específicamente en la declaración de la variable y volvemos a ejecutar la macro ***Variables_Locales4***, pero esta segunda versión donde llamamos varias veces a la macro ***Contar***.

```
Option Explicit

Sub Variables_Locales4

   Call Contar()
   Call Contar()
   Call Contar()

End Sub

Sub Contar
Static iContador As Integer

   iContador = iContador + 1
   MsgBox iContador

End Sub
```

¿Lo notaste?, observa cómo declaramos la variable *iContador* en la macro Contar, pero ahora, en vez de usar la palabra clave **Dim**, usamos la palabra clave **Static**, con lo que le estamos indicando que conserve el valor entre las llamadas a la macro, este valor lo conservará mientras se ejecute la macro que llamó a la macro donde está declarada la variable Static, como podrás demostrarlo, ejecutando varias veces la macro ***Variables_Locales4***, el valor máximo siempre es tres, en nuestro caso, por que llamamos a la macro Contar tres veces y esta, incrementa el valor de la variable una unidad cada vez. OJO, aunque estemos declarando la variable iContador como Static, dentro de la macro Contar, no por ello deja de ser una variable de ámbito local, como puedes comprobarlo fácilmente tratando de mostrar su valor desde la macro ***Variables_Locales4***, el siguiente cambio en la macro te debería dar un error de -variable no definida-.

```
Option Explicit
```

```
Sub Variables_Locales4

   Call Contar()
   Call Contar()
   Call Contar()
   'La siguiente linea “debería” darte un error
   MsgBox iContador

End Sub
```

¿Verdad que es muy divertido?, ahora, le haremos un pequeño cambio a la macro ***Variables_Locales4***, para que cuente hasta que le digamos que “No”, si lo prefieres, puedes crear otra macro para que la anterior quede tal cual. Espero que lo hayas notado, pero contínuamente te invito a experimentar y a que tomes tus propias decisiones y te forjes tus propios criterios para tu programación, aquí como en la vida, dice el maestro Savater, -nos vienen bien las enseñanzas de los maestros, pero al final, estamos solos para decidir-, pues eso, tú decides.

```
Option Explicit

Sub Variables_Locales5
Dim iRespuesta As Integer

   Do
         Call Contar()
         iRespuesta = MsgBox( "¿Continuar contando?", 4 + 32, "Continuar")
         'Recuerda que si el usuario presiona Si, el valor que devuelve MsgBox es 6
   Loop While iRespuesta = 6

End Sub
```

4.8.2 Ámbito Privado

Este segundo nivel, corresponde a las variables declaradas en la cabecera de un modulo usando la palabra clave **Private**, recuerda que la cabecera es el inicio de un modulo, donde “siempre” tienes que agregar la otra importante palabra clave **Option Explicit**, para que nos avise siempre si no hemos declarado una variable, esto, te recuerdo, no es obligatorio, pero te aseguro que lo agradecerás más de una vez. Observa el ejemplo siguiente:

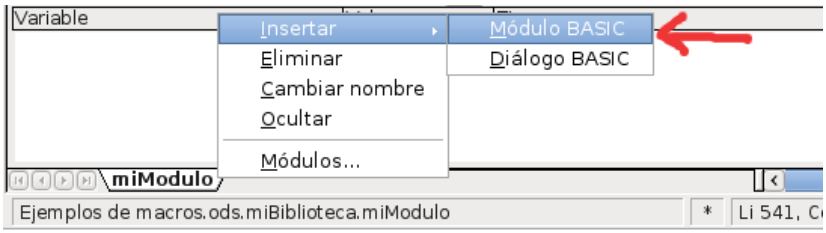
```
Option Explicit
'Observa que en vez de Dim estamos usando Private
Private intContador As Integer

Sub Sumar_y_Restar
   'Llamamos a la macro Sumar
   Call Sumar()
   'Mostramos el valor de la variable
   MsgBox intContador
   'Llamamos a la macro Restar
   Call Restar()
   'Mostramos de nuevo el valor de la variable
   MsgBox intContador

End Sub

Sub Sumar

   'Aquí aumentamos la variable en dos unidades
   intContador = intContador + 2

End Sub
```

```
Sub Restar

   'y aquí restamos a la variable una unidad
   intContador = intContador - 1

End Sub
```

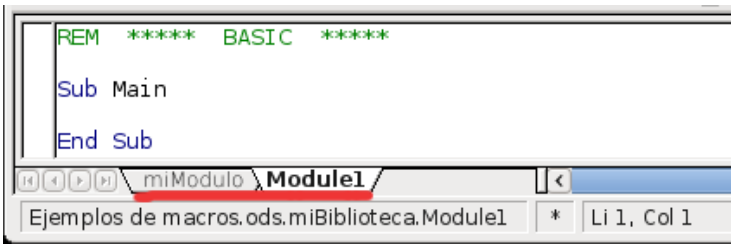
Nota como ahora, la variable *intContador*, es "visible" y manipulable por cualquier macro o función declarada en el modulo, **ojo, la teoría dice que estas variables, solo son visibles para el modulo donde se declaran, pero en mis pruebas TAMBIEN son visibles en otros módulos**, antes de demostrarte esto, veamos una variante de la macro anterior para seguir sumando y restando.

```
Option Explicit
Private intContador As Integer

Sub Sumar_y_Restar2
Dim iRes As Integer

   Do
         iRes = MsgBox( "Para sumar presiona SI" & Chr(13) & "Para restar presiona NO" & _
         Chr(13) & "Para salir presiona CANCELAR", 3, "Sumar y Restar")
         Select Case iRes
               Case 6
                     Call Sumar()
               Case 7
                     Call Restar()
         End Select
         MsgBox intContador
   Loop Until iRes = 2

End Sub
```

Ahora sí, vamos a adelantarnos un poquito en los temas y veremos cómo agregar un segundo módulo a nuestra biblioteca para agregar más macros y demostrar cómo las variables declaradas a nivel modulo con Private son visibles por cualquier otro modulo, para ello, da un clic con el botón secundario de tu ratón (normalmente el derecho) sobre la etiqueta del modulo actual donde estés trabajando, esta acción te tiene que mostrar un menú contextual, de este, seleccionas *Insertar* y después *Modulo BASIC* como te muestro en la siguiente imagen.



Tienes que ver el nuevo módulo al lado del anterior, como sé que eres observador, notarás que en el menú contextual anterior, tienes la opción de cambiar el nombre del módulo, te dejo a tu criterio que lo hagas, por ahora, nos basta tener dos módulos para pruebas como en:



Nota que al insertar un modulo, de forma predeterminada, este agrega la declaración de una nueva macro, vamos a renombrarla y trataremos de acceder a la variable intContador declarada y usada en el modulo uno, veamos que pasa:

```
Option Explicit

Sub MostrarValor()

   MsgBox intContador

End Sub
```

Nota que la macro anterior está en el módulo 2 y al ejecutarla nos muestra el valor de la variable, que es cero, pues recuerda que el valor inicial de toda variable numérica es cero, pero observa como “no” nos da error, como dice la teoría, pues toma la declaración de la variable del modulo 1. Volvamos al modulo 1, copia y ejecuta la siguiente macro.

```
Option Explicit
Private intContador As Integer

Sub ValorVariable
   intContador = 100
   'Llamamos a la macro del modulo 2
   Call MostrarValor()
End Sub
```

Creo que es bastante claro que la variable se puede manipular y acceder desde cualquier otro modulo, aun y cuando se declare con Private, lo cual espero se arregle en futuras versiones, pues de este modo, usar Private y Dim es indistinto y el siguiente tema no tiene razón de ser, incluso, en mis pruebas, la variable es accedida desde otras bibliotecas del archivo. Te agradeceré si haces tus pruebas y si notas algo diferente o si te funciona como “debería” me lo hagas saber.

4.8.3 Ámbito de Dominio Publico

En este ámbito, se supone, que las variables declaradas a nivel cabecera de modulo usando la palabra clave Dim, son visibles para cualquier modulo de la biblioteca, pero en mis pruebas es visible hasta por los módulos de otras bibliotecas del archivo y se comporta exactamente del mismo modo que las variables de ámbito privado vistas en el tema anterior, por lo que no lo repetiré aquí, solo esperemos que pronto se arregle y funcione como dicen que debe hacerlo.

```
Option Explicit
Dim intContador As Integer
```

4.8.4 Ámbito Global

Este ámbito es igual al de dominio publico en cuanto a su visibilidad, pero difiere en cuanto a su función, pues esta guarda su ultimo valor, aun y cuando se termina la macro. Para que una variable sea de ámbito global, debe ser declarada en la cabecera de un módulo con la palabra clave Global de la siguiente manera.

```
Option Explicit
Global intgContador As Integer

Sub Variable_Global()

   intgContador = intgContador + 100
   MsgBox intgContador

End Sub
```

Al ejecutar varias veces la macro anterior, notarás cómo guarda el valor, aun y cuando ya terminó la ejecución de la macro. Su uso podría compararse en función a las variables de ámbito local, declaradas con la palabra clave Static, pero a nivel global de todas las macros del archivo.

Y con esto terminamos un tema más, la recomendación para este tema es procurar usar lo más posible variables de ámbito local y después de ámbito publico, esto es por que conforme ya no uses una variable, al ser de ámbito local, esta se destruye y se libera la memoria usada por ella al concluir la macro donde se use. Por supuesto esto no es restrictivo, solo tu práctica y experiencia te dirán cual es la mejor opción. El decidir qué variables usar y con qué ámbito no es un tema menor y cuando uno va empezando se convierte casi en un problema existencial, pero no te preocupes, esperemos que con los muchos ejemplos que mostramos, te sea más fácil aprender a decidir lo mejor en cada caso.

4.9 Funciones y subrutinas – Divide y vencerás

"Todo lo complejo puede dividirse en partes simples"
Rene Descartes

En ocasiones, las funciones e instrucciones incorporadas del lenguaje, no son suficientes para resolver algún problema planteado o su complejidad nos obliga a pensar en otra alternativa, en otras ocasiones, el código se vuelve tan largo que se vuelve difícil de leer, analizar o mejorar. En estos casos, como en otros más que tal vez se te presenten, recurrimos a crear nuestras propias funciones y subrutinas. En este capítulo veremos cómo declararlas y usarlas.

En el tema 4.2 Instrucciones y funciones en OOo Basic , vimos algunos conceptos que nos serán muy útiles en este, comencemos con los ejemplos y después las explicaciones, copia y ejecuta la siguiente macro.

```
Option Explicit

Sub MostrarMensaje1()

   MsgBox "Estoy aprendiendo OOo Basic", 48, "Aprendiendo OOo Basic"
   MsgBox "Es fácil y divertido", 48, "Aprendiendo OOo Basic"
   MsgBox "Ya voy a medio camino", 48, "Aprendiendo OOo Basic"

End Sub
```

Sí, no tiene nada de extraordinario, pero nos resultara muy útil para nuestros propósitos. Vamos a suponer una segunda macro similar a la anterior.

```
Sub MostrarMensaje2()

   MsgBox "Es un poco tarde", 48, "Aprendiendo OOo Basic"
   MsgBox "Ya tengo sueño", 48, "Aprendiendo OOo Basic"
   MsgBox "Solo acabamos este tema", 48, "Aprendiendo OOo Basic"

End Sub
```

Ahora tenemos dos macros que hacen cosas muy similares, y sigamos con nuestras suposiciones, supongamos que de nuestros mensajes, el icono mostrado y el titulo del cuadro de dialogo, siempre son los mismos, por lo que lo único que cambia es la cadena mostrada, este caso es idóneo para que hagamos una subrutina, que no es más que una macro creada por nosotros a la que comúnmente (no es obligatorio) se le pasa un/unos parámetro(s) o argumento(s) y realice una tarea. Es decir, "haga" algo. Copia la siguiente macro y modifica las dos primeras macros para que queden de la siguiente manera.

```
Option Explicit

Sub MostrarMensaje3()

   Call MuestraMensaje( "Estoy aprendiendo OOo Basic" )
   Call MuestraMensaje( "Es fácil y divertido" )
```

```
    Call MuestraMensaje( "Ya voy a medio camino" )
    Call MuestraMensaje( "Es un poco tarde" )
    Call MuestraMensaje( "Ya tengo sueño" )
    Call MuestraMensaje( "Solo acabamos este tema" )

End Sub

Sub MuestraMensaje(Mensaje As String)

    MsgBox Mensaje, 48, "Aprendiendo OOo Basic"

End Sub
```

Lo importante es que aprendas como llamamos a una macro con argumentos y es muy importante que los argumentos pasados sean del mismo tipo de los esperados, observa como la macro ***MuestraMensaje*** "necesita" un argumento llamado Mensaje que es de tipo String, entonces, al llamarla, "necesitamos" pasarle una variable, no importa el nombre, sino que sea de tipo String. También observa que usamos la palabra clave Call para llamar a la macro, esto no es requisito, las siguiente líneas son equivalentes:

```
Sub MostrarMensaje3()

    Call MuestraMensaje( "Estoy aprendiendo OOo Basic" )
    MuestraMensaje( "Estoy aprendiendo OOo Basic" )
    MuestraMensaje "Estoy aprendiendo OOo Basic"

End Sub
```

Observa cómo en la primera usamos Call y paréntesis, en la segunda omitimos Call y en la tercera quitamos Call y los paréntesis. En lo personal el uso de Call solo lo uso para saber que es una subrutina personalizada, queda a tu criterio su uso u omisión. El uso de paréntesis como habrás notado también es opcional, como muchas instrucciones que usa OpenOffice.org hacen uso de los paréntesis, también, en lo personal, hago uso de ellos, pero ya sabes, tienes la última palabra.

Ahora, veremos un tema importante: a una subrutina, se le pueden pasar los argumentos o parámetros de dos maneras, una es por valor y otra por referencia, la diferencia es muy sencilla pero es de suma importancia saber la diferencia. Cuando pasamos los argumentos por valor, en realidad lo que se hace es pasarle una "copia" del valor de la variable, por lo cual, el valor de este argumento solo se puede usar "dentro" de la subrutina que la usa, en cambio, cuando los argumentos se pasan por referencia, lo que estamos haciendo es pasarle la "ubicación" de la variable en la memoria, por lo cual, podemos modificar su valor "dentro" de la subrutina, comprobémoslo con un ejemplo.

```
Option Explicit

Sub PasoPorReferencia()
Dim sMensaje As String

    'Asignamos una cadena a la variable
    sMensaje = "La travesía de mil kilómetros comienza con un paso"
    'Llamamos a la subrutina y le pasamos el argumento
    Call CambiaValor( sMensaje )
    'Mostramos la variable con el nuevo valor, cambiado en la subrutina
    MsgBox sMensaje

End Sub

Sub CambiaValor(Cadena As String)
```

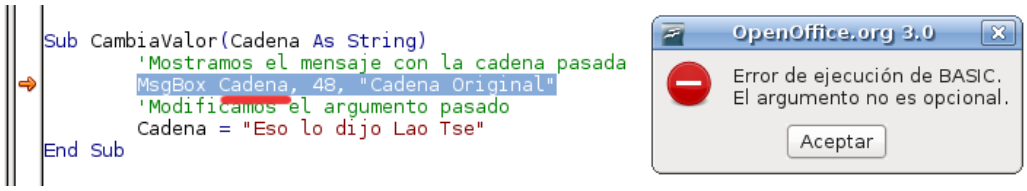
```
    'Mostramos el mensaje con la cadena pasada
    MsgBox Cadena, 48, "Cadena Original"
    'Modificamos el argumento pasado
    Cadena = "Eso lo dijo Lao Tse"

End Sub
```

Observa cómo en este caso, modificamos el argumento pasado y el cambio se ve reflejado en la variable origen, no así, en el siguiente ejemplo.

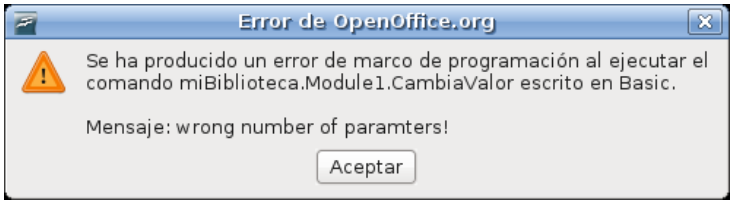
```
Option Explicit

Sub PasoPorValor()
Dim sMensaje As String

   'Asignamos una cadena a la variable
   sMensaje = "El trabajo es el refugio de los que no tienen nada que hacer"
   'Llamamos a la subrutina y le pasamos el argumento
   Call NoCambiaValor( sMensaje )
   'Mostramos la variable que nos muestra lo mismo, es decir
   'no hemos podido cambiar su valor en la subrutina
   MsgBox sMensaje

End Sub

'Observa el cambio en la declaracion del argumento usando ByVal
Sub NoCambiaValor( ByVal Cadena As String )

   'Mostramos el mensaje con la cadena pasada
   MsgBox Cadena, 48, "Cadena Original"
   'Intentamos modificar el argumento pasado
   Cadena = "Eso lo dijo Oscar Wilde"

End Sub
```

Ahora, la variable origen quedó intacta, pero observa la diferencia en la declaración de la subrutina, para pasar un argumento por valor, tienes que usar la palabra clave **ByVal** “antes” del nombre del argumento. Por default, salvo que se indique lo contrario, los argumentos se pasan por referencia. Una pregunta que nos ayuda a saber si declaramos una variable por valor o referencia es: ¿necesito manipular su valor en la subrutina?, si la respuesta es sí se pasa por referencia, si es no, se pasa por valor. He observado que hay dos criterios encontrados en el uso de los argumentos por valor o referencia, algunos programadores opinan que todos los argumentos deberían pasarse por referencia, pues al pasar la dirección de la variable en memoria, no estamos usando más memoria a este criterio podría argumentarse que los equipos actuales tienen mucha más memoria, otros dicen que todos los argumentos deberían pasarse por valor, pues una subrutina tiene que ser como una unidad completa por sí misma, y la memoria usada se libera al finalizar la subrutina. Las preocupaciones del uso de la memoria, por el momento, se las vamos a dejar a grandes proyectos, como OpenOffice.org por ejemplo, donde están involucradas millones de líneas de código, para nuestros fines prácticos, tú decides, si pasas los argumentos en subrutinas y funciones por valor o referencia.

Todas las subrutinas que tengan argumentos tienen que ser invocadas (llamadas) desde otra macro, si intentas ejecutar directamente una subrutina con argumentos desde el IDE, te dará el siguiente error.



Y si la intentas ejecutar desde la interfaz del usuario, es decir, desde el menú ***Herramientas*** | ***Macros*** | ***Ejecutar macro...***, te dará el siguiente error.



Las subrutinas nos ayudan a dividir nuestro código en bloques lógicos más manejables. Cierta ocasión, no recuerdo dónde, leí que una macro, procedimiento o subrutina, no "debería" tener más de una página de líneas, esto por supuesto es sumamente subjetivo, escribir "código compacto" no es sinónimo de "código eficiente", una vez más, la práctica diaria te dictará los mejores argumentos para programar.

Habrá ocasiones, en que quieras salir anticipadamente de una subrutina, para ello, existe una instrucción que se llama **Exit Sub**, veamos su uso.

```
Option Explicit

Sub Ejemplo_ExitSub()
Dim sFrase As String

   sFrase = Trim(InputBox("Escribe una frase"))
   Call ContarLetras( sFrase )

End Sub

Sub ContarLetras( Cadena As String)

   If Cadena = "" Then
         Exit Sub
   Else
         MsgBox "Hay" & Str(Len(Cadena)) & " letras en la cadena" & Chr(13) & Chr(13) & Cadena
   End If

End Sub
```

Observa cómo en la subrutina ***ContarLetras*** si el argumento pasado está vacío, sale inmediatamente de la macro con la instrucción **Exit Sub**, algunos autores no recomiendan el uso de la instrucción Exit Sub, en "teoría" una subrutina "debería" ser completa, es decir, no "debería" necesitar salidas "forzadas", por supuesto queda a tu criterio y experiencia su uso, por ejemplo, la subrutina anterior podría quedar así.

```
Sub ContarLetras1( Cadena As String)

   If Cadena <> "" then
         MsgBox "Hay" & Str( Len(Cadena) ) & " letras en la cadena" & Chr(13) & Chr(13) & Cadena
   End If
```

```
End Sub
```

Observa cómo sólo ejecutamos el código si el argumento Cadena "no" está vacío, incluso, algunos autores más exigentes, argumentan que la **"validación de datos"** (no se te olvide esta frase), se tiene que hacer "antes" de llamar a una subrutina, es decir, que estas "deberían" hacer sólo su trabajo y pasarle los datos correctos, para ejemplificar esto, observa las modificaciones a las macros anteriores.

```
Sub Ejemplo_ExitSub2()
Dim sFrase As String

   sFrase = Trim(InputBox("Escribe una frase"))
   If sFrase <> "" Then
         Call ContarLetras2( sFrase )
   End If

End Sub

Sub ContarLetras2( Cadena As String)

   MsgBox "Hay" & Str(Len(Cadena)) & " letras en la cadena" & Chr(13) & Chr(13) & Cadena

End Sub
```

Nota que la validación la hacemos con la linea.

```
If sFrase <> "" Then
```

Por lo que al llamar a la macro ***ContarLetras2***, ya le estamos pasando un dato validado, es decir correcto, porque dentro de la subrutina no es necesario hacer ninguna validación, por supuesto esto tiene sus pros y sus contras, sus defensores y sus detractores. Sólo te digo, aplica tu criterio que la experiencia y la práctica serán tus mejores aliadas. Lo que nunca, otra vez lo repito, nunca debes dejar de hacer, es **"validar tus datos"**, garantizar que sean correctos, no importa dónde lo hagas pero valídalos. Un gran porcentaje de errores en muchos programas es por una deficiente validación de datos y también, ya te darás cuenta, que un gran porcentaje de código, se nos va, validando datos.

Veamos otro ejemplo de una subrutina. Supongamos que nos piden hacer una macro que solicite el radio de un círculo y calcule su área.

```
Option Explicit

Sub CalcularAreaCirculo1()
Dim dArea As Double
Dim sRadio As Single
Const PI As Single = 3.1416

   'Solicitamos el radio del círculo a calcular, observa que primero limpiamos los espacios
   'con Trim, después convertimos el valor a tipo Single, que es el tipo de variable esperado
   sRadio = CSng( Trim( InputBox( "¿Cuál es el radio?", "Área de un círculo", "1" ) ) )
   'Sólo si tenemos valores mayores a cero
   If sRadio > 0 Then
         dArea = PI * ( sRadio ^ 2 )
         MsgBox "El área de un círculo de radio = " & Str(sRadio) & " es =: " & Str(dArea)
   End If

End Sub
```

Si creamos una subrutina, podría ser algo así.

```
Option Explicit

Sub CalcularAreaCirculo2()
Dim dArea As Double
Dim sRadio As Single

   sRadio = CSng( Trim( InputBox( "¿Cuál es el radio?", "Área de un círculo", "1" ) ) )
   If sRadio > 0 Then
         Call DevuelveAreaCirculo( sRadio, dArea )
         MsgBox "El área de un círculo de radio " & Str(sRadio) & " es =: " & Str(dArea)
   End If

End Sub

Sub DevuelveAreaCirculo( Radio As Single, Area As Double )
Const PI As Single = 3.1416

   Area = PI * ( Radio ^ 2 )

End Sub
```

Si bien lo anterior funciona, no es común usar subrutinas para manipular variables y argumentos, lo más usual, si se quiere “devolver” un valor, es usar una función en vez de una subrutina, la macro anterior, usando una función, quedaría de la siguiente manera.

```
Option Explicit

Sub CalcularAreaCirculo3()
Dim dArea As Double
Dim sRadio As Single

   sRadio = CSng( Trim( InputBox( "¿Cuál es el radio?", "Área de un círculo", "1" ) ) )
   If sRadio > 0 Then
         'Observa cómo usamos la función y asignamos el resultado a una variable
         dArea = AreaCirculo( sRadio )
         MsgBox "El area de un circulo de radio = " & Str(sRadio) & "  es =: " & Str(dArea)
   End If

End Sub

Function AreaCirculo( Radio As Single ) As Double
Const PI As Single = 3.1416

   'Observa como usamos el nombre de la función para devolver al valor
   AreaCirculo = PI * ( Radio ^ 2 )

End Function
```

Observa el modo de declarar una función, ahora, en vez de usar la palabra clave Sub, usamos una nueva palabra clave **Function**, la declaración de los argumentos tiene las mismas características usadas en las subrutinas, como el paso por valor o referencia, pero un cambio importante, es el tipo de valor “devuelto” por una función, es decir, las funciones también devuelven un “tipo” específico de valor, por supuesto, también pueden devolver el tipo por default de las variables que es Variant pero la recomendación es especificar siempre que te sea posible el tipo de valor que esperas devuelva la función. Otro cambio importante es que para devolver el valor dentro de la función, en vez de usar otro argumento o variable, usamos el mismo nombre de la función como destino y para finalizar, en vez de usar End Sub como hasta ahora, en “todas” las declaraciones de **Function**, debemos cerrarlas con **End Function,** si no, te dará un error.

El paso de argumentos en subrutinas y funciones, tiene otra posibilidad muy interesante, el paso de argumentos opcionales, continuemos con nuestro ejemplo del circulo, supongamos que ahora se nos pide hacer una macro que calcule ya sea el área o el perímetro de un circulo, para usar la misma función, le agregaremos un argumento opcional para saber si lo que queremos calcular es el perímetro.

```
Option Explicit

Sub CalcularCirculo
Dim dArea As Double
Dim dPeri As Double
Dim sRadio As Single

   sRadio = CSng( Trim( InputBox( "¿Cuál es el radio?", "Círculo", "1" ) ) )
   If sRadio > 0 Then
         'Aquí usamos la función SIN el argumento opcional
         dArea = Circulo( sRadio )
         'Y aquí usamos la función CON el argumento opcional
         dPeri = Circulo( sRadio, True )
         MsgBox "Área = " & Str(dArea) & chr(13) & _
               "Perímetro = " & Str(dPeri), 64, "Círculo"
   End If

End Sub


Function Circulo( Radio As Single, Optional Perimetro As Boolean ) As Double
Const PI As Single = 3.1416

   'Comprobamos si el parámetro se paso o no
   If IsMissing( Perimetro ) Then
         'Si no se paso le asignamos el valor por default
         Perimetro = False
   End If

   If Perimetro Then
         Circulo = PI * ( Radio * 2 )
   Else
         Circulo = PI * ( Radio ^ 2 )
   End If

End Function
```

Nota el uso de la palabra clave **Optional**, para fines de lo que se nos pide usamos una variable tipo boleana, verdadera o falsa (Boolean), pero en tus funciones puede declarar al igual que los otros argumentos, del tipo que necesites, puedes declarar tantos argumentos opcionales como necesites, el punto importante es que “compruebes” si se pasó o no el argumento para que en su defecto, le asignes un valor por default a dicho argumento, para verificar si se paso o no un argumento usas la función de OOo Basic **IsMissing(Argumento)**, como se ve en el ejemplo anterior. Todos los parámetros que declares como opcionales deben de ir como últimos argumentos declarados.

De tarea, modifica la macro para que le preguntes al usuario, ¿qué es lo que desea calcular? y por supuesto, solo muestra el valor del cálculo solicitado.

A las funciones también es posible pasarle como argumentos, matrices, lo único que tienes que considerar es que los argumentos esperados como matrices, tienes siempre que declararlos como Variant si para llenar dicha matriz usas la función Array, en otros casos, puedes usar el mismo tipo con que declares tu matriz, veamos un ejemplo de uno y otro.

```
Option Explicit
```

```
Sub Sumando
Dim mDatos(9) As Integer
Dim co1 As Integer

   'Llenamos la matriz con datos aleatorios entre 1 y 100
   For co1 = LBound( mDatos() ) To UBound( mDatos() )
         mDatos( co1 ) = Rnd() * 99 + 1
   Next

   MsgBox "La suma de la matriz es = " & Str( SumaMatriz( mDatos() ) )

End Sub

Function SumaMatriz ( Datos() As Integer ) As Integer
Dim co1 As Integer

   For co1 = LBound( Datos() ) To UBound( Datos() )
         SumaMatriz = SumaMatriz + Datos( co1 )
   Next

End Function
```

Observa cómo se declaró la matriz tipo Integer, tanto la variable en la macro como el argumento en la declaración de la función, ahora, intentemos usar una matriz, haciendo uso de la función Array y veamos qué pasa.

```
Option Explicit

Sub Sumando2()
Dim mDatos() As Integer
Dim iSuma As Integer

   'Llenamos la matriz con la funcion Array
   mDatos() = Array(10,20,30,40,50,60,70,80,90)
   'Intentamos sumar la matriz
   iSuma = SumaMatriz( mDatos() )

   MsgBox Str( iSuma )

End Sub

Function SumaMatriz ( Datos() As Integer ) As Integer
Dim co1 As Integer

   For co1 = LBound( Datos() ) To UBound( Datos() )
         SumaMatriz = SumaMatriz + Datos( co1 )
   Next

End Function
```

Nos da un error, ¿verdad?, la razón es que la función espera una matriz de tipo Integer y se le esta pasando una matriz de tipo Variant, esto es por que la función Array, **siempre**, no importa como hayas declarado la matriz, siempre devuelve una matriz tipo Variant. Corrige la declaración de la función y ahora sí, debe funcionar.

```
Function SumaMatriz ( Datos() As Variant ) As Integer
Dim co1 As Integer

   For co1 = LBound( Datos() ) To UBound( Datos() )
         SumaMatriz = SumaMatriz + Datos( co1 )
   Next
```

```
End Function
```

Observa cómo cambiamos a tipo Variant el tipo de matriz que espera con lo cual, ya no nos da error.

En las funciones, también es posible hacer uso de la instrucción Exit, claro, aquí usaremos **Exit Function**, tiene las mismas consideraciones que te comenté para la instrucción Exit Sub, así que no ahondaré en ello, tan solo te muestro un sencillo ejemplo de su uso.

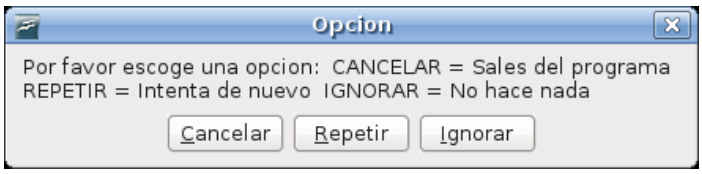
```
Option Explicit

Sub Correo()
Dim sCorreo As String

   sCorreo = Trim(InputBox("Dame tu correo"))
   If ValidarCorreo( sCorreo ) Then
         MsgBox "Correo válido"
   Else
         MsgBox "Correo NO válido"
   End If

End Sub

'Para fines didácticos, sólo validaremos que el correo tenga
'el obligado símbolo de arroba (@) y que no sea ni el primer
'ni el ultimo carácter
Function ValidarCorreo( Correo As String ) As Boolean
Dim pos As Integer

   'Si el argumento Correo esta vacío, salimos de la función
   If Correo = "" Then
         'Si lo deseas esta linea la puedes omitir, pues al salir con Exit Function
         'la función devuelve Falso, pero tal vez en otros casos que no sea booleana
         'la respuesta, necesites asignarle un valor predeterminado diferente
         ValidarCorreo = False
         Exit Function
   Else
         'Buscamos la posición de la arroba con la función InStr
         pos = InStr( 1, Correo, "@" )
         'No debe ser ni el primero ni el ultimo carácter
         'en el siguiente tema aprenderemos más de los operadores lógicos
         If pos > 1 And pos < Len(Correo) Then
               ValidarCorreo = True
         Else
               ValidarCorreo = False
         End If
   End If

End Function
```

Para terminar este tema, veamos el ejemplo de una función que hace uso de una función incorporada del lenguaje que es muy bonita. Supongamos que tenemos que mostrar muchos mensajes al usuario, por ejemplo, el siguiente.

```
Option Explicit

Sub MostrarMensajes1()
Dim sMensaje As String

   sMensaje = "Por favor escoge una opcion: CANCELAR = Sales del programa " & _
              "REPETIR = Intenta de nuevo IGNORAR = No hace nada"
   MsgBox sMensaje, 2, "Opcion"
```

```
End Sub
```

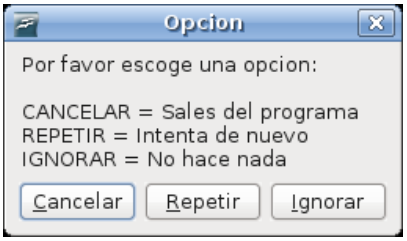
Como podrás notar, la estética no es muy bonita que digamos.



Mejoremos un poco la vista de nuestro cuadro de mensaje, insertando unos saltos de linea, que, como sabes, se hace usando la función Chr pasandole el argumento 10, que es el valor ASCII para el salto de linea, de la siguiente manera.

```
Sub MostrarMensajes2()
Dim sMensaje As String

   sMensaje = "Por favor escoge una opcion:" & Chr(10) & Chr(10) & _
              "CANCELAR = Sales del programa" & Chr(10) & _
              "REPETIR = Intenta de nuevo" & Chr(10) & _
              "IGNORAR = No hace nada"
   MsgBox sMensaje, 2, "Opción"

End Sub
```

Ahora si, nuestro cuadro de mensaje tiene mucha mejor presentación.



Pero imagínate que tienes que mostrar múltiples mensajes con diferentes cadenas y estar concatenando los saltos de linea en cada una, no es una actividad que digamos muy placentera, para ello, hagamos una función que lo haga por nosotros.

```
Function InsertarSaltos( Datos() ) As String

   InsertarSaltos = Join( Datos(), Chr(13) )

End Function
```

Observa que función más sencilla y que útil y linda es, la función **Join** de OOo Basic, es una función que me gusta mucho, espera como argumentos, una matriz y un carácter separador, lo que hace es juntar cada elemento de la matriz separados por el carácter separador, en el ejemplo siguiente, juntamos en una cadena unos números separados por un guion.

```
Sub JuntarDatos()
Dim mDatos() As Variant
Dim sCadena As String

   mDatos() = Array(100,200,300,400,500)
```

```
    sCadena = Join( mDatos(), "-")
    MsgBox sCadena

End Sub
```

La función *InsertarSaltos* la usamos de la siguiente manera.

```
Sub MostrarMensajes3()
Dim mMensajes() As Variant
Dim sMensaje As String

    'Recuerda que la función Array SIEMPRE devuelve una matriz Variant
    mMensajes() = Array("Por favor escoge una opción:","","CANCELAR = Sales del programa","REPETIR =
                Intenta de nuevo","IGNORAR = No hace nada")
    'Llamamos a la función que inserta los saltos de linea
    sMensaje = InsertarSaltos( mMensajes() )
    'Mostramos el mensaje
    MsgBox sMensaje, 2, "Opción"

End Sub
```

Dividir nuestro código en subrutinas y funciones, es el pan nuestro de cada día cuando se desarrollan aplicaciones, no debes de tener duda pues más adelante las usaremos frecuentemente, por ahora, otro tema ha llegado a su fin.

!!Feliz programación¡¡

4.10 Operadores

OOo Basic soporta los siguiente operadores aritméticos, de relación y lógicos.

4.10.1 " ^ " Exponenciación (aritmético)

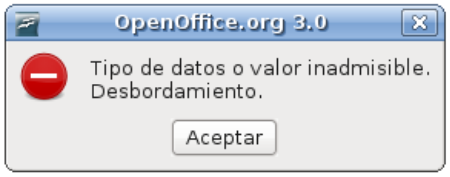
Se usa para elevar un número llamado base a otro llamado exponente. El número resultante del siguiente ejemplo, tal vez te resulte familiar. a^n

```
Option Explicit

Sub Potencias1()
Dim iNum1 As Integer
Dim iNum2 As Integer
Dim iResul As Integer

   iNum1 = 2        'Base
   iNum2 = 10       'Exponente

   iResul = iNum1 ^ iNum2

   MsgBox Str(iResul)

End Sub
```

En general con todos los operadores, debes de tener la precaución de prever lo mejor posible, que el resultado de la operación no exceda el "tamaño" de la variable destino, por ejemplo, la siguiente operación te tiene que dar un error de desbordamiento, es decir, el resultado no cabe en la variable iResul declarada como entera (Integer).

```
Sub Potencias2()
Dim iNum1 As Integer
Dim iNum2 As Integer
Dim iResul As Integer

   iNum1 = 3        'Base
   iNum2 = 10       'Exponente

   iResul = iNum1 ^ iNum2

   MsgBox Str(iResul)

End Sub
```

```
Sub Potencias2()
Dim iNum1 As Integer
Dim iNum2 As Integer
Dim iResul As Integer

        iNum1 = 3       'Base
        iNum2 = 10      'Exponente

        iResul = iNum1 ^ iNum2

        MsgBox Str(iResul)

End Sub
```



La solución es cambiar la variable iResul por una más grande, en este caso Long, pero según tus necesidades podría ser incluso más grande.

```
Dim iResul As Long
```

La base y el exponente no tienen por qué ser enteros, observa el cambio de declaración de las variables.

```
Sub Potencias3()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Double

   iNum1 = 2.53     'Base
   iNum2 = 5.54     'Exponente

   iResul = iNum1 ^ iNum2

   MsgBox Str(iResul)

End Sub
```

Tampoco tienen por qué ser positivos, cuando la base es negativa, el signo de la potencia lo determinará el exponente, si este es par será positiva, si es impar será negativa, de acuerdo a las leyes de los signos que ya conoces desde la escuela elemental, como se comprueba en los siguiente ejemplos.

```
Sub Potencias4()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Double

   iNum1 = -2       'Base
   iNum2 = 4        'Exponente
   iResul = iNum1 ^ iNum2
   MsgBox Str(iResul)     'Resultado positivo

   iNum1 = -2       'Base
   iNum2 = 5        'Exponente
   iResul = iNum1 ^ iNum2
   MsgBox Str(iResul)     'Resultado negativo

End Sub
```

En el caso de que el exponente sea negativo, lo que hace el lenguaje, como lo dictan las matemáticas, invierte la base para que el exponente sea positivo.

```
Sub Potencias5()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Double

   iNum1 = 2        'Base
   iNum2 = -3       'Exponente
   iResul = iNum1 ^ iNum2
   MsgBox Str(iResul)

End Sub
```

$$2^{-3}=\frac{1}{2^3}=\frac{1}{8}=0.125$$

4.10.2 “ * ” Multiplicación (aritmético)

La multiplicación es la operación aritmética que nos permite hacer una suma abreviada, al resultado se le llama producto y los números multiplicados factores.

```
Option Explicit

Sub Multiplicacion1()
Dim iNum1 As Integer
Dim iNum2 As Integer
Dim iResul As Integer

   'Factores
   iNum1 = 365
   iNum2 = 34

   iResul = iNum1 * iNum2

   MsgBox Str(iResul)

End Sub
```

Recuerda que la variable donde asignes el resultado, debe ser lo suficientemente grande para contenerlo, que puedes multiplicar valores enteros y no enteros, positivos y negativos, en este caso, las leyes de los signos aplican correctamente.

```
Option Explicit

Sub Multiplicacion2()
Dim iNum1 As Integer
Dim iNum2 As Integer
Dim iResul As Integer

   iNum1 = 11
   iNum2 = 56
   iResul = iNum1 * iNum2
   MsgBox Str(iResul)

   iNum1 = -11
   iNum2 = 56
   iResul = iNum1 * iNum2
   MsgBox Str(iResul)

   iNum1 = 11
   iNum2 = -56
   iResul = iNum1 * iNum2
```

```
    MsgBox Str(iResul)

    iNum1 = -11
    iNum2 = -56
    iResul = iNum1 * iNum2
    MsgBox Str(iResul)

End Sub
```

Recuerda que en las multiplicaciones el orden de los factores no altera el producto (propiedad conmutativa).

```
Sub Multiplicacion3()
Dim iNum1 As Integer
Dim iNum2 As Integer
Dim iResul As Long

    iNum1 = 12345
    iNum2 = 18
    iResul = iNum1 * iNum2
    MsgBox Str(iResul)

    iNum1 = 18
    iNum2 = 12345
    iResul = iNum1 * iNum2
    MsgBox Str(iResul)

End Sub
```

4.10.3 “ / ” División (aritmético)

Operación aritmética para saber cuantas veces cabe un número en otro, es, podría decirse, una resta abreviada. El número a dividir se llama “dividendo”, al otro “divisor”, al resultado entero se le llama “cociente” y si no es exacta la división a lo que resta se le llama “residuo”. Es la operación inversa a la multiplicación.

Puedes dividir numero enteros y no enteros, negativos y positivos, las leyes de los signos aplican de la misma forma que en la multiplicación.

```
Option Explicit

Sub Division1()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Double

    iNum1 = 123456790           'Dividendo
    iNum2 = 123                 'Divisor
    iResul = iNum1 / iNum2

    MsgBox Str(iResul)

End Sub

Sub Division2()
Dim iNum1 As Integer
Dim iNum2 As Integer
```

```
Dim iResul As Integer

   iNum1 = 589            'Dividendo
   iNum2 = -25            'Divisor
   iResul = iNum1 / iNum2

   MsgBox Str(iResul)

End Sub
```

En el caso de la división, el orden de los operandos si importa, es muy diferente que un número sea dividendo que divisor como podemos verlo en el siguiente ejemplo.

```
Sub Division3()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Double

   iNum1 = 98765
   iNum2 = 321
   iResul = iNum1 / iNum2
   MsgBox Str(iResul)

   iNum1 = 321
   iNum2 = 98765
   iResul = iNum1 / iNum2
   MsgBox Str(iResul)

End Sub
```

4.10.4 “ \ ” División entera (aritmético)

Con este operador obtenemos solo la parte entera de una división.

```
Option Explicit

Sub Division_Entera1()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Single

   iNum1 = 100
   iNum2 = 15

   iResul = iNum1 / iNum2
   MsgBox Str(iResul)

   'Devuelve solo la parte entera
   iResul = iNum1 \ iNum2
   MsgBox Str(iResul)

End Sub
```

En algunos casos, cuando el valor decimal es muy cercano al siguiente entero, este operador te devuelve el siguiente entero, como en.

```
Sub Division_Entera2()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Single

   iNum1 = 100
   iNum2 = 25.1

   iResul = iNum1 / iNum2
   MsgBox Str(iResul)

   iResul = iNum1 \ iNum2
   MsgBox Str(iResul)

End Sub
```

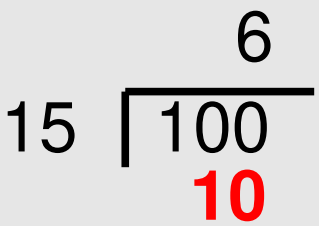
Si quieres asegurarte que “siempre” te regrese “sólo” la parte entera, mejor usa la función **Int**(valor) que te devuelve sólo la parte entera del valor pasado, como te muestro aquí.

```
Sub Division_Entera3()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Single

   iNum1 = 100
   iNum2 = 25.1

   iResul = iNum1 / iNum2
   MsgBox Str(iResul)

   iResul = iNum1 \ iNum2
   MsgBox Str(iResul)

   iResul = Int( iNum1 / iNum2 )
   MsgBox Str(iResul)

End Sub
```

4.10.5 “Mod” Resto de una división entera (aritmético)

Este operador nos devuelve el residuo entero de una división entera.

```
Option Explicit

Sub Resto_Division1()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Single

   iNum1 = 100
   iNum2 = 15

   iResul = iNum1 Mod iNum2
   MsgBox Str(iResul)

End Sub
```



El siguiente ejemplo hace uso de los operadores “Mod” y “\” (división entera) para convertir un número decimal en binario.

```
Sub Decimal_Binario()
Dim iDec As Integer
Dim sBin As String
Dim iBase As Integer

   'Número entero a convertir
   iDec = 100
   'El sistema binario es base 2
   iBase = 2
   'Hasta que el número sea menor que la base
   Do Until iDec < iBase
         'Obtenemos el residuo y lo vamos concatenando
         sBin = (iDec Mod iBase) & sBin
         'Obtenemos la división entera y reasignamos el numero
         'si no haces esto, te quedaras en un ciclo infinito
         iDec = iDec \ iBase
   Loop
   'Por ultimo concatenamos el ultimo cociente
   sBin = iDec & sBin
   'Mostramos el valor en binario
   MsgBox sBin

End Sub
```

4.10.6 “ + ” Suma (aritmético)

La adición, es la operación aritmética que nos permite combinar dos cantidades para obtener una sola. A dichas cantidades se les llama sumandos y al resultado suma.

```
Option Explicit

Sub Sumas1()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Single

   iNum1 = 159.25
   iNum2 = 489.55

   iResul = iNum1 + iNum2

   MsgBox Str( iResul )

End Sub
```

Por su propiedad conmutativa, el orden de los sumandos no altera la suma.

```
Sub Sumas2()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Single

   iNum1 = 1974
   iNum2 = 34
```

```
    iResul = iNum1 + iNum2

    MsgBox Str( iResul )

    iNum1 = 34
    iNum2 = 1974

    iResul = iNum1 + iNum2

    MsgBox Str( iResul )

End Sub
```

Por leyes de los signos, la suma tendrá el signo del mayor sumando.

```
Sub Sumas3()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Single

    iNum1 = 4268
    iNum2 = -258

    iResul = iNum1 + iNum2

    MsgBox Str( iResul )

    iNum1 = -5689
    iNum2 = 197

    iResul = iNum1 + iNum2

    MsgBox Str( iResul )

End Sub
```

4.10.7 “ - ” Resta (aritmético)

La sustracción o resta es la operación aritmética que nos permite conocer la diferencia entre dos números, al primero se le llama minuendo y al segundo sustraendo. Es la operación inversa de la suma.

```
Option Explicit

Sub Restas1()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Single

    iNum1 = 5000          'Minuendo
    iNum2 = 2125          'Sustraendo

    iResul = iNum1 - iNum2

    MsgBox Str( iResul )
```

```
End Sub
```

La resta no tiene propiedad conmutativa, es decir, si intercambias el minuendo y el sustraendo, no te dará el mismo resultado, pero fijate que curioso resultado.

```
Sub Restas2()
Dim iNum1 As Single
Dim iNum2 As Single
Dim iResul As Single

   iNum1 = 562
   iNum2 = 956
   iResul = iNum1 - iNum2
   MsgBox Str( iResul )

   iNum1 = 956
   iNum2 = 562
   iResul = iNum1 - iNum2
   MsgBox Str( iResul )

End Sub
```

4.10.8 Operadores de relación

Estos operadores nos permiten comparar el valor de dos expresiones o términos, siempre devuelven un valor verdadero (True) o falso (False), es decir, un valor booleano.

```
Option Explicit

Sub Operadores_Relacion()
Dim bResul As Boolean
Dim iNum1 As Integer
Dim iNum2 As Integer

   iNum1 = 99
   iNum2 = 19

   bResul = iNum1 = iNum2
   MsgBox "Son iguales? " & bResul

   bResul = iNum1 <> iNum2
   MsgBox "Son distintos? " & bResul

   bResul = iNum1 < iNum2
   MsgBox "Menor que? " & bResul

   bResul = iNum1 > iNum2
   MsgBox "Mayor que? " & bResul

   bResul = iNum1 <= iNum2
   MsgBox "Menor o igual? " & bResul

   bResul = iNum1 >= iNum2
   MsgBox "Mayor o igual? " & bResul

End Sub
```

Ten cuidado de no confundir los signos de igual (=), el primero de izquierda a derecha es el signo igual usado como asignación, el resultado de la expresión a la derecha del signo se asigna a la variable a la izquierda del signo, el segundo signo igual (en negritas y rojo) es usado como operador de relación entre dos términos, este ejemplo es más claro pues esta entre paréntesis.

```
bResul = ( iNum1 = iNum2 )
```

4.10.9 Not – Negación (lógico)

Cambia el valor de la expresión de falso a verdadero y viceversa. Se aplica a solo un operador.

```
Option Explicit

Sub Negacion()
Dim bValor As Boolean

   'Valor original Verdadero
   bValor = True
   Msgbox bValor

   'Le aplicamos el operador
   bValor = Not bValor
   Msgbox bValor

   'Valor original Falso
   bValor = False
   Msgbox bValor

   'Le aplicamos el operador
   bValor = Not bValor
   Msgbox bValor

End Sub
```

Su tabla de verdad es muy sencilla, siendo esta una herramienta para conocer los posibles valores de una expresión cuando se le aplican operadores lógicos.

A	Not A
V	F
F	V

4.10.10 And – Y (lógico)

Se aplica a dos operadores, devuelve verdadero (True) solo en el caso de que los dos operadores sean verdaderos (True), cualquier otro valor, devuelve falso (False). Ahora, primero te muestro su tabla de verdad y después los ejemplos.

A	B	A And B

V	V	V
V	F	F
F	V	F
F	F	F

```
Option Explicit

Sub Conjuncion()
Dim bValor1 As Boolean
Dim bValor2 As Boolean
Dim bResul As Boolean

   'Solo en caso de que los dos sean verdaderos
   'el resultado sera verdadero
   bValor1 = True
   bValor2 = True
   bResul = bValor1 And bValor2
   Msgbox bResul

   bValor1 = True
   bValor2 = False
   bResul = bValor1 And bValor2
   Msgbox bResul

   bValor1 = False
   bValor2 = True
   bResul = bValor1 And bValor2
   Msgbox bResul

   bValor1 = False
   bValor2 = False
   bResul = bValor1 And bValor2
   Msgbox bResul

End Sub
```

4.10.11 Or – O (lógico)

Se aplica a dos operadores. Si los dos operadores son falsos (False), devuelve falso (False), cualquier otro valor devuelve verdadero (True). Su tabla de verdad es.

A	B	A Or B
V	V	V
V	F	V
F	V	V
F	F	F

```
Option Explicit

Sub Disyuncion()
Dim bValor1 As Boolean
```

```
Dim bValor2 As Boolean
Dim bResul As Boolean

   bValor1 = True
   bValor2 = True
   bResul = bValor1 Or bValor2
   Msgbox bResul

   bValor1 = True
   bValor2 = False
   bResul = bValor1 Or bValor2
   Msgbox bResul

   bValor1 = False
   bValor2 = True
   bResul = bValor1 Or bValor2
   Msgbox bResul

   bValor1 = False
   bValor2 = False
   bResul = bValor1 Or bValor2
   Msgbox bResul

End Sub
```

4.10.12 Xor – O exclusiva (lógico)

Se aplica a dos operadores. Si los dos operadores son iguales devuelve falso (False), si son diferentes devuelve verdadero (True). Su tabla de verdad es.

A	B	A Xor B
V	V	F
V	F	V
F	V	V
F	F	F

```
Option Explicit

Sub Operador_Xor()
Dim bValor1 As Boolean
Dim bValor2 As Boolean
Dim bResul As Boolean

   bValor1 = True
   bValor2 = True
   bResul = bValor1 Xor bValor2
   Msgbox bResul

   bValor1 = True
   bValor2 = False
   bResul = bValor1 Xor bValor2
   Msgbox bResul

   bValor1 = False
   bValor2 = True
   bResul = bValor1 Xor bValor2
```

```
    Msgbox bResul

    bValor1 = False
    bValor2 = False
    bResul = bValor1 Xor bValor2
    Msgbox bResul

End Sub
```

4.10.13 Eqv – Equivalencia (opuesto a Xor) (lógico)

Se aplica a dos operadores. Si los dos operadores son iguales devuelve verdadero (True), si son diferentes devuelve falso (False). Su tabla de verdad es.

A	B	A Eqv B
V	V	V
V	F	F
F	V	F
F	F	V

```
Option Explicit

Sub Operador_Eqv()
Dim bValor1 As Boolean
Dim bValor2 As Boolean
Dim bResul As Boolean

   bValor1 = True
   bValor2 = True
   bResul = bValor1 Eqv bValor2
   Msgbox bResul

   bValor1 = True
   bValor2 = False
   bResul = bValor1 Eqv bValor2
   Msgbox bResul

   bValor1 = False
   bValor2 = True
   bResul = bValor1 Eqv bValor2
   Msgbox bResul

   bValor1 = False
   bValor2 = False
   bResul = bValor1 Eqv bValor2
   Msgbox bResul

End Sub
```

4.10.14 Imp – Implicación (lógico)

Se aplica a dos operadores. Si el primer operador es verdadero (True) y el segundo es Falso (False) devuelve falso (False), cualquier otro valor, devuelve verdadero (True). Su tabla de verdad es.

A	B	A Imp B
V	V	V
V	F	F
F	V	V
F	F	V

```
Option Explicit

Sub Operador_Imp()
Dim bValor1 As Boolean
Dim bValor2 As Boolean
Dim bResul As Boolean

   bValor1 = True
   bValor2 = True
   bResul = bValor1 Imp bValor2
   Msgbox bResul

   bValor1 = True
   bValor2 = False
   bResul = bValor1 Imp bValor2
   Msgbox bResul

   bValor1 = False
   bValor2 = True
   bResul = bValor1 Imp bValor2
   Msgbox bResul

   bValor1 = False
   bValor2 = False
   bResul = bValor1 Imp bValor2
   Msgbox bResul

End Sub
```

Los operadores de relación y lógicos vistos en este apartado, son de vital importancia para su implementación en cualquier tipo de condición en ciclos o bifurcaciones. Recuerda que siempre obtendrás un valor verdadero (True) o falso (False). Ten a la mano siempre las tablas de verdad de cada uno.

4.10.15 Precedencia de operadores

En resumen, los operadores que puedes usar en OOo Basic, son los siguientes, están ordenados de mayor a menor precedencia, es decir, la prioridad que les asigna el lenguaje cuando en una sentencia están implicados dos o mas operadores. Los operadores que están en la misma linea, tiene el mismo nivel de precedencia

Operador	Operación	Tipo
^	Exponenciación	Aritmético
* , /	Multiplicación y división	Aritmético

\	División entera	Aritmético
Mod	Resto de una división entera	Aritmético
+ , -	Suma y resta	Aritmético
= , <> , < , > <= , >=	Igual, distinto, menor que, mayor que, menor o igual que, mayor o igual que	Relacionales
Not	Negación	Lógico
And	Y	Lógico
Or	O	Lógico
Xor	O exclusiva	Lógico
Eqv	Equivalencia (opuesto a Xor)	Lógico
Imp	Implicación	Lógico

La prioridad de ejecución es muy importante, pues si no la utilizas correctamente, te puede dar resultados diferentes a los esperados, toma en cuenta las siguiente reglas:

- La precedencia va de mayor a menor, es decir, se ejecutan primero los operadores de mayor jerarquía
- Si los operadores tienen la misma precedencia, se ejecutan de izquierda a derecha
- La precedencia se puede “romper” (cambiar) usando paréntesis, tienen prioridad los más internos y hacia afuera

```
Option Explicit

Sub Precedencia_Operadores1()
Dim num As Double

   'Observa como la prioridad va de izquierda a derecha
   '4 16 1 3
   num = 2 ^ 2 * 4 Mod 3 + 2
   MsgBox num

   'Y ahora va de derecha a izquierda
   '16 32 34
   num = 2 + 2 * 4 ^ 2
   MsgBox num

End Sub
```

Lo más sencillo, es que tú determines el orden de las operaciones, ya sea en sucesivas líneas como en el primer ejemplo o usando paréntesis como en el segundo ejemplo.

```
Sub Precedencia_Operadores2()
Dim num As Double

   num = 45 * 56
   num = num / 10
   num = num ^ 3
   num = num + 125
   MsgBox num

   num = (((45 * 56) / 10) ^ 3) + 125
   MsgBox num

   'Observa como, sin los paréntesis da un resultado totalmente diferente
```

```
    num = 45 * 56 / 10 ^ 3 + 125
    MsgBox num

End Sub
```

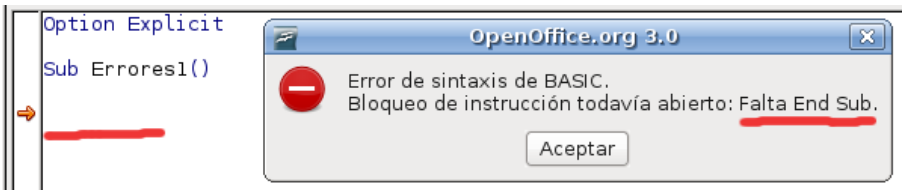
Los paréntesis nos ayudan a determinar el orden en que queramos hacer una operación cuando la precedencia de operaciones no es muy clara. Mi recomendación como casi siempre lo notaras, es que mantengas el control de tu código. Los operadores de relación y lógicos son muy importantes al aplicarse en condiciones para bucles y bifurcaciones, es importante que los domines.

4.11 Control de errores

En teoría, un programa no debería tener errores o no necesitaría de un control de errores, en la práctica, sabemos que esto no es así. En general podríamos hablar de tres tipos de errores dependiendo de cuándo se producen o en qué contexto.

1) **Errores en tiempo de diseño:** son aquellos que se cometen cuando se está codificando, programando, escribiendo nuestro código, generalmente son detectados por el IDE en cuanto tratamos de ejecutar el código y normalmente nos muestra un mensaje indicándonos el tipo de error cometido, muy comúnmente son errores de sintaxis, recordando que sintaxis es: -*Conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje de programación*- por ejemplo, el siguiente código, te deberá mostrar el siguiente mensaje de error que es muy claro:

```
Option Explicit

Sub Errores1()
```



En el apéndice punto 11.2, te muestro una lista de los errores en tiempo de diseño más frecuentes.

2) **Errores lógicos**: estos errores, son los más difíciles de detectar y corregir pues frecuentemente no te dan un mensaje o no se detiene el código, sino simplemente el código “no hace” las tareas para las que se supone esta desarrollado o “no devuelve” los valores esperados, entre más grande y complejo sea tu código, la probabilidad de que tenga errores lógicos aumenta considerablemente. Hay una frase que se le atribuye (no he podido comprobarlo) a Linus Torvalds que dice -ante los ojos de muchos, los errores son evidentes-, por supuesto, esto solo puede suceder con el software libre, como el que tú y yo usamos y desarrollamos, ¿verdad?, pues solo en él tenemos a nuestra disposición el código fuente que es la única forma de verificar que un software

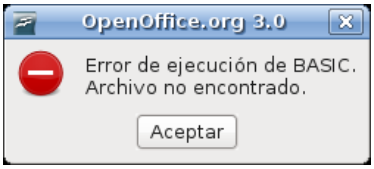
haga lo que dice que hace. Únicamente la experiencia y la práctica te ayudarán a minimizar este tipo de errores.

3) **Errores en tiempo de ejecución**: estos errores se producen durante el tiempo que se está ejecutando tu programa o código. OOo Basic cuenta con instrucciones y palabras claves para controlar este tipo de errores y son los que aprenderemos en este capitulo.

Copia y ejecuta la siguiente macro:

```
Option Explicit

Sub Manejo_Errores1()
Dim sRutaArchivo As String

   sRutaArchivo = Trim(InputBox("Escribe la ruta del archivo a borrar"))

   Kill sRutaArchivo

   MsgBox "El archivo " & sRutaArchivo & " se borro correctamente"

End Sub
```

Si escribes la ruta de un archivo que no existe, te “debe” dar el mensaje de error y nota que el último mensaje con MsgBox ya “no” se muestra.



Claro, estarás pensando que es más fácil y más eficiente el validar antes que exista el archivo y estarás en lo correcto.

```
Sub Manejo_Errores2()
Dim sRutaArchivo As String

   sRutaArchivo = Trim(InputBox("Escribe la ruta del archivo a borrar"))

   If sRutaArchivo<>"" And Dir(sRutaArchivo)<>"" Then
        Kill sRutaArchivo
        MsgBox "El archivo " & sRutaArchivo & " se borro correctamente"
   Else
        MsgBox "Ruta invalida o archivo no existe", 0, "Borrar archivo"
   End If

End Sub
```

En la validación anterior estamos usando una función de OOo Basic que tal vez no conozcas, me refiero a la función **Dir**(valor), donde valor puede ser una ruta de archivo, si “no” encuentra el archivo, devuelve una cadena vacía.

Considera que cuando accedes a recursos externos, no tienes la seguridad que esos recursos estarán “siempre” disponibles, en el caso de los archivos, otros procesos pueden acceder y manipularlos, por ello, en este y otros casos, considera la utilización de un controlador de errores como en el ejemplo siguiente.

```
Option Explicit

Sub Manejo_Errores3()
Dim sRutaArchivo As String

On Error Goto CONTROLAERRORES

   sRutaArchivo = Trim(InputBox("Escribe la ruta del archivo a borrar"))

   Kill sRutaArchivo

   MsgBox "El archivo " & sRutaArchivo & " se borro correctamente"

Exit Sub

CONTROLAERRORES:
   Select Case Err
         Case 0
         Case 53
         MsgBox "No se encontró la ruta especificada" & Chr(13) & Chr(13) & sRutaArchivo, 48
         Case Else
               MsgBox "Ocurrió el error numero: " & Err & Chr(13) & Error & _
               Chr(13) & "En la linea " & Erl
   End Select
   On Error Goto 0
End Sub
```

Veamos todas las partes del código anterior. Después de declarar la macro y las variables a usar, tenemos una nueva instrucción.

```
On Error Goto CONTROLAERRORES
```

La sintaxis general de esta instrucción es:

On Error Goto *NombreEtiqueta*

En donde ***NombreEtiqueta*** es cualquier palabra que cumpla las características vistas para nombrar a las variables y literalmente significa -En caso de error salta (o va) a *NombreEtiqueta*. Después de iniciar el controlador de errores, tenemos propiamente el código de nuestra macro. Enseguida, observa que usamos la instrucción Exit Sub, esto es para que si, nuestro código se ejecuta sin problemas, salga de la macro “sin” ejecutar el controlador de errores. Ahora sí, observa cómo declaramos la etiqueta usada en la declaración del controlador de errores, nota que esta etiqueta, excepto por que es indistinto las mayúsculas y minúsculas, es exactamente igual a la usada en la declaración del controlador de errores, también nota, muy importante, que esta termina en dos puntos (:) , estos, son indispensables para que no te muestre el siguiente mensaje de error en tiempo de diseño.

Inmediatamente después de la declaración de la etiqueta, observa que iniciamos un Select Case con la variable **Err**, esta, es una variable tipo Long de OOo Basic que contiene el número de error que se haya provocado, esta variable tendrá valor 0 si no ocurrió ningún error.

En nuestro ejemplo, sabemos que si no se encuentra un archivo, ocurre el error 53, el cual manipulamos con la linea:

```
Case 53
   MsgBox "No se encontró la ruta especificada" & Chr(13) & Chr(13) & sRutaArchivo, 48
```

En caso de que ocurra un error no especificado o desconocido, mostramos el número de error (variable **Err**), la descripción del error (variable **Error**) y la línea donde este ocurrió (variable **Erl**), cuya valiosa información nos ayudará a corregirlo.

```
Case Else
        MsgBox "Ocurrió el error numero: " & Err & Chr(13) & Error & _
              Chr(13) & "En la linea " & Erl
```

Después de cerrar la estructura Select Case con End Select, tenemos la linea:

```
On Error Goto 0
```

Que no hace otra cosa que reinicializar las variables de error, es decir **Err**, **Error** y **Erl**. La estructura de control o administración de errores que acabamos de ver, afortunadamente, no es la única pero es la que generalmente se usa, hay algunas variantes que a criterio puedes usar, veamos algunas. En caso de que quieras establecer un control de errores genérico, podrías usar el siguiente.

```
Sub Manejo_Errores4()
On Error Goto CONTROLAERRORES

   'Aquí va todo tú código

Exit Sub
CONTROLAERRORES:
   If Err <> 0 Then
         MsgBox "Ocurrió el error numero: " & Err & Chr(13) & Error & Chr(13) & "En la linea " & Erl
   End If
   On Error Goto 0
End Sub
```

En vez de mostrar el error con MsgBox, puedes optar por guardar en un archivo de registro estos errores, aunque siempre es bueno mostrarle al usuario que algo salió mal para poder corregirlo.

Puedes simular el error que quieras simplemente asignando el número de error a la variable Error de la siguiente manera.

```
   'Aqui va todo tú código
   Error(12)
```

Esta es la lista de valores de errores que puedes usar en la variable Error o que en algún momento te pueden aparecer, toma en cuenta que algunos de estos errores “solo” ocurren en tiempo de diseño y varios de ellos “**no**” puedes interceptarlos, tan solo corregirlos en tu código.

Códigos de error			
Nº	*Descripción*	*Nº*	*Descripción*
2	Error de sintaxis no especificado	70	Permiso denegado
3	Return sin Gosub	71	Disco no preparado

Códigos de error			
Nº	Descripción	Nº	Descripción
4	Restaurar desde el principio	73	Prestación no implementada
5	Llamada a procedimiento no válida	74	Imposible cambiar nombre con unidad distinta
6	Desbordamiento	75	Error de acceso a ruta/archivo
7	Memoria agotada	76	Ruta no encontrada
8	Matriz ya dimensionada	91	Variable de objeto no definida
9	Sunbíndice fuera de rango	93	Cadena de secuencia no válida
10	Definición duplicada	94	Uso de Null no válido
11	División por cero	323	Imposible cargar módulo
12	Variable no definida	341	Índice de objeto no válido
13	Discordancia de tipo	366	No hay documento o vista activos
14	Parámetro no válido	380	Valor de propiedad incorrecto
18	Interrupción de usuario	382	Propiedad de sólo lectura
20	Continuar sin error	394	Propiedad de sólo escritura
28	Espacio de pila agotado	420	Referencia de objeto no válida
35	Sub o Function no definidos	423	Propiedad o método no encontrados
48	Error al cargar DLL	424	Objeto necesario
49	Convención de llamada a DLL incorrecta	425	Uso de objeto no válido
51	Error interno	430	La clase no admite OLE
52	Nombre de archivo o número incorrectos	438	El objeto no admite este método
53	Archivo no encontrado	440	Error de automatización OLE
54	Modo de archivo incorrecto	445	El objeto no admite esta acción
55	Archivo ya abierto	446	El objeto no admite argumentos con nombre
57	Error de E/S de dispositivo	447	El objeto no admite la configuración de entorno local actual
58	Archivo ya existente	448	Argumento mencionado no encontrado
59	Longitud de registro incorrecta	449	Argumento no opcional
61	Disco lleno	450	Número de argumentos incorrecto
62	Entrada más allá del final del archivo	451	Objeto no es una colección
63	Número de registro incorrecto	452	Ordinal no válido
67	Demasiados archivos	453	Función DLL especificada no encontrada
68	Dispositivo no disponible	460	Formato de portapapeles no válido

OOo Basic cuenta con una instrucción que complementa el uso de un controlador de errores, esta instrucción es **Resume**, básicamente tiene tres posibles usos, de la primer forma, nos permite regresar, es decir, volver a intentar ejecutar la línea que provocó el error, por supuesto, esto es recomendable ya que se hayan corregido o atendido las causas del error, por ejemplo.

```
Option Explicit
```

```
Sub Manejo_Errores5()
Dim sRutaArchivo As String
Dim iRes As Integer

On Error Goto CONTROLAERRORES

   sRutaArchivo = Trim(InputBox("Escribe la ruta del archivo a borrar"))

   Kill sRutaArchivo

   MsgBox "El archivo " & sRutaArchivo & " se borró correctamente"

Exit Sub

CONTROLAERRORES:
   Select Case Err
         Case 0
         Case 53
               iRes = MsgBox ("No se encontró la ruta especificada" & Chr(13) & Chr(13) &
sRutaArchivo & Chr(13) & Chr(13) & "¿Deseas intentarlo de nuevo?", 32 + 4 )
               If iRes = 6 Then
                     Resume
               End If
         Case Else
               MsgBox "Ocurrió el error numero: " & Err & Chr(13) & Error & _
                     Chr(13) & "En la linea " & Erl
   End Select
   On Error Goto 0
End Sub
```

Observa cómo dentro del bloque donde controlamos el error, preguntamos al usuario si desea volver a intentarlo.

```
Case 53
   iRes = MsgBox ("No se encontró la ruta especificada" & Chr(13) & Chr(13) & sRutaArchivo &
Chr(13) & Chr(13) & "¿Deseas intentarlo de nuevo?", 32 + 4 )
   If iRes = 6 Then
         Resume
   End If
```

Si no haces lo anterior, puedes quedar en un bucle infinito entre la linea que provoco el error y el controlador de errores.

La segunda forma, usando **Resume Next**, te permite continuar la ejecución de tu código en la línea siguiente a la que provocó el error, por ejemplo.

```
Sub Manejo_Errores6()
Dim sRutaArchivo As String
Dim iRes As Integer

On Error Goto CONTROLAERRORES

   sRutaArchivo = Trim(InputBox("Escribe la ruta del archivo a borrar"))
   Kill sRutaArchivo
   MsgBox "El archivo " & sRutaArchivo & " se borro correctamente"

Exit Sub

CONTROLAERRORES:
   Select Case Err
         Case 0
         Case 53
```

```
            iRes = MsgBox ("No se encontró la ruta especificada" & Chr(13) & Chr(13) &
sRutaArchivo & Chr(13) & Chr(13) & "¿Deseas continuar?", 32 + 4 )
            If iRes = 6 Then
                  Resume Next
            End If
      Case Else
            MsgBox "Ocurrió el error numero: " & Err & Chr(13) & Error & _
                        Chr(13) & "En la linea " & Erl
   End Select
   On Error Goto 0
End Sub
```

Aquí lo importante es que notes cómo nos muestra el mensaje de confirmación de borrado, aún y cuando este no se hizo efectivamente, tu tarea es corregir eso.

La ultima forma es usar la instrucción **Resumen** ***NombreEtiqueta***, en donde ***NombreEtiqueta*** tiene las mismas consideraciones vistas al inicio de este tema, ejemplo.

```
Sub Manejo_Errores7()
Dim sRutaArchivo As String
Dim iRes As Integer

On Error Goto CONTROLAERRORES

REINTENTARLO:
   sRutaArchivo = Trim(InputBox("Escribe la ruta del archivo a borrar"))
   Kill sRutaArchivo
   MsgBox "El archivo " & sRutaArchivo & " se borro correctamente"

Exit Sub

CONTROLAERRORES:
   Select Case Err
      Case 0
      Case 53
            iRes = MsgBox ("No se encontró la ruta especificada" & Chr(13) & Chr(13) &
sRutaArchivo & Chr(13) & Chr(13) & "¿Deseas intentarlo de nuevo?", 32 + 4 )
            If iRes = 6 Then
                  Resume REINTENTARLO
            End If
      Case Else
            MsgBox "Ocurrió el error numero: " & Err & Chr(13) & Error & _
                  Chr(13) & "En la linea " & Erl
   End Select
   On Error Goto 0
End Sub
```

No dejes de observar la declaración de la etiqueta correspondiente a donde va a saltar la instrucción Resume, aunque el uso más común es redirigir a un segundo bloque donde se sale de la macro de forma controlada, por ejemplo.

```
Sub Manejo_Errores8()
Dim sRutaArchivo As String

On Error Goto CONTROLAERRORES

   sRutaArchivo = Trim(InputBox("Escribe la ruta del archivo a borrar"))
   Kill sRutaArchivo
   MsgBox "El archivo " & sRutaArchivo & " se borró correctamente"

Goto SALIDACONTROLADA
```

```
CONTROLAERRORES:
   Select Case Err
         Case 0
         Case 53
               MsgBox "No se encontró la ruta especificada" & Chr(13) & Chr(13) & sRutaArchivo &
Chr(13) & Chr(13) & "El programa terminará", 48
               Resume SALIDACONTROLADA
         Case Else
               MsgBox "Ocurrió el error numero: " & Err & Chr(13) & Error & _
                     Chr(13) & "En la linea " & Erl
   End Select
   On Error Goto 0

SALIDACONTROLADA:
   'Aquí va todo el código que quieras, por ejemplo
   'cierre de archivos y bases de datos, la idea
   'es salir de forma controlada de tu macro
   MsgBox "Se saldrá correctamente"

End Sub
```

Nota también que en vez de usar Exit Sub, “antes” de la etiqueta y código de controlador de errores, usamos la instrucción **Goto *NombreEtiqueta***, para saltar directamente a la etiqueta de salida. Estas opciones son solo variantes, una vez más tú decidirás que es lo que mejor se acomoda a tu forma de trabajar y de razonar, la recomendación es que no abuses de los saltos con Resume o Goto, por claridad, limita su uso lo más posible.

Para terminar este tema, te muestro otra instrucción para “omitir” errores, me refiero a la instrucción **On Error Resume Next**, que significa -En caso de error, continua el la siguiente línea-, y se usa de la siguiente manera:

```
Sub Manejo_Errores9()
Dim sRutaArchivo As String

On Error Resume Next

   sRutaArchivo = Trim(InputBox("Escribe la ruta del archivo a borrar"))
   Kill sRutaArchivo
   MsgBox "El archivo " & sRutaArchivo & " se borro correctamente"

End Sub
```

Intentemos controlar el error.

```
Sub Manejo_Errores10()
Dim sRutaArchivo As String

On Error Resume Next
   sRutaArchivo = Trim(InputBox("Escribe la ruta del archivo a borrar"))
   Kill sRutaArchivo
   If Err = 0 Then
         MsgBox "El archivo " & sRutaArchivo & " se borro correctamente"
   End If

End Sub
```

Notaras que no es posible, no importa el número de errores que sucedan.

```
Sub Manejo_Errores11()
Dim sRutaArchivo As String
```

```
On Error Resume Next

   Error(2)
   Error(3)
   Error(4)
   Error(5)
   Error(6)

   If Err = 0 Then
        MsgBox "No ocurrió ningún error"
   End If

End Sub
```

Puedes tener juntas una después de otra las dos instrucciones para controlar errores.

```
On Error Resume Next
On Error Goto CONTROLAERRORES
```

Pero tendrá prioridad On Error Resume Next, por lo que no es aconsejable, de hecho, no te recomiendo el uso de On Error Resume Next a menos que estés seguro de lo que haces.

4.12 Validación de datos

Con la experiencia te darás cuenta que hacer una macro o programa libre de errores no es tema menor y que entre más grande sea tu código, la probabilidad de que tenga errores aumenta considerablemente. Una fuente frecuente de errores, son los datos que un usuario alimenta a un programa, por supuesto, es nuestra responsabilidad como programadores el minimizar este tipo de errores, esto lo logramos filtrando los datos que el usuario alimenta, es decir "validando" los datos, no se te olvide, **VALIDANDO LOS DATOS**, veras que la validación de datos, consume bastante líneas de código, a veces, muchas más que el proceso en si de estos, por ello es un tema muy importante en programación. OOo Basic cuenta con algunas funciones que nos ayudan a validar datos, otras, tienes que programartelas (e ingeniertales) tu mismo, veamos algunas de ellas, pero recuerda que son solo guías, pues las opciones son tantas y tan variadas como programadores las implementen. Para el mejor aprovechamiento de este tema, tienes que saber que los lenguajes Basic son muy nobles en al conversión de datos de un tipo a otro, muchas de estas conversiones las puede hacer el lenguaje de forma directa, a esta conversión se se llama **implícita**, pero lo recomendable es que nosotros controlemos, siempre que sea posible, estas conversiones, a esta forma se le llama **explicita** y nos apoyamos en varias funciones de OOo Basic para hacer estas conversiones.

Como sabes, la instrucción MsgBox, espera como parámetro "requerido" un argumento llamado Mensaje de tipo String, es decir, una cadena de texto.

```
MsgBox Mensaje As String, [Tipo As Integer], [Titulo As String]
```

Veamos los siguientes ejemplos.

```
Option Explicit
```

```
Sub Mensajes1()
Dim dFecha As Date
Dim iNumero As Integer
dim bOpcion As Boolean

   dFecha = DateSerial(74,1,15)
   iNumero = 2008
   bOpcion = True

   MsgBox dFecha
   MsgBox iNumero
   MsgBox bOpcion

End Sub
```

Observa como, aunque le pasemos un número o una fecha o incluso un valor booleano, la instrucción MsgBox sigue trabajando, esto es por que el lenguaje, hace internamente (de forma **implícita**) una conversión de tipos, mientras le sea posible, convierte del tipo que reciba, a una cadena de texto (string), lo ideal, es que nosotros hagamos esta conversión de forma explicita, como en el siguiente ejemplo.

```
Sub Mensajes2()
Dim dFecha As Date
Dim iNumero As Integer
dim bOpcion As Boolean

   dFecha = DateSerial(74,1,15)
   iNumero = 2008
   bOpcion = True

   MsgBox CStr( dFecha )
   MsgBox CStr( iNumero )
   MsgBox CStr( bOpcion )

End Sub
```

Observa como hacemos uso de la función **CStr**(valor) que convierte el tipo de "valor" a una cadena de texto (String). De esta forma, "casi" podemos estar seguros de que no nos dará un error inesperado. OOo Basic cuenta con funciones para convertir a la mayoría de los tipos de variables posibles.

Función	*Descripción*
CStr(valor)	Convierte a cadena de texto (String)
CByte(valor)	Convierte a tipo Byte
CInt(valor)	Convierte a tipo entero (Integer)
CLng(valor)	Convierte a tipo entero largo (Long)
CSng(valor)	Convierte a tipo simple (Single)
CDbl(valor)	Convierte a tipo doble (Double)
CBool(valor)	Convierte a tipo booleano (Boolean)
CDate(valor)	Convierte a tipo fecha (Date)

OOo Basic cuenta con otras funciones que nos ayudan a saber que tipo de dato tiene una variable, veamos las siguientes.

Cuando necesites saber si una variable contiene un número usamos **IsNumeric.**

```
Option Explicit

Sub EsNumero()
Dim sDato As String

   sDato = Trim( InputBox( "Introduce un numero" ) )
   If IsNumeric( sDato ) Then
         MsgBox "Es un numero"
   Else
         MsgBox "No es un numero"
   End If

End Sub
```

Cuando necesites saber si una variable contiene una fecha usamos **IsDate.**

```
Sub EsFecha()
Dim sDato As String

   sDato = Trim( InputBox( "Introduce una fecha" ) )
   If IsDate( sDato ) Then
         MsgBox "Es una fecha"
   Else
         MsgBox "No es una fecha"
   End If

End Sub
```

Y cuando necesites saber si una variable es una matriz usamos **IsArray.**

```
Sub EsMatriz()
Dim mDato As Variant

   If IsArray( mDato() ) Then
         MsgBox "Es una matriz"
   Else
         MsgBox "No es una matriz"
   End If

   mDato = Array(1,2,3,4,5)

   If IsArray( mDato() ) Then
         MsgBox "Es una matriz"
   Else
         MsgBox "No es una matriz"
   End If

End Sub
```

Veamos algunos ejemplos de validaciones comunes, como dicen en mi pueblo -ni son todas las que están, ni están todas las que son-, solo pretendo que sean de referencia y las adaptes y complementes a tus necesidades.

Obligamos al usuario a introducir un dato, el que sea, la única condición es que no sea una cadena vacía.

```
Sub Validar1()
Dim sDato As String
```

```
Dim bSalir As Boolean

   Do
         sDato = Trim( InputBox("Introduce los datos") )
         If sDato <> "" Then
               bSalir = True
         End If
   Loop Until bSalir

End Sub
```

Habrás observado en el ejemplo anterior, que aun cuando el usuario presione Cancelar, de todos modos no le hacemos caso y lo obligamos a introducir un dato, si bien lo podemos hacer, no es común, lo habitual es proporcionar siempre al usuario una forma de cancelar un proceso.

```
Sub Validar2()
Dim sDato As String
Dim bSalir As Boolean
Dim iRes As Integer

   Do
         sDato = Trim( InputBox("Introduce los datos") )
         If sDato <> "" Then
               'Aquí va tu código para manipular los datos
               bSalir = True
         Else
               iRes = MsgBox( "Parece que no escribiste nada, ¿Deseas salir?", 32 + 4, "Salir" )
               If iRes = 6 Then
                     bSalir = True
               End If
         End If
   Loop Until bSalir

End Sub
```

Ahora, obligamos al usuario a introducir una cadena de texto, que “no” sea número ni fecha.

```
Sub Validar3()
Dim sDato As String
Dim bSalir As Boolean
Dim iRes As Integer

   Do
         sDato = Trim( InputBox("Introduce los datos") )
         If sDato <> "" Then
               If Not(IsDate(sDato)) And Not(IsNumeric(sDato)) Then
                     'Aquí va tu código para manipular los datos
                     MsgBox "Es una cadena" & Chr(13) & Chr(13) & sDato
                     bSalir = True
               Else
                     MsgBox "Valor NO valido"
               End If
         Else
               iRes = MsgBox( "Parece que no escribiste nada, ¿Deseas salir?", 32 + 4, "Salir" )
               If iRes = 6 Then
                     bSalir = True
               End If
         End If
   Loop Until bSalir
```

```
End Sub
```

Que solo introduzca números.

```
Sub Validar4()
Dim sDato As String
Dim bSalir As Boolean
Dim iRes As Integer

   Do
         sDato = Trim( InputBox("Introduce un numero") )
         If sDato <> "" Then
               If IsNumeric(sDato) Then
                     'Aquí va tu código para manipular los datos
                     MsgBox "Es un numero" & Chr(13) & Chr(13) & sDato
                     bSalir = True
               Else
                     MsgBox "Valor NO valido"
               End If
         Else
               iRes = MsgBox( "Parece que no escribiste nada, ¿Deseas salir?", 32 + 4, "Salir" )
               If iRes = 6 Then
                     bSalir = True
               End If
         End If
   Loop Until bSalir

End Sub
```

Que introduzca una fecha.

```
Sub Validar5()
Dim sDato As String
Dim bSalir As Boolean
Dim iRes As Integer

   Do
         sDato = Trim( InputBox("Introduce una fecha") )
         If sDato <> "" Then
               If IsDate(sDato) Then
                     'Aquí va tu código para manipular los datos
                     MsgBox "Es una fecha" & Chr(13) & Chr(13) & sDato
                     bSalir = True
               Else
                     MsgBox "Valor NO valido"
               End If
         Else
               iRes = MsgBox( "Parece que no capturaste nada, ¿Deseas salir?", 32 + 4, "Salir" )
               If iRes = 6 Then
                     bSalir = True
               End If
         End If
   Loop Until bSalir

End Sub
```

Un número, con un rango definido, digamos, entre 50 y 100.

```
Sub Validar6()
Dim sDato As String
Dim bSalir As Boolean
```

```
Dim iRes As Integer

   Do
         sDato = Trim( InputBox("Introduce un numero") )
         If sDato <> "" Then
               If IsNumeric(sDato) And Val(sDato)>=50 And Val(sDato)<=100 Then
                     'Aquí va tu código para manipular los datos
                     MsgBox "Es un numero, en el rango 50-100" & Chr(13) & Chr(13) & sDato
                     bSalir = True
               Else
                     MsgBox "Valor NO valido"
               End If
         Else
               iRes = MsgBox( "Parece que no capturaste nada, ¿Deseas salir?", 32 + 4, "Salir" )
               If iRes = 6 Then
                     bSalir = True
               End If
         End If
   Loop Until bSalir

End Sub
```

Una fecha que no sea fecha futura.

```
Sub Validar7()
Dim sDato As String
Dim bSalir As Boolean
Dim iRes As Integer

   Do
         sDato = Trim( InputBox("Introduce una fecha") )
         If sDato <> "" Then
               If IsDate(sDato) Then
                     If CDate(sDato) <= Date() Then
                           'Aquí va tu código para manipular los datos
                           MsgBox "Es una fecha valida" & Chr(13) & Chr(13) & sDato
                           bSalir = True
                     Else
                           MsgBox "Es una fecha futura"
                     End If
               Else
                     MsgBox "Valor NO valido"
               End If
         Else
               iRes = MsgBox( "Parece que no capturaste nada, ¿Deseas salir?", 32 + 4, "Salir" )
               If iRes = 6 Then
                     bSalir = True
               End If
   End If
Loop Until bSalir

End Sub
```

El siguiente ejemplo es muy divertido, le pedimos al usuario que introduzca solo vocales, pero ya sabes como son los usuarios, seguro capturan lo que se les da la gana, así que filtraremos su texto, dejando solo las vocales.

```
Sub Validar8()
Dim sDato As String
Dim bSalir As Boolean
Dim iRes As Integer
Dim co1 As Integer
```

```
Dim sLetra As String
Dim sTmp As String
Dim pos As Integer

   Do
         sDato = Trim( InputBox("Introduce solo vocales") )
         If sDato <> "" Then
               If Not(IsDate(sDato)) And Not(IsNumeric(sDato)) Then
                     'Aquí va tu código para manipular los datos
                     For co1 = 1 To Len(sDato)
                           'Tomamos una por una cada letra capturada
                           sLetra = Mid(sDato,co1,1)
                           'Averiguamos si es una vocal mayúscula o minúscula
                           pos = InStr(1,"aeiouAEIOU",sLetra)
                           'Si encuentra la letra en la lista de vocales InStr te
                           'devuelve la posición, con que sea mayor a cero sabemos
                           'que ha sido encontrada
                           If pos > 0 Then
                                 sTmp = sTmp & sLetra
                           End If
                     Next
                     MsgBox sTmp
                     bSalir = True
               Else
                     MsgBox "Valor no valido"
               End If
         Else
               iRes = MsgBox( "Parece que no capturaste nada, ¿Deseas salir?", 32 + 4, "Salir" )
               If iRes = 6 Then
                     bSalir = True
               End If
         End If
   Loop Until bSalir

End Sub
```

En esta macro tenemos dos nuevas instrucciones de OOo Basic, Mid, que nos sirve para extraer una cadena de otra e InStr que encuentra la posición de una cadena en otra.

No quiero que se te olvide, la **VALIDACION** de los datos que introduce el usuario es “muy importante” y si conjugas esto con el control de errores, te aseguro que podrás minimizar el riesgo de fallos en tus aplicaciones, sino un 100%, si en un porcentaje bastante alto.

Si tienes una validación interesante y que este al nivel básico que venimos manejando, puedes enviármela para incluirla aquí o en su defecto, si tienes problemas con alguna validación, planteala en las listas de correo o foros y si es lo suficientemente ilustrativa, también la incluiremos aquí.

!!Feliz programación¡¡

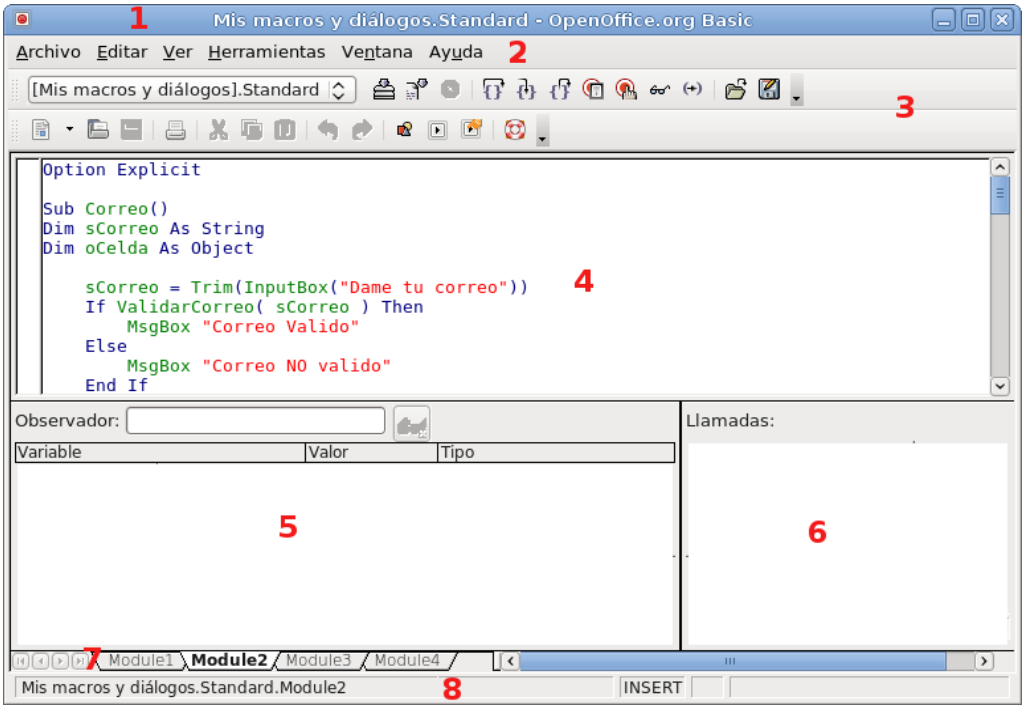
4.13 El IDE – Mucho más que un editor

A estas alturas ya habrás notado la gran utilidad del IDE, te colorea las palabras dependiendo de su contexto y te muestra mensajes preventivos o de error siempre que sea posible. En este capitulo, veremos algunas otras características del IDE que nos pueden ayudar a

depurar nuestras macros, entendiendo por depurar, las técnicas que nos ayudan a encontrar más fácilmente un error, sobre todo lógico, que son, a veces, los más complicados de encontrar.

Para empezar demos un repaso a nuestro centro de trabajo, es decir, al IDE y sus diferentes partes:

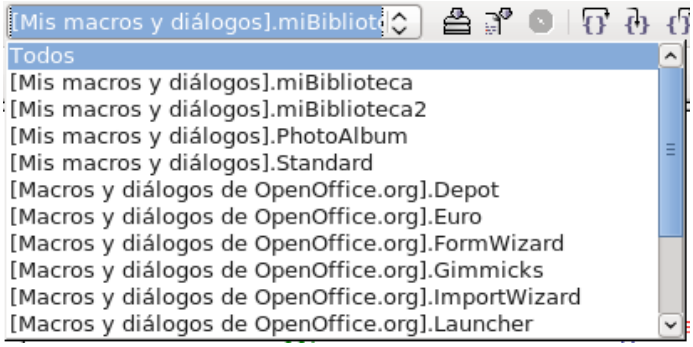
1. Barra de titulo
2. Barra de menús
3. Barras de herramientas
4. Ventana de edición
5. Ventana de observador
6. Ventana de llamadas
7. Barra de desplazamiento de módulos y diálogos
8. Barra de estado



Algunas de estas ya te serán muy familiares, otras son nuevas pero las veremos en este capitulo. Como ya mencionamos, el IDE, es solo otra aplicación de OpenOffice.org, que nos permite crear y editar nuestras macros, la barra de titulo y la de menús es consistente con las demás aplicaciones que integran la aplicación, por lo que no las veremos aquí, Las barras de herramientas las veremos a continuación.

La primer barra de herramientas se llama MacroBar, por lo que supongo que en castellano será, Barra de macros, ya que el nombre no aparece en su barra de título, como sí lo hace en la mayoría de las barras de herramientas.

1. El primer control que es un cuadro de lista desplegable (control ComboBox) te permite seleccionar el archivo que esté abierto y la biblioteca que quieras editar.



Observa como la primer opción es Todos, si seleccionas esta opción, en la barra de desplazamiento de módulos, te mostrara “todos” los módulos y diálogos de todos los archivos abiertos, así como del archivo especial Mis macros y los integrados en OpenOffice.org, por lo que esta opción te podría llegar a mostrar varias decenas de módulos dependiendo de cuantos archivos tengas abiertos y de cuantos módulos tengas con código.

El resto de los botones son los siguientes.

1. Icono de compilar: Me gustaría mucho que el código de nuestras macros se pudiese compilar, que es lo que dice hace este icono, pero todo el código de nuestras macros es interpretado, por lo que este icono, lo único que hace es volver a recorrer el código del modulo actual en busca de errores, pero no compila código.

2. Icono de ejecutar: Este icono ejecuta “solo” la primer macro del módulo activo, pero ya vimos las técnicas para ejecutar las demás macros, equivale a presionar la tecla F5. Recuerda que en algunas distribuciones de OpenOffice.org, puedes ejecutar la macro donde se encuentre el cursor, pero en la oficial aun no.

3. Icono de detener macro: Solo esta activo cuando ejecutamos una macro, normalmente se usa en conjunción con los iconos 4 y 5.

4. Icono pasar al siguiente: Permite ejecutar línea a línea el código de nuestra macro, muy útil para depurar y encontrar errores, normalmente se usa junto con el “Observador” para encontrar errores, equivale a presionar la tecla F8

5. Icono entrar en el proceso: Realiza la misma acción del icono 4, los iconos 4 y 5, solo se ejecutan en la primer macro del modulo activo, por lo que para depurar macros, es mejor usar “puntos de ruptura”, te das cuenta qué línea es la siguiente a ejecutar por una flecha amarilla en el margen izquierdo del editor.

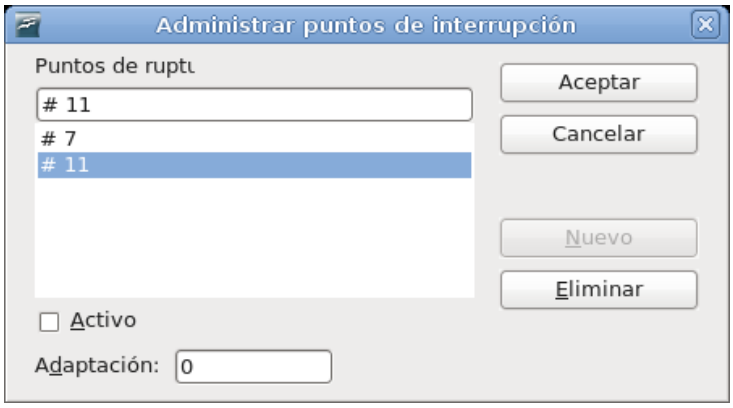
```
Option Explicit

Sub EsNumero()
Dim sDato As String

    sDato = Trim( InputBox( "Introduce un numero" ) )
    If IsNumeric( sDato ) Then
        MsgBox "Es un numero"
    Else
        MsgBox "No es un numero"
    End If

End Sub
```

6. Icono salir de proceso: Cuando ejecutamos una macro línea por línea, este icono nos permite continuar la ejecución de la macro normalmente, equivale a presionar la tecla F5.

7. Icono de activar/desactivar puntos de ruptura: Un punto de ruptura es una "señal" que le indica al IDE que detenga la ejecución en la línea indicada por un punto rojo en el margen izquierdo de nuestro editor, con este icono activamos (mostramos) y desactivamos (ocultamos) los puntos rojos que son puntos de ruptura en la línea actual donde esté el cursor de escritura, recuerda que el cursor de escritura es la raya vertical negra de todo editor de texto, también puedes activar y desactivar puntos de ruptura con la tecla F9 o dando un doble clic en la línea que quieras, pero muy importante, dentro del margen izquierdo del editor. Observa en la siguiente imagen los puntos rojos.

```
Option Explicit

Sub Correo()
Dim sCorreo As String
Dim oCelda As Object

    sCorreo = Trim(InputBox("Dame tu correo"))
    If ValidarCorreo( sCorreo ) Then
        MsgBox "Correo Valido"
    Else
        MsgBox "Correo NO valido"
    End If
End Sub
```

8. Icono gestionar puntos de ruptura: Nos permite administrar los puntos de ruptura existentes en nuestro modulo actual, es decir, nos permite agregar, eliminar o desactivar puntos de ruptura en un cuadro de dialogo. Cada punto de ruptura estará indicado por la línea en donde se encuentre y podrás seleccionar y desactivar el punto que quieras, con lo cual el punto rojo pasará a ser un punto gris, también puede eliminar el punto de interrupción que quieras tan solo seleccionándolo y presionando el botón Eliminar, así mismo, puedes agregar el que quieras, tan solo indicando el numero de línea donde deseas establecerlos y presionar el botón nuevo, pero lo mejor es agregar los puntos desde el editor, exactamente en la línea que quieras.



9. Icono de habilitar inspección: Permite agregar la variable seleccionada a la ventana de observador para visualizar el valor de dicha variable durante la ejecución de una macro, esta ventana normalmente se usa ejecutando la macro línea a línea (F8) o junto con puntos de ruptura para poder observar el valor de las variables, más adelante veremos a detalle esta herramienta

10. Icono de buscar paréntesis: Con el cursor de escritura posicionado “antes” de un paréntesis de apertura o cierre, nos permite seleccionar todo el texto dentro de dichos paréntesis e incluyendo los paréntesis, muy útil sobre todo en grandes funciones anidadas donde se pierden los paréntesis.

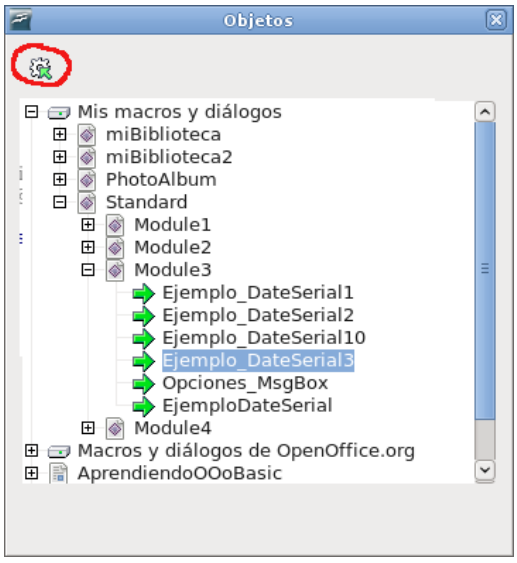
11. Icono de insertar código Basic: Permite, a partir de un archivo de texto (normalmente con extensión BAS) insertar el código Basic que contengo en el modulo actual en la posición del cursor de escritura actual.

12. Icono de guardar código Basic: Nos permite exportar a un archivo de texto con extensión BAS el código seleccionado o todo el modulo si no hay nada seleccionado.

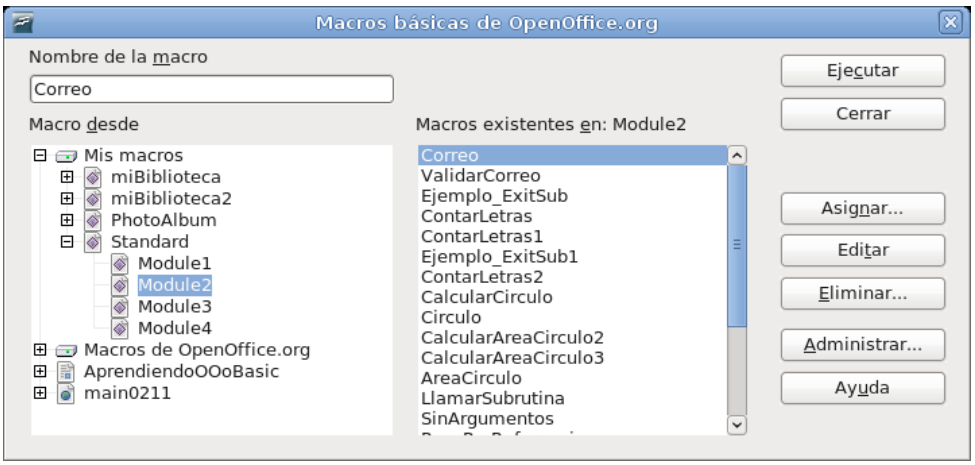
13. Icono de importación de Dialogo: Nos permite importar un cuadro de diálogo.

La siguiente barra de herramientas se llama Estándar, y es muy similar a la mayoría de las barras estándar de las demás aplicaciones, te permite abrir, guardar, imprimir, copiar, pegar, etc, por lo que no veremos aquí estas opciones, salvo por los tres iconos siguientes solo disponibles en el IDE.

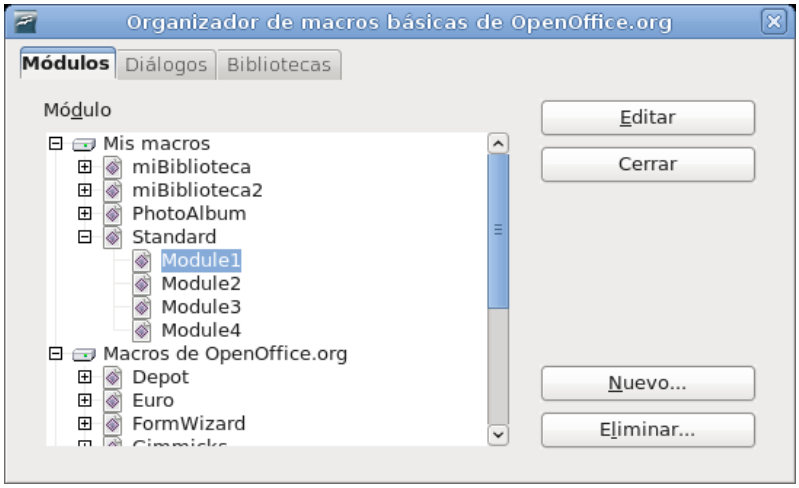
1. Icono catalogo de objetos: Nos muestra un cuadro de dialogo que nos permite navegar en forma de árbol entre los archivos abiertos y dentro de sus bibliotecas, módulos y macros disponibles, con un doble clic sobre la macro seleccionada nos lleva a ella o usando el botón Mostrar (circulo rojo en la imagen).



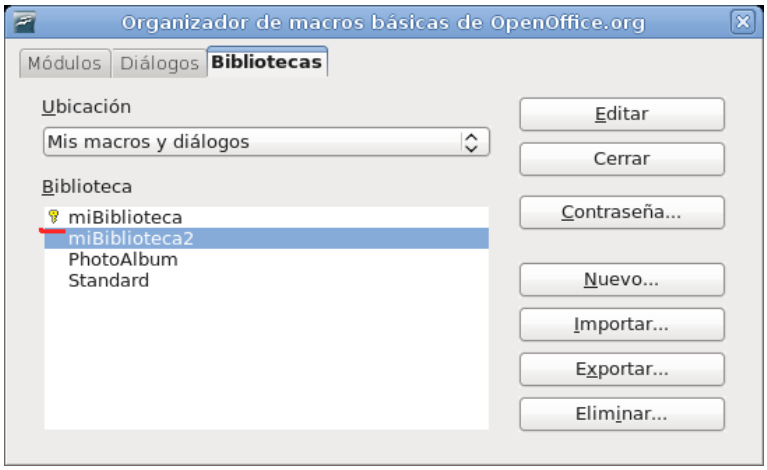
2. Icono de seleccionar macro: Nos muestra un cuadro de dialogo para administrar nuestras macros, nos permite también navegar entre los archivos abiertos, sus bibliotecas y sus módulos, por ahora, principalmente la hemos usados para ejecutar la macro que queramos, pero también podemos editar y eliminarlas, una opción muy interesante y poderosa es “Asignar” una macro, que se verá más adelante.



3. Icono seleccionar modulo: Nos muestra un cuadro de dialogo que nos permite administrar módulos, diálogos y bibliotecas, agregarlos y eliminarlos entre otras opciones como agregar una contraseña a una biblioteca.



Observa la llave al lado izquierdo de **miBiblioteca** que nos indica que esta biblioteca tiene establecida una contraseña para su visualización, la cual te solicitará si intentas editarla.

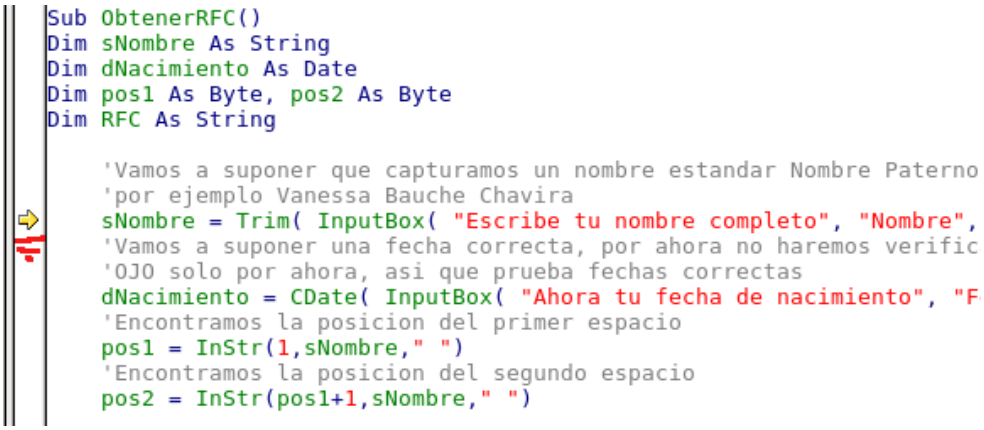


Veamos ahora cómo nos pueden ayudar los puntos de ruptura, la ejecución línea a línea, la ventana de observador y la ventana de llamadas a depurar nuestras macros. Puedes usar cualquier macro de las que hayamos desarrollado en estos apuntes o cualquiera que tengas, lo importante es seguir la secuencia correcta de los pasos siguientes, por ejemplo, el siguiente código, cópialo en la parte superior de cualquier módulo, es decir, asegúrate de que sea la "primer" macro de dicho módulo de código.

```
Option Explicit

'Construimos un seudo-RFC, el RFC es un dato personal y único en México y digo seudo
'por que no es tan simple como lo pongo aquí, pero nos sirve muy bien de ejemplo
'Primeras dos letras del apellido paterno
'Primera letra del apellido materno
'Primera letra del nombre
'Año de nacimiento de dos dígitos
'Mes de nacimiento de dos dígitos
'Dia de nacimiento de dos dígitos
Sub ObtenerRFC1()
Dim sNombre As String
Dim dNacimiento As Date
Dim pos1 As Byte, pos2 As Byte
Dim RFC As String

   'Vamos a suponer que capturamos un nombre estandar Nombre Paterno Materno
```

```
    'por ejemplo Vanessa Bauche Chavira
    sNombre = Trim( InputBox( "Escribe tu nombre completo", "Nombre" ))
    'Vamos a suponer una fecha correcta, por ahora no haremos verificación
    'OJO solo por ahora, así que prueba fechas correctas
    dNacimiento = CDate( InputBox( "Ahora tu fecha de nacimiento", "Fecha" ))
    'Encontramos la posición del primer espacio
    pos1 = InStr(1,sNombre," ")
    'Encontramos la posicion del segundo espacio
    pos2 = InStr(pos1+1,sNombre," ")

    RFC = UCase( Mid(sNombre,pos1+1,2) & Mid(sNombre,pos2+1,1) & Mid(sNombre,1,1) ) & _
          Right(CStr(Year(dNacimiento)),2) & Format(Month(dNacimiento),"00" ) & _
          Format(Day(dNacimiento),"00" )
    'Mostramos el RFC construido
    MsgBox RFC

End Sub
```

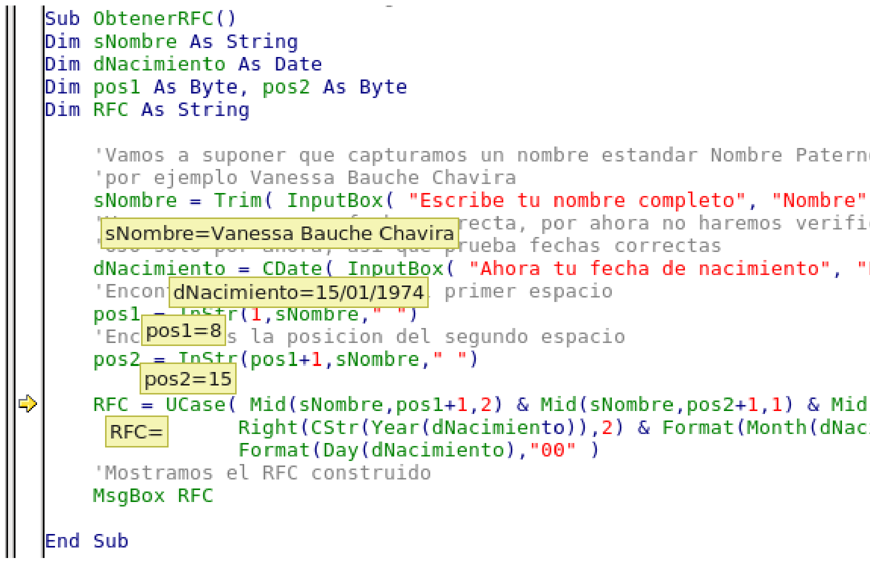
Tienes que ser muy observador, presiona la tecla F5, por supuesto nuestra macro se ejecutara de forma normal, captura los datos solicitados correctamente y verifica que el resultado sea el esperado. Ahora, presiona F8, nota cómo aparece una flecha amarilla en el margen izquierdo de nuestro editor, empezando por la primer línea de nuestra macro, continua presionando F8 y nota como la flecha amarilla va avanzando línea por línea en nuestro código, continúa así hasta terminar de ejecutar la macro.



Vuelve a empezar presionando F8, pero ahora, detente exactamente en la línea siguiente, que si copiaste el código anterior, tiene que ser la línea 28 aproximadamente.

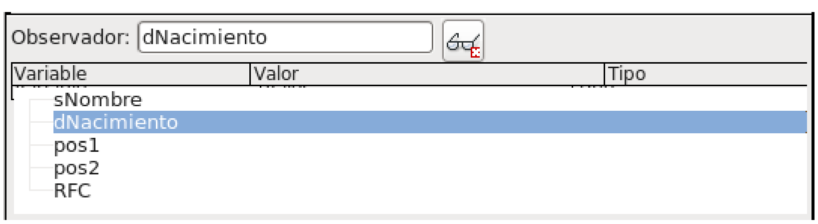
```
RFC = UCase( Mid(sNombre,pos1+1,2) & Mid(sNombre,pos2+1,1) & Mid(sNombre,1,1) ) & _
             Right(CStr(Year(dNacimiento)),2) & Format(Month(dNacimiento),"00" ) & _
             Format(Day(dNacimiento),"00" )
```

Posiciona el cursor sobre cada una de las variables sNombre, dNacimiento, pos1, pos2 y RFC, ¿que notas?, muy bien, muy bien, observa como el IDE nos muestra el valor almacenados en cada una de las variables en un pequeño cuadro de texto amarillo que desaparece en cuanto movemos el cursor. No esta de más aclararte que la imagen siguiente esta editada para mostrarte el valor de todas la variables.

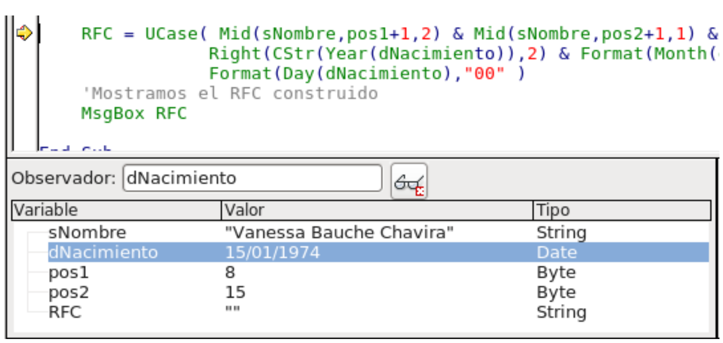


También observa que, mientras estas ejecutando el código, línea por línea, en cualquier momento puedes continuarla de forma normal presionando F5 o de plano detener la macro con el botón Detener Macro o presionando Ctrl+Shift+Q.

Con justa razón te preguntaras, ¿esto no es muy practico cuando tenemos decenas de variables? Y sí, tendrás razón, ahora para ello, para ver el valor de las variables, usaremos la ventana del observador de la siguiente manera. Selecciona una variable y presiona el icono “Habilitar inspección” o presiona la tecla F7 que tiene el mismo efecto, repite este proceso con todas las restantes variables y ve observando cómo en la ventana del observador se van listando las variables.

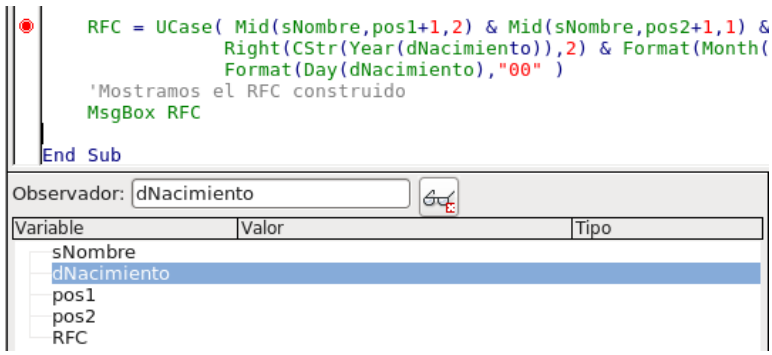


Ahora, vuelve a ejecutar la macro paso a paso, es decir, línea a línea y ve observando la ventana de inspección, al llegar a la línea donde construimos el RFC, tienes que ver algo muy similar a.

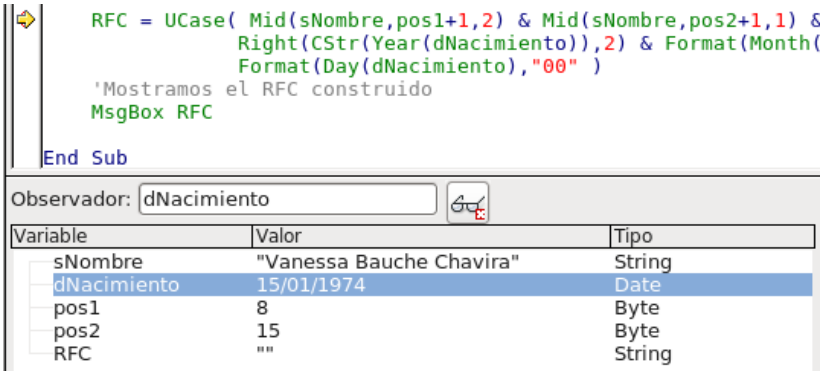


Mira que lindo, tenemos el valor de todas nuestras variables, además, te muestra el tipo de variable que es, por supuesto si estas variables cambian su valor en el transcurso de ejecución de nuestra macro, estos cambios los podrás ir analizando en la ventana del observador,

he ahí su importancia, una frecuente fuente de errores, es precisamente que en ocasiones al procesar los datos tan rápido, puede ser que una variable no tenga o no obtenga los datos que necesitamos. Y surge una nueva pregunta, ¿tenemos que ejecutar nuestro código línea a línea para poder ver los valores de las variables?, ¿y si tenemos decenas o centenas de líneas de código?, muy buena pregunta, veo que eres un alumno atento, para estos casos, en vez de ejecutar nuestro código línea a línea, usaremos puntos de interrupción, que, como vimos más atrás, son señales que le dejamos al IDE para que se detenga en la línea indicada. Para agregarlos usamos el icono “activar/desactivar puntos de ruptura” o presionamos la tecla F9, pero, “muy importante”, primero posicionamos el cursor sobre la línea donde nos interese establecer el punto de interrupción. Agrega un punto de interrupción en la línea donde construimos el RFC de modo que te quede de la siguiente manera.



Observa cómo aún tenemos las variables en la ventana del observador, ahora, ejecuta la macro con F5 y observa como después de alimentar los datos, el código se detiene exactamente en la línea que establecimos.



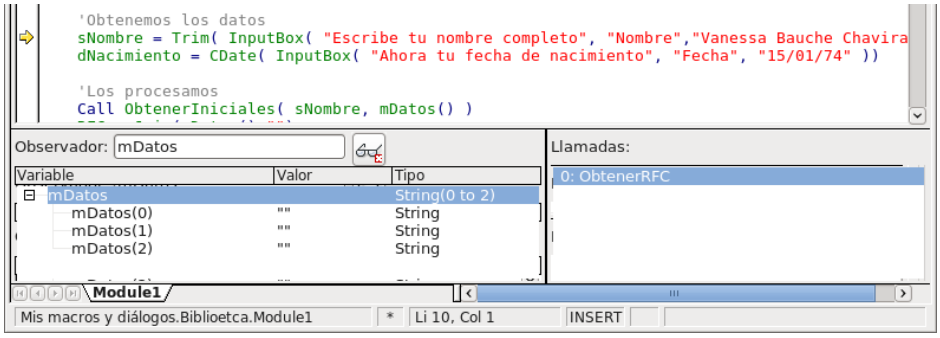
Ahora, puedes continuar ejecutando línea a línea (F8) o continuar normalmente (F5), puedes establecer tantos puntos de interrupción como quieras o necesites. Las variables que agregues a la ventana del observador y los puntos de interrupción que establezcas, solo estarán disponibles en tu sesión actual, si cierras tu archivo los perderás. En cualquier momento puedes quitar los puntos de interrupción con F9 sobre la línea donde esta el punto de interrupción o las variables de la ventana del observador, solo selecciona la variable que quieras quitar en la ventana del observador y la eliminas de esta con el icono que esta inmediatamente encima de estas.

Para terminar este tema tan divertido, veamos para qué sirve la ventana de llamadas y veamos cómo la ventana de observador también nos puede mostrar el valor de una matriz de datos, para esto, modifica el código anterior para dividirlo en una subrutina y en una función, de modo que quede de la siguiente manera.

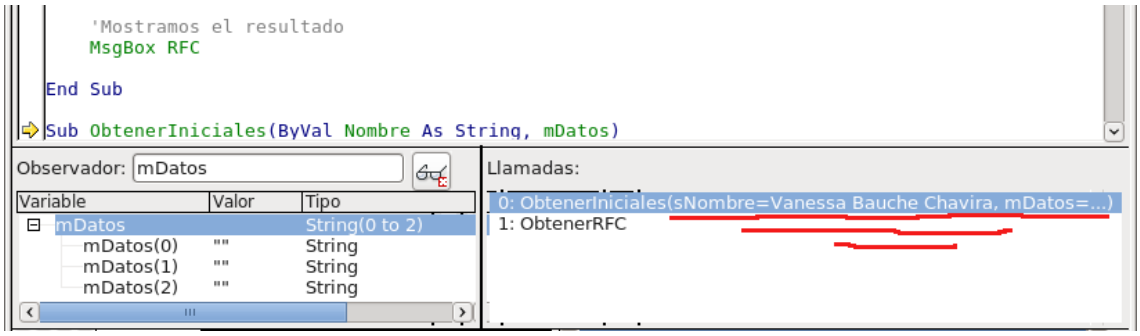
```
Option Explicit

Sub ObtenerRFC2()
```

```
Dim sNombre As String
Dim dNacimiento As Date
Dim RFC As String
Dim mDatos(2) As String

   'Obtenemos los datos
   sNombre = Trim( InputBox( "Escribe tu nombre completo", "Nombre","Vanessa Bauche Chavira" ))
   dNacimiento = CDate( InputBox( "Ahora tu fecha de nacimiento", "Fecha", "15/01/74" ))

   'Los procesamos
   Call ObtenerIniciales( sNombre, mDatos() )
   RFC = Join(mDatos(),"")
   RFC = RFC & FechaRFC( dNacimiento )

   'Mostramos el resultado
   MsgBox RFC

End Sub

Sub ObtenerIniciales(ByVal Nombre As String, mDatos)
Dim pos1 As Byte, pos2 As Byte

   pos1 = InStr(1,Nombre," ")
   pos2 = InStr(pos1+1,Nombre," ")

   mDatos(0)=UCase(Mid(Nombre,pos1+1,2))
   mDatos(1)=UCase(Mid(Nombre,pos2+1,1))
   mDatos(2)=UCase(Mid(Nombre,1,1))

End Sub

Function FechaRFC(ByVal Nacimiento As Date) As String

   FechaRFC= Right(CStr(Year(Nacimiento)),2) & Format(Month(Nacimiento),"00" ) &
Format(Day(Nacimiento),"00" )

End Function
```

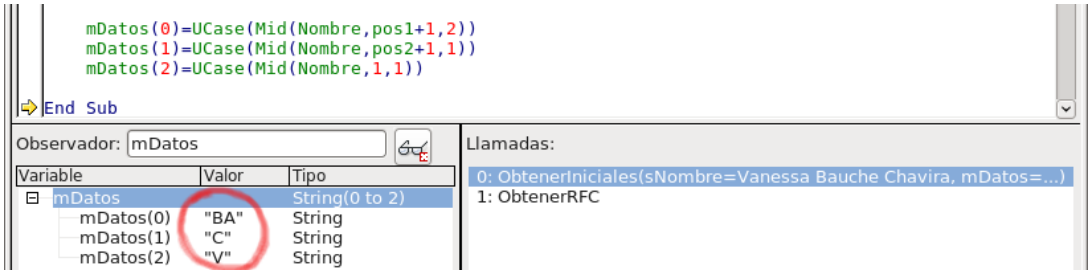
Ejecuta el código primero (F5) para verificar que obtenemos los resultados esperados, después agrega la variable mDatos a la ventana del observador (F7) y ejecuta el código paso a paso (F8), detente en la primer línea donde aparece la variable sNombre, para tener algo muy similar a la siguiente imagen.



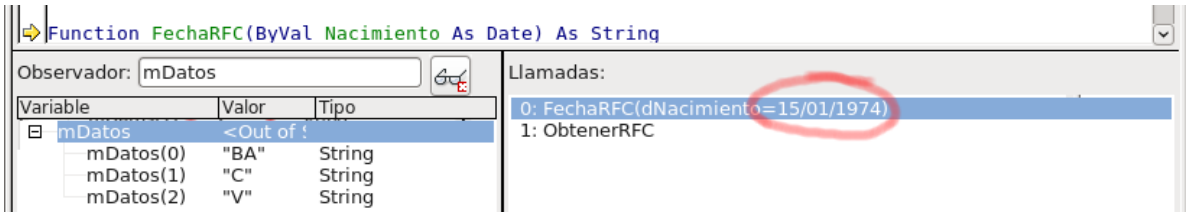
Tal vez en la ventana del observador tendrás que darle un clic al símbolo del signo más a la izquierda del nombre de la variable para que te despliegue el contenido de la matriz, que, en esta caso, esta vacía. Continúa ejecutando paso a paso (F8) y detente en.



Reitero, tienes que ser muy observador, nota cómo al llegar a la línea donde llamamos a la subrutina, el código salta a dicha subrutina y muy importante, observa la ventana de llamadas cómo te muestra el nombre de la subrutina y además, los valores que se le están pasando, muy útil para saber si estamos estableciendo correctamente los valores. Continúa la ejecución paso a paso y detente en.



Observa cómo ahora la matriz ya tiene datos y estos se muestran en la ventana del observador. Continua paso a paso hasta el inicio de la función FechaRFC y observa la ventana de llamadas que una vez más muestra la función y sus valores pasados.



Termina la ejecución de la macro (F5) y observa que la pila de llamadas se vacía, así como el valor de las variable en la ventana del observador. Y sin querer queriendo dimos un breve repaso a varios temas vistos anteriormente. No se te olvide que las teclas F5 y F8 solo tienen efecto en la primer macro declarada en el modulo activo, para ejecutar paso a paso otras macros, establece primero un punto de ruptura en la linea que te interese dentro de dicha macro y después ejecutas la macro de forma normal para que se detenga en el punto de ruptura establecido, a partir de ahí ya puedes usar F8 para continuar paso a paso...

Más adelante, aprenderemos que el Observador también nos sirve para analizar “objetos”.

!!Feliz programación¡¡

5 Trabajando con OpenOffice.org

A mi juicio, entender como esta construido OpenOffice.org, es la diferencia entre aprovechar todo el potencial que tiene OOo Basic o perderte en un laberinto infranqueable por andar buscando "similitudes" con "otros" programas. Tal vez pecando de un excesivo reduccionismo, OpenOffice.org es un LEGO, el famoso juego de construcción con piezas de plástico, que si no sabes que es, te estas perdiendo de un maravilloso juego. Cada pieza del LEGO es una pieza completa, pero que se puede embonar perfectamente bien con miles de piezas más para "armar" y "crear" nuevos componentes, eso es OpenOffice.org precisamente, esta divido en una finita y conocida cantidad de partes, a cada una de estas partes, le podemos llamar "objeto", pero en el lenguaje de OpenOffice.org se le denomina "***servicio***", que no se te olvide "***servicio***", ya que este será el nombre que usaremos de aquí en adelante, a su vez, estos "servicios" se organizan en "módulos", pero ojo, no confundas estos "módulos", con los vistos en el capitulo 2, de hecho, esta forma de organización no tiene mayor relevancia para los que programamos en OO Basic, no así para los que usan otros lenguajes con OpenOffice.org (Java, C++, etc.), así que solo como conocimiento recuerdalo y concentrate en los "***servicios***". A su vez, estos "***servicios"***, pueden contener otros "***servicios"***, por lo que si sabes usar un "***servicio"*** y este esta implementado en otro componente, puedes aplicar tus conocimientos de este "***servicio"*** en el nuevo componente sin ningún problema. En OpenOffice.org, a través de OOo Basic, hay forma de saber que "***servicios"*** implementa un componente, lo cual, por su importancia y por la falta de documentación en español es lo que trataremos de aprender a hacer de la mejor forma, también, para que recuerdes que "no hay que estar adivinando" nada, sabiendo "preguntar" lo correcto y consultando el lugar correcto, podemos profundizar óptimamente en las entrañas de OpenOffice.org, mi propósito en este capitulo, es enseñarte a "preguntarle" a los "***servicios***" la información necesaria para ponerlos a trabajar.

NOTA MUY IMPORTANTE

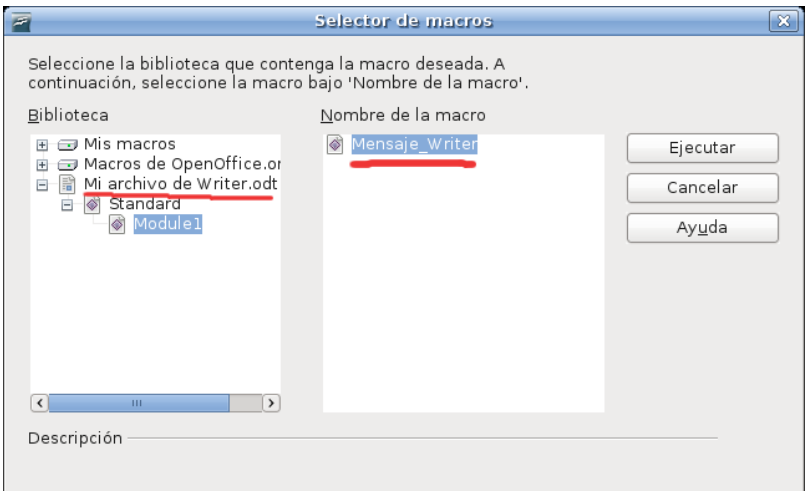
Te sugiero ampliamente, no continuar con los siguientes capítulos, hasta no haber comprendido cabalmente este, así mismo, doy por sentado que no tienes dudas de lo visto y explicado en capítulos anteriores, se que mis herramientas pedagógicas no son las mejores, pero estoy en la mejor disposición de ayudarte a resolver tus dudas, así que anda, si tienes alguna, vuelve a leer el tema, si aun queda alguna, envíame lo mas claramente posible, que es lo que no queda claro, por supuesto, si tienes algún aporte para ayudar a transparentar este tema, no dejes de mencionarlo.

5.1 Mis macros – un archivo especial.

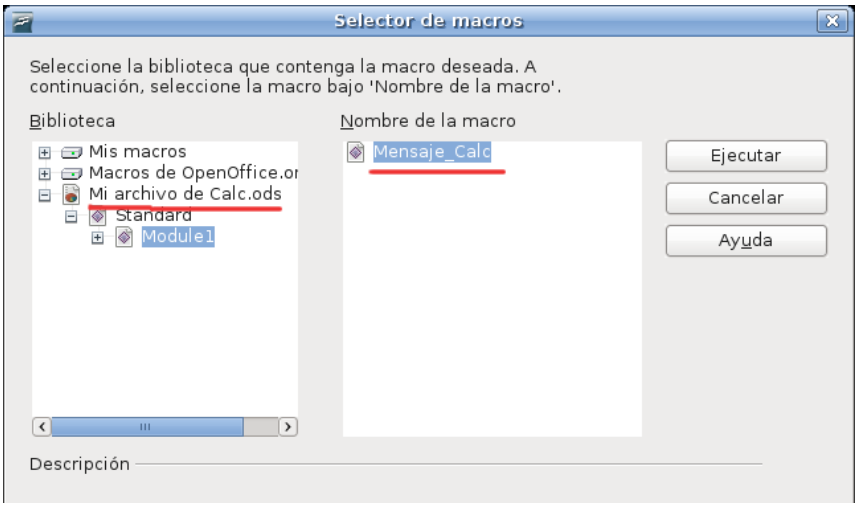
Hasta ahora, la mayoría de las macros que hemos desarrollado y probado, las hemos escrito “dentro” de un documento de OpenOffice.org, si has seguido estas notas desde el inicio, propusimos un archivo de Calc, aunque muy bien pudiste haber usado un archivo de Writer. Como se que eres observador, seguro habrás notado cada vez que habrías tu archivo para editar las macros o ejecutabas alguna, aparte del archivo que estuvieras usando, el organizador de macros te mostraba o bien, todos los archivos que tienes abiertos o solo el archivo activo para ejecutar una macro, veamos en que casos te mostraba uno y en que casos lo otro, esta diferencia es importante para el desarrollo de los siguientes temas. Para empezar a clarificar esto, crea dos nuevos archivos, uno de Calc y otro de Writer, guardarlos con el nombre que quieras, agregales una macro de prueba que nos muestre un mensaje, por ejemplo.

```
Option Explicit

Sub Mensaje_Calc()

   MsgBox "Estoy en mi archivo de Calc"

End Sub

Sub Mensaje_Writer()

   MsgBox "Estoy en mi archivo de Writer"

End Sub
```

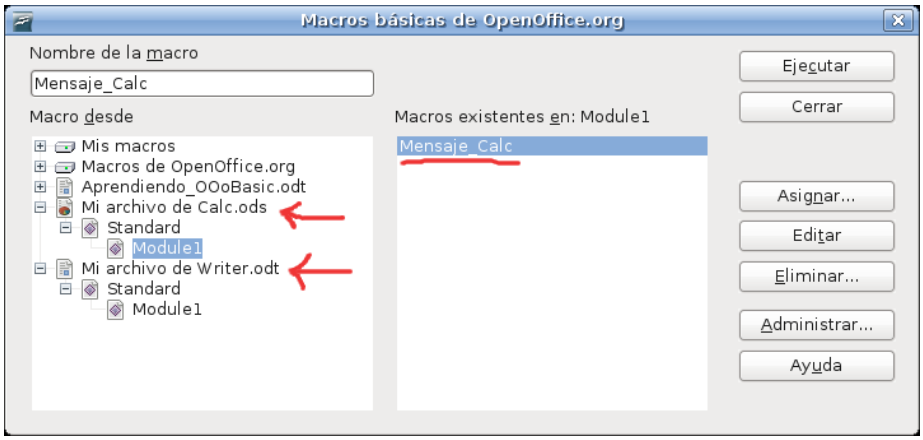
Toma nota de que son dos macros en “dos” archivos diferentes. Ahora, trata de ejecutar cada macro desde el menú Herramientas | Macros | Ejecutar macros..., intentemos primero con la macro que esta en el archivo de Writer, observa como solo nos muestra el archivo de Writer y no el de Calc.



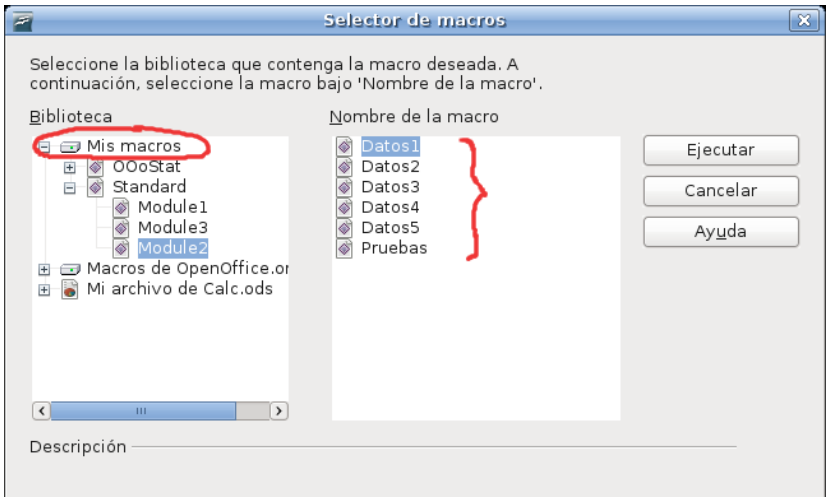
Ahora la macro que esta en el archivo de Calc y observa como solo nos muestra el archivo de Calc y no el de Writer.



Por supuesto, podemos "forzar" que nos muestre todos los archivos abiertos, si, en vez de ir al menú *Herramientas | Macros | Ejecutar macros...*, optamos por ir al menú *Herramientas | Macros | Organizar macros | OpenOffice.org Basic...*, que nos mostrará:



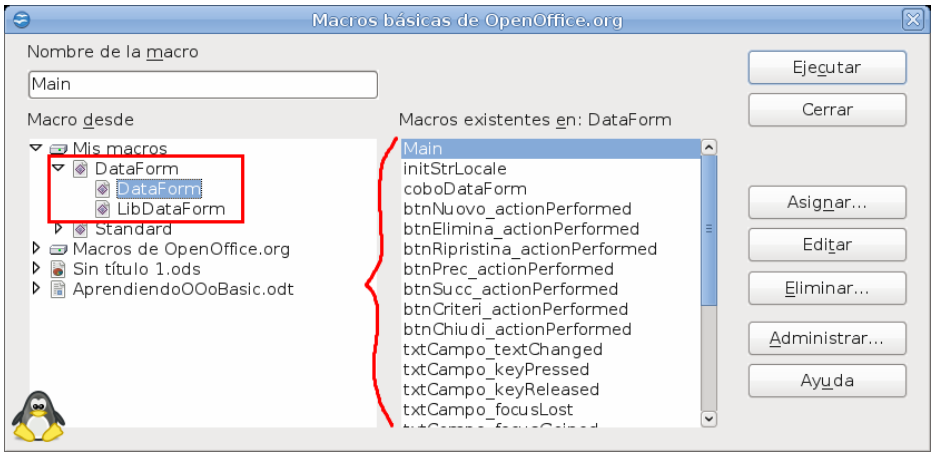
Ahora si, observa como nos muestra todos los archivos abiertos y podemos tanto editar las macros que contenga o ejecutarlas. Para que tengamos disponibles nuestras macros con cualquier archivo y poder ejecutarlas desde *Herramientas | Macros | Ejecutar macros...*, podemos optar por usar el archivo especial que se llama Mis macros, que puedes ver aquí:



Dentro de el, puedes crear todas las bibliotecas que quieras y todos los módulos que quieras, así como los diálogos que aprenderemos a crear y programar más adelante. A partir de aquí, queda a tu criterio decidir donde guardas tus macros, si dentro de archivos individuales o dentro de Mis macros. Este archivo se encuentra en el directorio del usuario dentro de la carpeta oculta (en GNU/Linux) de OpenOffice.org, por ejemplo:

En GNU/Linux: /home/USUARIO/.openoffice.org/3/user/basic.
En Windows 7: C:\Users\USUARIO\AppData\Roaming\OpenOffice.org\3\user\basic

En este directorio podrás ver como carpetas las bibliotecas que vayas agregando y dentro de ellas los módulos creados, por supuesto, es muy útil respaldar de vez en cuando esta carpeta. En esta carpeta también se guardarán la mayoría de las extensiones que instales, en la siguiente imagen vemos una extensión instalada.

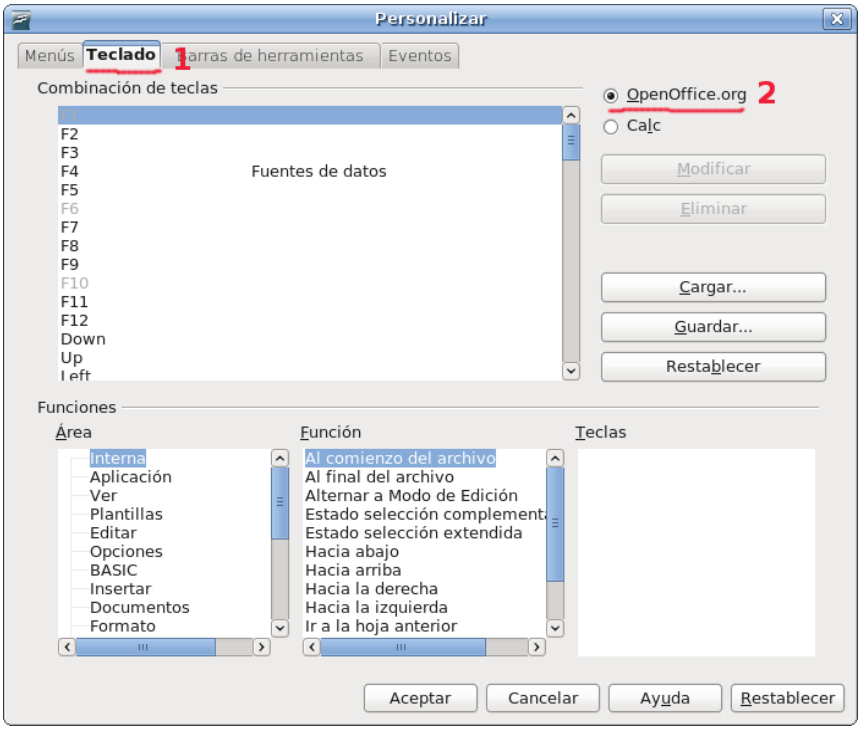


5.2 Asignando macros

Hasta ahora, hemos ejecutado todas nuestras macros desde el IDE o llamándolas desde el menú herramientas, pero también podemos asignar las macros para ejecutarlas a menús, barras de herramientas, combinaciones de teclas e incluso asignarlas a sucesos de la aplicación, por ejemplo; cuando abrimos un archivo, esto es sumamente útil y poderoso, veamos algunos ejemplos de como hacer esto, como primer paso, hay que escribir la macro que nos interesa asignar, algo tan sencillo como mostrar la fecha y hora actuales, claro, con un poco de formato para que sea vea bien:

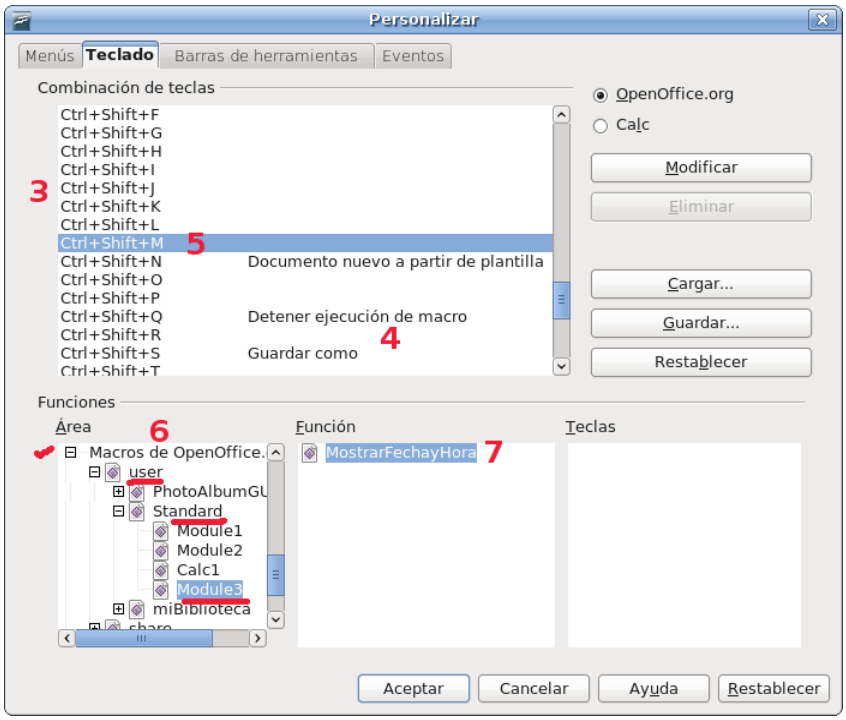
```
Sub MostrarFechayHora()
Dim sMensaje As String

   sMensaje = Format( Now(), "dddd dd \d\e mmmm \d\el yyyy " & Chr(13) & _
                        "y \son la\s HH \ho\ra\s con mm \minuto\s"

   MsgBox sMensaje, 64,"Hoy es"

End Sub
```

Ahora, tienes que decidir “cuando” o “como” quieres ejecutarla, veamos primero lo más sencillo, hagamos que se ejecute con una combinación de teclado, para ello, desde cualquier aplicación de OpenOffice.org, ve al menú ***Ver*** | ***Barra de herramientas*** | ***Personalizar...*** que te mostrara el siguiente cuadro de dialogo:



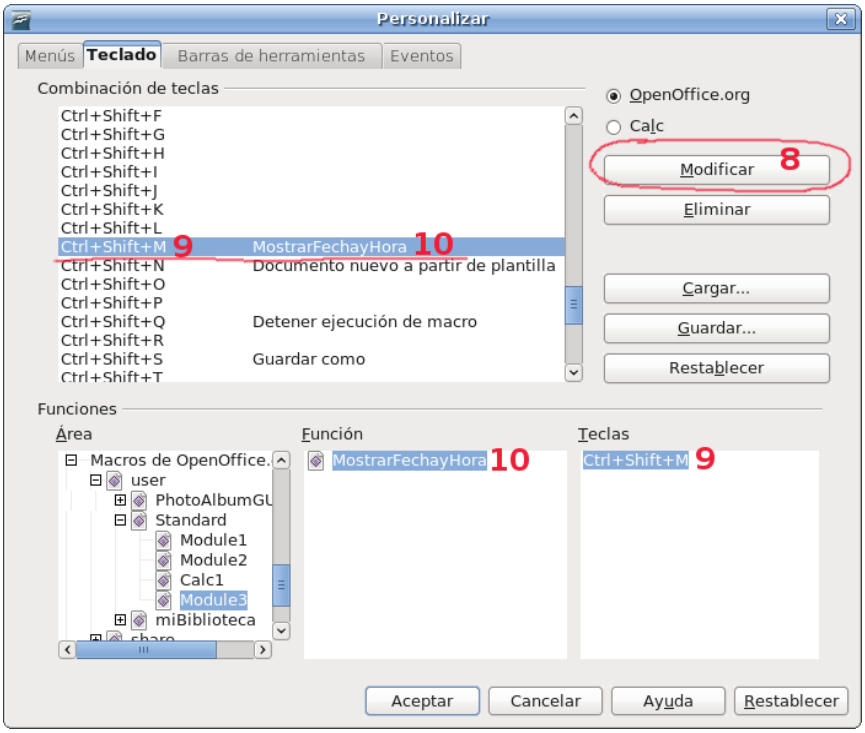
Asegurate de que este seleccionada la ficha Teclado (1), así como mostrar las Combinaciones de teclas de OpenOffice.org (2). El siguiente paso es que escojas la combinación de teclas que quieres asignar a nuestra macro, para ello, desplazate en el cuadro de lista Combinación de teclas (3) y escoge la que más te guste, cuida de seleccionar una que este libre,

te va a dar cuenta las que ya están en uso, por que delante de ellas tienen la tarea asignada (4), también notaras que algunas teclas, como por ejemplo la tecla F1, están en gris desactivadas, esto es por que estas teclas no puedes reasignarlas. Pero observa que tienes una amplia gama de opciones de donde escoger, para nuestro ejemplo, hemos seleccionado CTRL + SHIFT + M (5), el siguiente paso es seleccionar la macro que deseamos ejecutar con esta combinación de teclas, para nuestro ejemplo, primero desplazate en el cuadro de lista Área (6) hasta encontrar la opción Macros de OpenOffice.org, despliega su contenido y seleccionar “user” que se refiere al archivo especial “Mis Macros”, solo te resta seleccionar la biblioteca y el modulo donde esta la macro a asignar, para nuestro ejemplo se llama MostrarFechayHora (7).

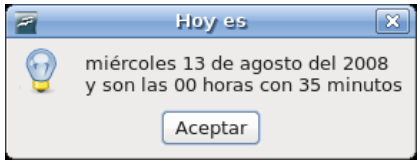


Observa como tienes opción de cargar combinaciones de teclas guardadas, así como de guardar las actuales y restablecer las predeterminadas, te queda de tarea “jugar” y experimentar con estas opciones.

Como ultimo paso, presiona el botón de comando Modificar (8) que asigna finalmente la combinación de teclas (9) a la macro seleccionada (10). Para terminar, presiona el botón de comando Aceptar.



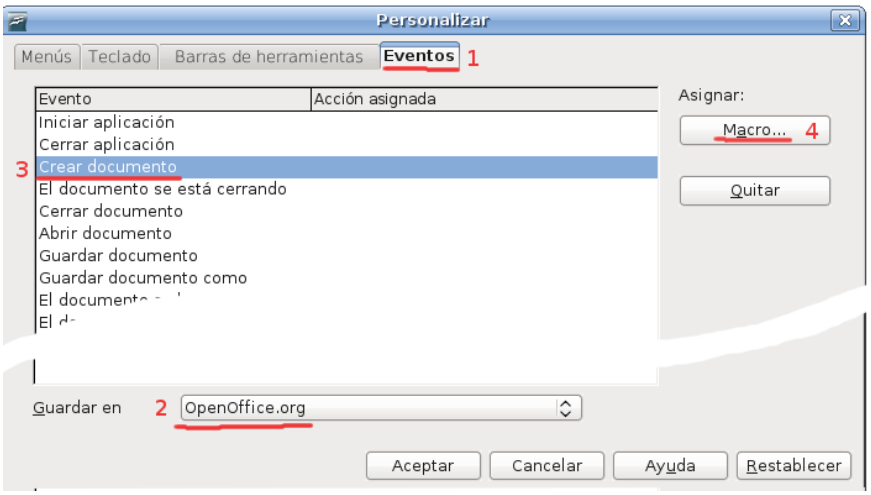
Ahora, solo nos resta probar que funcione, desde cualquier documento, presiona la combinación de teclas seleccionada (Ctrl+Shift+M) y si todo esta correcto, te tiene que mostrar:



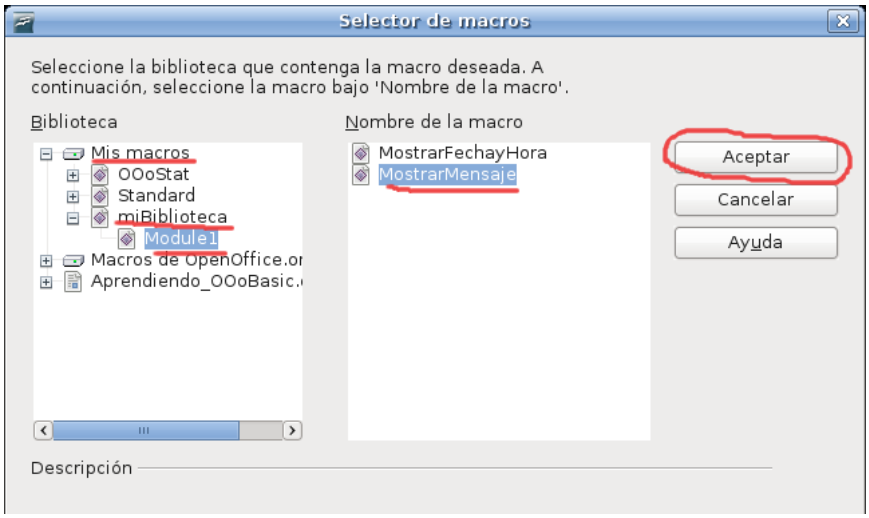
Con un poco de imaginación puedes crear tus accesos directos para probar todas las macros que quieras de una forma sencilla. Ahora veamos como asignar una macro a un evento, a un suceso del sistema, para ello, vamos a crear una sencilla macro de ejemplo:

```
Sub MostrarMensaje()
Dim sMensaje As String

   sMensaje = "No te olvides de guardar tu documento"

   MsgBox sMensaje, 64,"Creando documento"

End Sub
```

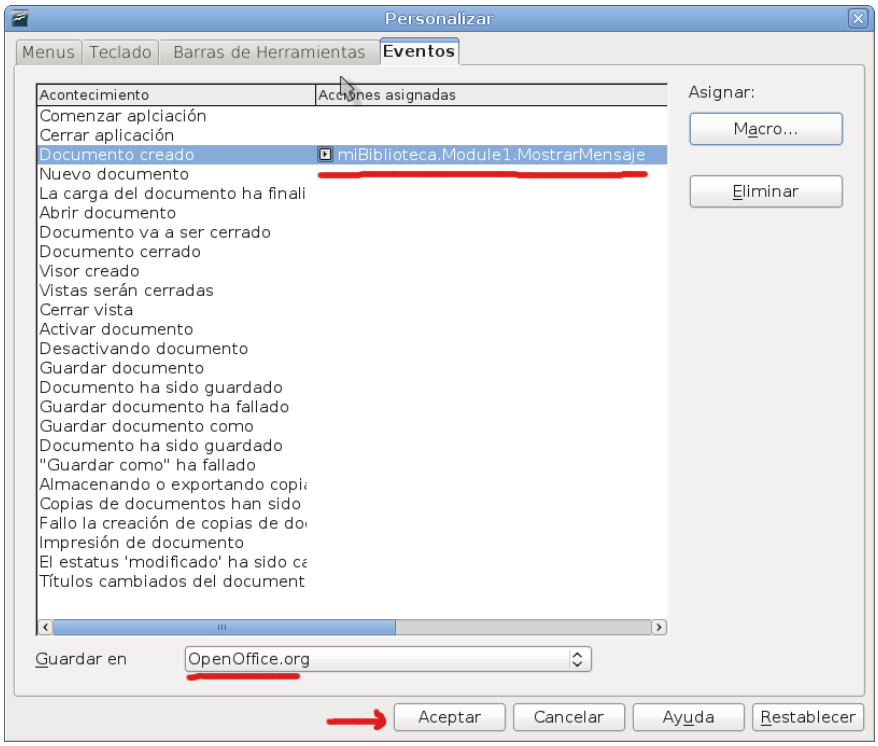
Ve al menú *Ver | Barra de herramientas | Personalizar...*, esta vez, selecciona la ficha Eventos (1) y asegurate de seleccionar OpenOffice.org (2) para guardar la asignación que hagamos. Para nuestro ejemplo seleccionaremos el "**evento**" Crear documento (3) que se ejecuta cada vez que creamos un nuevo documento.



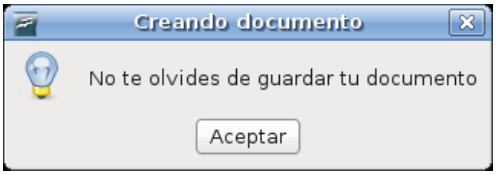
Ahora, con el botón de comando Macro... (4), te desplegara un cuadro de dialogo que ya conoces, solo te resta navegar dentro del archivo, la biblioteca y el modulo que contiene la macro deseada, selecciona la macro, presiona Aceptar y listo.



Observa como delante del “evento” queda seleccionada la macro, solo te resta presionar el botón de comando Aceptar y probar la macro, crea tantos nuevos documentos como quieras.



Con cada nuevo documento creado, de cualquier aplicación de OOo, te tiene que mostrar el mensaje:



Asignar macros a eventos no es un tema menor, pues puede ser tan sencillo como nuestros ejemplos o tan complejo como conectarte a bases de datos o a una página web y muchas cosas más, más adelante veremos a detalle el tema de los eventos y la asignación de macros. Asignar una macro a un menú o a una barra de herramientas es trivial, así que te queda de tarea intentar hacerlo.

5.3 Creando nuestro primer “servicio” (objeto).

Para crear, acceder y manipular un servicio, “necesitas” asignarlo a una variable de objeto, de la siguiente manera.

```
Dim NombreVariable As Object
```

E inicializarla para poder usarla, para ello, usamos la función *createUnoService*.

```
Sub AccediendoAOpenOffice1()
Dim appOpenOffice As Object

   appOpenOffice = createUnoService( "com.sun.star.frame.Desktop" )

End Sub
```

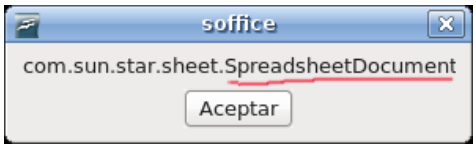
El servicio com.sun.star.frame.Desktop es usado muy frecuentemente, pues nos permite acceder a “toda” la aplicación y sus componentes, por ello, existe una palabra clave que nos da acceso inmediato a este “servicio”; **“StarDesktop”** por lo que, la siguientes dos líneas son equivalentes.

```
   appOpenOffice = createUnoService( "com.sun.star.frame.Desktop" )

   appOpenOffice = StarDesktop
```

Ejecutando la macro anterior, por ahora, no veras ningún efecto, pero aquí empieza lo divertido e interesante, vamos a explorar por dentro del OpenOffice.org, modifica la macro para que quede de la siguiente manera.

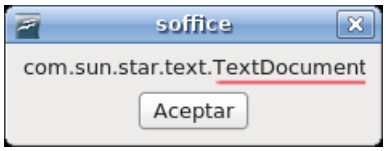
```
Option Explicit

Sub AccediendoAOpenOffice2()
Dim appOpenOffice As Object
Dim oActivo As Object

   'Inicializamos una variable con el servicio com.sun.star.frame.Desktop
   appOpenOffice = StarDesktop

   'Obtenemos una referencia al componente activo
   oActivo = appOpenOffice.CurrentComponent

   'Mostramos el tipo de componente
   MsgBox oActivo.Identifier

End Sub
```

Ahora si, al ejecutar te tiene que mostrar el siguiente mensaje.



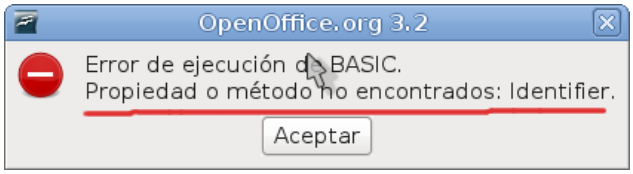
Pero, tal vez te muestre este otro.

Incluso tal vez el siguiente.



¿Por qué? Porque dependerá desde donde “llames” (ejecutes) la macro, verifícalo, tal vez la ejecutaste directamente desde el IDE, o tal vez la llamaste desde una hoja de calculo o desde un archivo de texto. En el tema anterior vimos como llamar a una macro desde donde queramos, aunque lo más practico para los siguientes ejemplos es que programes tus macros dentro del archivo “*Mis macros*”, queda a tu criterio.

Abre seis archivos, uno por cada tipo de documento que podemos manejar en OpenOffice.org (hoja de calculo, documento de texto, presentación, dibujo, base de datos y formula) no necesariamente tienes que guardarlos, pero no estaría de más por que los vamos a usar en repetidas ocasiones. Ejecuta la macro anterior desde cada uno de estos documentos. Todos excepto la base de datos te tienen que mostrar el mensaje correctamente y en el caso de la base de datos, te tiene que mostrar el error siguiente.



Aprendete muy bien estas dos nuevas palabras “**propiedad**” y “**método**”, las vas a usar de aquí en adelante y siempre que programes en OOo Basic. En este caso, es una “**propiedad**” lo que esta mal, es decir, la propiedad **Identifier**, no la implementan las bases de datos. Sabiendo esto, podríamos empezar a proponer una macro genérica para saber que tipo de documento es el activo, probemos con la siguiente.

```
Sub AccediendoAOpenOffice3()
Dim appOpenOffice As Object
Dim oActivo As Object

   'Inicializamos una variable con el servicio com.sun.star.frame.Desktop
   appOpenOffice = StarDesktop

   'Obtenemos una referencia al componente activo
   oActivo = appOpenOffice.CurrentComponent

   'Mostramos el tipo de componente
   Select Case oActivo.Identifier
         Case "com.sun.star.sheet.SpreadsheetDocument"
               MsgBox "Soy una hoja de calculo"
         Case "com.sun.star.presentation.PresentationDocument"
               MsgBox "Soy una presentacion"
         Case "com.sun.star.drawing.DrawingDocument"
               MsgBox "Soy un dibujo"
         Case "com.sun.star.formula.FormulaProperties"
               MsgBox "Soy una formula"
         Case "com.sun.star.text.TextDocument"
               MsgBox "Soy un documento de texto"
   End Select

End Sub
```

Ejecuta la macro anterior desde cada uno de los seis archivos, excepto desde la base de datos pues te dará un error, claro, puedes, si quieres, implementar un controlador de errores pero por ahora no lo haremos, busquemos otra alternativa, por ejemplo, probemos ahora la propiedad **ImplementationName** y veamos que pasa,

```
Sub AccediendoAOpenOffice4()
Dim oActivo As Object

   'Obtenemos una referencia al componente activo
   oActivo = StarDesktop.CurrentComponent

   'Mostramos el tipo de componente
   MsgBox oActivo.ImplementationName

End Sub
```

Esta vez si nos muestra un mensaje en todos los documentos, pero notaras que esta propiedad en algunos documentos no es muy clara, aquí el resumen de lo que tendrías que obtener en cada caso:

Aplicación	*Tipo de documento*	*Propiedad ImplementationName*
Writer	Documento de texto	SwXTextDocument
Calc	Hoja de calculo	ScModelObj
Impress	Presentaciones	SdXImpressDocument
Draw	Dibujo	SdXImpressDocument
Base	Base de datos	com.sun.start.comp.dba.ODatabaseDocument
Math	Formula	com.sun.start.comp.math.FormulaDocument

¿Te fijas que interesante?, nota el valor devuelto en las hojas de calculo y como las presentaciones y los dibujos devuelven el mismo resultado, por lo que esta propiedad, a pesar de estar implementada en todos los tipos de documentos, no resulta muy idónea para diferenciar a cada uno. ¿Y si en vez de devolver una cadena de texto, le preguntamos a cada componente si son lo que dicen ser?, es decir, por ejemplo, una hoja de calculo implementa propiedades y métodos (servicios) para manipular hojas de calculo, en este caso, este servicio se llama "com.sun.star.sheet.SpreadsheetDocument", y cada documento soporta un tipo de servicio diferente. Todos los objetos admiten un método que nos permite preguntar si dicho objeto soporta un servicio en especial, este método se llama **supportsService** y hay que pasarle como argumento el nombre del servicio que queremos validar y nos devolverá verdadero (True) o falso (False) según soporte o no el servicio, por lo que la sintaxis competa de este método es:

```
Variable As Boolean = Objeto.supportsService( Nombre_del_Servicio As String)
```

Veamoslo trabajando en el siguiente ejemplo.

```
Sub AccediendoAOpenOffice5()
Dim oActivo As Object
Dim bImplementaServicio As Boolean

   oActivo = StarDesktop.CurrentComponent

   bImplementaServicio = oActivo.supportsService("com.sun.star.sheet.SpreadsheetDocument")

   If bImplementaServicio Then
         MsgBox "Soy una hoja de calculo"
   Else
         MsgBox "NO soy una hoja de calculo"
```

```
   End If

End Sub
```

Y prueba desde todos nuestros documentos que funciona correctamente por lo que ya podemos implementar una macro mejor para saber que tipo de documento tenemos activo, simplemente consultando si soporta su respectivo “*servicio”.*

```
Sub AccediendoAOpenOffice6()
Dim oActivo As Object

   oActivo = StarDesktop.CurrentComponent

   If oActivo.supportsService("com.sun.star.sheet.SpreadsheetDocument") Then
         MsgBox "Soy una hoja de calculo"
   ElseIf oActivo.supportsService("com.sun.star.text.TextDocument") Then
         MsgBox "Soy un documento de texto"
   ElseIf oActivo.supportsService("com.sun.star.presentation.PresentationDocument") Then
         MsgBox "Soy una presentacion"
   ElseIf oActivo.supportsService("com.sun.star.drawing.DrawingDocument") Then
         MsgBox "Soy un dibujo"
   ElseIf oActivo.supportsService("com.sun.star.sdb.OfficeDatabaseDocument") Then
         MsgBox "Soy una base de datos"
   ElseIf oActivo.supportsService("com.sun.star.formula.FormulaProperties") Then
         MsgBox "Soy una formula"
   ElseIf oActivo.supportsService("com.sun.star.script.BasicIDE") Then
         MsgBox "Soy el editor de codigo IDE"
   Else
         MsgBox "NO se que tipo de documento soy"
   End If

End Sub
```

Ahora si, ya podemos discriminar correctamente los documentos. Saber si un objeto implementa un determinado servicio, es muy importante para minimizar el riesgo de errores y para consultar directamente las propiedades y métodos que implementa dicho servicio sin tener que estar “adivinando” nada y precisamente para no estar adivinando, existe una propiedad que nos devuelve una matriz (array) con los servicios que soporta un objeto, esta propiedad se llama **SupportedServiceNames** y su sintaxis es.

```
Variable As Array = Objeto.SupportedServiceNames
```

Como siempre, no hay como los ejemplos.

```
Sub AccediendoAOpenOffice7()
Dim oActivo As Object
Dim mServicios() As String
Dim co1 As Byte

   'Obtenemos una referencia al componente activo
   oActivo = StarDesktop.CurrentComponent

   'Obtenemos una matriz con los servicios que implementa
   mServicios() = oActivo.SupportedServiceNames
   'Y mostramos el nombre de cada servicio
   For co1 = LBound( mServicios() ) To UBound( mServicios() )
         MsgBox mServicios( co1 )
   Next

End Sub
```

Y si ejecutas la macro anterior desde cada uno de nuestros seis documentos abiertos, tienes que obtener la siguiente lista de servicios en cada uno.

Aplicación	*Tipo de documento*	*Propiedad SupportedServiceNames*
Writer	Documento de texto	com.sun.star.document.OfficeDocument com.sun.star.text.GenericTextDocument com.sun.star.text.TextDocument
Calc	Hoja de calculo	com.sun.star.sheet.SpreadsheetDocument com.sun.star.sheet.SpreadsheetDocumentSettings
Impress	Presentaciones	com.sun.star.document.OfficeDocument com.sun.star.drawing.GenericDrawingDocument com.sun.star.drawing.DrawingDocumentFactory com.sun.star.presentation.PresentationDocument
Draw	Dibujo	com.sun.star.document.OfficeDocument com.sun.star.drawing.GenericDrawingDocument com.sun.star.drawing.DrawingDocumentFactory com.sun.star.drawing.DrawingDocument
Base	Base de datos	com.sun.star.document.OfficeDocument com.sun.star.sdb.OfficeDatabaseDocument
Math	Formula	com.sun.star.document.OfficeDocument com.sun.star.formula.FormulaProperties

Nota como Impress y Draw implementan casi los mismos servicios y observa como hay un servicio (com.sun.start.document.OfficeDocument), que, excepto en la hoja de calculo, es común a todos los documentos, pero no creas que la hoja de calculo no lo implementa, podemos demostrar que lo implementa con el siguiente ejemplo que tienes que ejecutar desde cualquier hoja de calculo, en todos los demás documentos también te funcionara, pero recuerda que solo queremos comprobar si la hoja de calculo implementa este servicio.

```
Sub AccediendoAOpenOffice8()
Dim oActivo As Object
Dim bImplementaServicio As Boolean

   oActivo = StarDesktop.CurrentComponent

   bImplementaServicio = oActivo.supportsService("com.sun.star.document.OfficeDocument")

   If bImplementaServicio Then
        MsgBox "Verdad que si esta implementado el servicio"
   Else
        MsgBox "NO esta implementado el servicio"
   End If

End Sub
```

Este servicio común a todos los documentos, también es muy usado e importante como más adelante podrás comprobarlo. Por ahora, copia la siguiente macro y pruebala de nuevo con todos los documentos.

```
Sub AccediendoAOpenOffice9()
Dim oActivo As Object
Dim mServicios() As String
Dim co1 As Byte

   oActivo = StarDesktop.CurrentComponent
```

```
    mServicios() = oActivo.getSupportedServiceNames()
    For co1 = LBound( mServicios() ) To UBound( mServicios() )
          MsgBox mServicios( co1 )
    Next

End Sub
```

Notaras que trabaja exactamente igual que la anterior, pero ve que no hemos usado la propiedad **SupportedServiceNames**, sino el método **getSupportedServiceNames()**, diferenciados por el prefijo get y los paréntesis, los dos hacen lo mismo, pero en OOo Basic, casi todos los métodos tienen su correspondencia en una propiedad, si te es útil, recuerda tus clases elementales de español, los objetos son como los sustantivos, las propiedades son adjetivos y los métodos son verbos, así mismo y muy importante, las propiedades siempre son de un tipo de dato soportado por OOo Basic (string, integer, array, etc.), por lo que tienes que tener las mismas consideraciones (y precauciones) vistas en varios temas atrás, principalmente cuando vimos variables y funciones, así mismo los métodos, cuando se les pasan argumentos y cuando devuelven un valor se ven afectados por estas consideraciones, por lo que reitero, ten cuidado con los tipos de datos que usas en ellos.

A lo largo de este capitulo, te mostrare, en la medida de lo posible, el uso como método y como propiedad, más adelante me limitare a usar métodos y tú usaras el que más de agrade.

Hasta ahora, hemos accedido a un solo documento, pero podemos acceder a todos los documentos actualmente abiertos en OpenOffice.org, copia y prueba la siguiente macro.

```
Option Explicit

Sub Controlando_OpenOffice1()
Dim oDocumentos As Object
Dim oDocumento As Object
Dim oEnumeraDocumentos As Object

   'Accedemos a todos los documentos actualmente abiertos
   oDocumentos = StarDesktop.getComponents()
   'Enumeramos cada uno
   oEnumeraDocumentos = oDocumentos.createEnumeration()
   'hasMoreElements devuelve verdadero (True) mientras haya elementos
   Do While oEnumeraDocumentos.hasMoreElements()
         'Nos desplazamos al siguiente elemento y lo asignamos
         oDocumento = oEnumeraDocumentos.nextElement()
         'Mostramos el tipo de documento
         MsgBox "Soy un documento de " & TipoDocumento( oDocumento )
   Loop

End Sub

Function TipoDocumento(ByRef Documento As Object) As String
Dim sAplicacion As String

   If Documento.supportsService("com.sun.star.sheet.SpreadsheetDocument") Then
         sAplicacion = "Calc"
   ElseIf Documento.supportsService("com.sun.star.text.TextDocument") Then
         sAplicacion = "Writer"
   ElseIf Documento.supportsService("com.sun.star.presentation.PresentationDocument") Then
         sAplicacion = "Impress"
   ElseIf Documento.supportsService("com.sun.star.drawing.DrawingDocument") Then
         sAplicacion = "Draw"
   ElseIf Documento.supportsService("com.sun.star.sdb.OfficeDatabaseDocument") Then
         sAplicacion = "Base"
   ElseIf Documento.supportsService("com.sun.star.formula.FormulaProperties") Then
         sAplicacion = "Math"
```

```
    ElseIf Documento.supportsService("com.sun.star.script.BasicIDE") Then
          sAplicacion = "Basic"
    Else
          sAplicacion = "Desconocido"
    End If

    TipoDocumento = sAplicacion

End Function
```

Toma nota de como hemos convertido una macro usada anteriormente en una función que nos devuelve el tipo de documento abierto. Muchos objetos en OpenOffice.org, solo son accesibles creando una “enumeración”, así que la estructura de la macro anterior la podrías usar con frecuencia. Una “enumeración”, es como un listado de objetos, se crean con el método createEnumeration(), el método .hasMoreElements() nos sirve para saber si todavía hay elementos y el método .nextElement(), permite movernos al siguiente elemento de la enumeración.

Empecemos con código un poco más divertido, la siguiente variante de la macro anterior, te cerrara “todos” los archivos de Calc que no tengan cambios por guardar, **CUIDADO**, si ejecutas la siguiente macro desde un archivo de Calc podrías cerrarte el archivo con resultados inesperados, como dicen en mi pueblo -sobre advertencia no hay engaño-.

```
Sub Controlando_OpenOffice2()
Dim oDocumentos As Object
Dim oDocumento As Object
Dim oEnumeraDocumentos As Object
Dim co1 As Integer

   oDocumentos = StarDesktop.getComponents()
   oEnumeraDocumentos = oDocumentos.createEnumeration()
   Do While oEnumeraDocumentos.hasMoreElements()
         oDocumento = oEnumeraDocumentos.nextElement()
         'Verificamos que sea un archivo de Calc
         If TipoDocumento( oDocumento ) = "Calc" Then
               'Si no ha sido modificado o los cambios ya están guardados
               'el método isModified devolverá falso (False)
               If Not oDocumento.isModified() Then
                     'el método dispose cierra el archivo
                     oDocumento.dispose()
                     co1 = co1 + 1
               End if
         End If
   Loop
   MsgBox "Se cerraron un total de " & CStr(co1) & " archivos de Calc"

End Sub
```

Ten mucho cuidado con el método **dispose**, te cerrará el archivo sin preguntarte nada, tenga o no tenga cambios guardados, usalo con conocimiento de causa. Observa como usamos la propiedad **isModified**(), para saber si el documento ha sido modificado.

La Interfaz de Programación de la Aplicación (API por sus siglas en ingles) proporciona, decenas incluso centenas de servicios con métodos y propiedades para controlar y manipular OpenOffice.org, la puedes consultar en linea en la página oficial de OpenOffice.org (por ahora solo en ingles) en: http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html

También puedes descargarla y después consultarla fuera de línea para su acceso inmediata o si no tienes conexión permanente a Internet, en el Apéndice te muestro como hacerlo. Para una consulta óptima de esta documentación, lo ideal es saber de forma anticipada que

servicio, método o propiedad estamos buscando, para ello, todos los objetos cuentan con unas propiedades especiales que nos dan mucha información, veamos cuales son.

5.3.1 Propiedades especiales de depuración

Ya vimos en el tema anterior como saber con el método “**supportsService**” si un objeto soporta un determinado servicio, pero tiene el inconveniente de que tenemos que saber anticipadamente el nombre de dicho servicio, lo cual a veces es un poco complicado. También vimos el uso del método **getSupportedServiceNames()** que nos devuelve una matriz con los nombres de los servicios que implementa el objeto.

Para obtener toda la información de un objeto, este cuenta con tres propiedades que nos informan de todas las propiedades, métodos e **“interfaces”** que implementa dicho objeto, estas propiedades son tres:

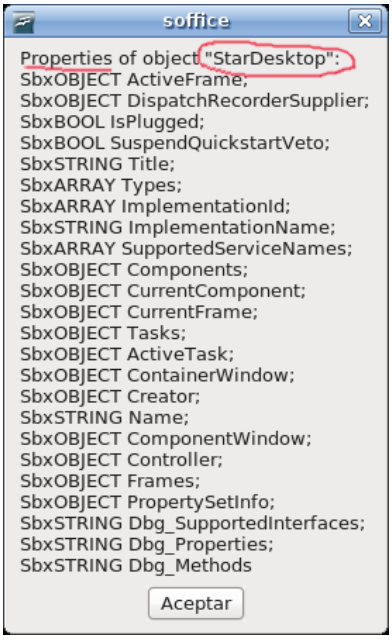
1. Dbg_SupportedInterfaces
2. Dbg_Properties
3. Dbg_Methods

Las tres son de tipo texto (string) y se usan de la siguiente manera.

```
Sub ConsultandoObjetos1()
Dim sDatos As String

   MsgBox StarDesktop.Dbg_Properties
   MsgBox StarDesktop.Dbg_Methods
   MsgBox StarDesktop.Dbg_SupportedInterfaces

End Sub
```

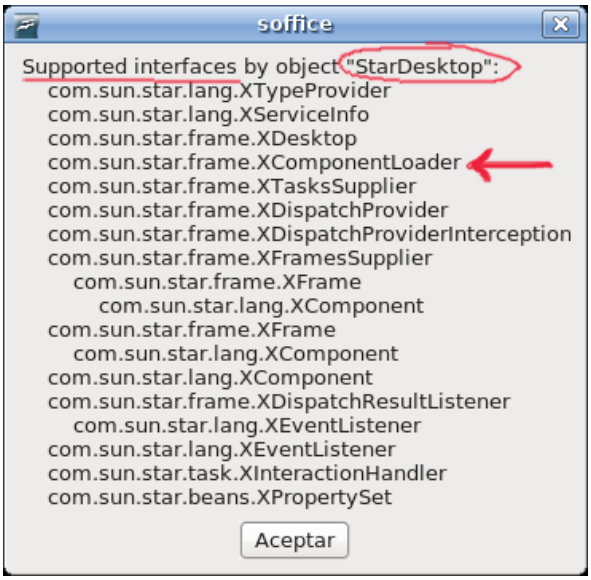
No te me confundas con esto de las **“interfaces”**, para nuestros fines es perfectamente valido que los llamemos “servicios”, simplemente piensa en las interfaces como una forma de organizar servicios, pero en OOo Basic no tiene importancia por que accedemos directamente a las propiedades y métodos de los servicios.

La pequeña macro anterior, te tiene que mostrar un cuadro de mensaje con las propiedades del objeto.



Sus métodos:

Y sus interfaces (servicios):



Observa el servicio señalado con una flecha, si vamos a su documentación, en línea (http://api.openoffice.org/docs/common/ref/com/sun/star/frame/XComponentLoader.html) o local (file:///opt/openoffice/sdk/docs/common/ref/com/sun/star/frame/XComponentLoader.html), te darás cuenta que solo tiene un método **loadComponentFromURL**, pero un método muy importante, pues nos permite crear documentos nuevos o abrir archivos existentes, lo cual veremos al terminar este tema.

Ahora probemos con los datos del documento activo.

```
Sub ConsultandoObjetos2()
Dim sDatos As String

   MsgBox StarDesktop.getCurrentComponent.Dbg_Properties
   MsgBox StarDesktop.getCurrentComponent.Dbg_Methods
   MsgBox StarDesktop.getCurrentComponent.Dbg_SupportedInterfaces

End Sub
```

Esta vez nos muestra muchos más datos, ¿verdad?, observa de que forma tan simple podemos obtener mucha información, en las propiedades, nota que se nos muestra el tipo de valor que puedes obtener de las propiedades y que será el mismo tipo para establecerla si la propiedad es de escritura. Por ejemplo, si ejecutamos la macro anterior desde una hoja de calculo podremos ver entre sus propiedades las siguientes:

1. SbxARRAY Printer
2. SbxSTRING Location
3. SbxOBJECT Sheets

Tenemos una propiedad que devuelve un array (1), otra una cadena (2.- string) y por ultimo un objeto (3.- object).

Ahora, en los métodos también nos muestra información valiosa, por ejemplo y siguiendo en la suposición de que nos informamos de una hoja de calculo, tenemos:

1. SbxSTRING getURL (void)
2. SbxVOID storeAsURL (SbxSTRING, SbxARRAY)
3. SbxINTEGER resetActionLocks (void)

En este caso tenemos el método **getUrl** que devuelve una cadena (string) y no necesita argumentos (1), el método **storeAsURL** requiere una cadena (string) y un array como argumentos (2), pero no devuelve ningún valor y por ultimo el método **resetActionLocks**, devuelve un entero (integer) y no necesita argumentos (3). Por ahora lo importante es saber reconocer la sintaxis de esta información para usarla en nuestras macros.

Por ultimo, cualquiera de los servicios mostrados, muestra la información necesaria para ir a la documentación del API y saber que otros servicios soporta, que métodos y que propiedades.

1. com.sun.star.datatransfer.XTransferable
2. com.sun.star.view.XPrintable
3. com.sun.star.sheet.XSpreadsheetDocument

Creo que con esto queda demostrado que no hay que estar adivinando nada, solo hay que hacer las preguntas y búsquedas correctas.

Dentro de las macros incorporadas en OpenOffice.org, existen varias que nos ayudan a depurar nuestras macros, para el tema que estamos tratando, existe una en especial que te escribe la información de las tres propiedades vistas en un nuevo documento de Writer, la macro se llama **WritedDgInfo** y solo hay que pasarle el objeto del cual queremos obtener su información, veamoslo con un ejemplo:

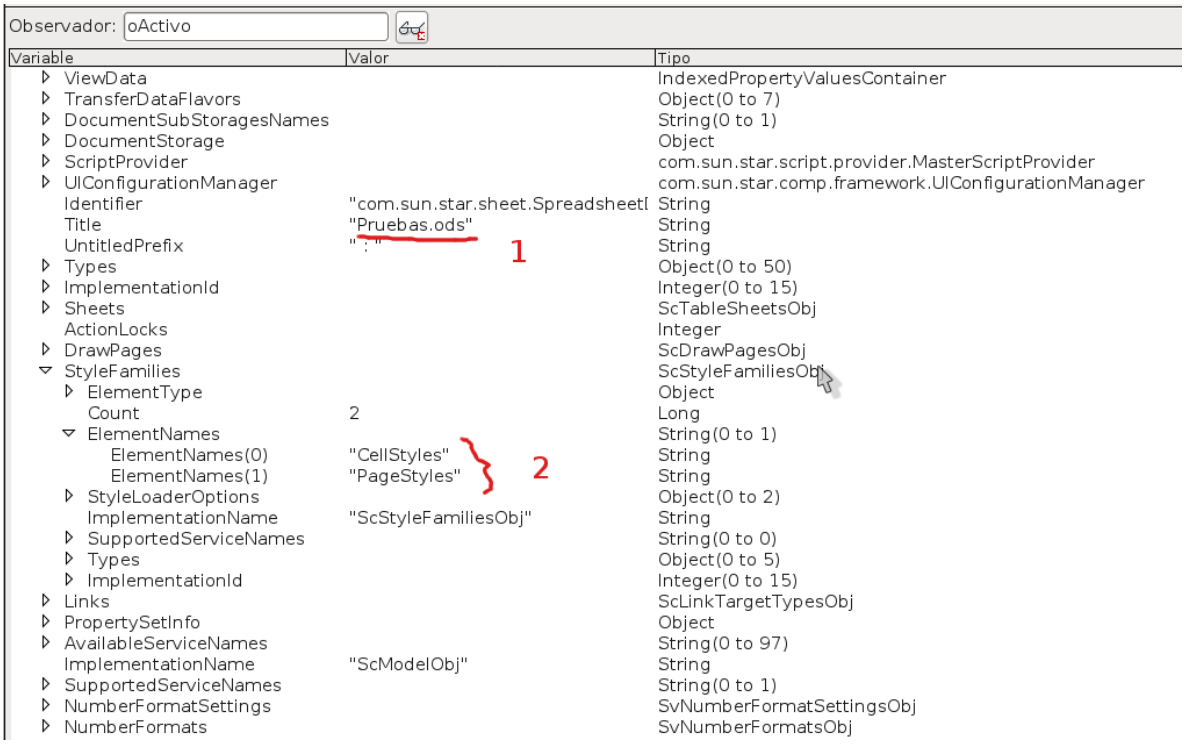
```
Sub ConsultandoObjetos3()

   BasicLibraries.LoadLibrary( "Tools" )
   Call WritedbgInfo( StarDesktop.getCurrentComponent() )

End Sub
```

Prueba la macro anterior, llamándola desde diferentes documentos, aunque la puedes ejecutar siempre y cuando la variable pasada apunte “efectivamente” a una variable de objeto. Con unas pequeñas variantes podemos hacer que en vez de un documento de Writer, nos escriba la información en una hoja de calculo, en el Apéndice te muestro como hacerlo.

Para terminar este tema, veamos como nos puede ayudar la ventana del Observador también con los objetos, copia la siguiente macro, establece un punto de ruptura como te muestro en la imagen siguiente y agrega al observador la variable oActivo.

```
Option Explicit

Sub ConsultandoObjetos4()
Dim oActivo As Object

    oActivo = StarDesktop.getCurrentComponent()

    MsgBox oActivo.Title

End Sub
```

Ejecuta la macro, regresa al IDE y observa los datos de la variable que tenemos en observación:



Por ahora, la mayor parte de esta información no te dirá mucho, pero observa que interesante nos muestra el título del documento (1) y los tipos de estilos (2) que aprenderemos a usar más adelante. Por ahora, solo familiarizate con la forma en que se muestra la información, el tipo de este y el valor mostrado, esto, es de mucha utilidad como podremos comprobarlo más adelante.

5.4 Trabajando con documentos

En donde veremos como manipular documentos de OpenOffice.org, crearlos y guardarlos. El trabajo común a la mayoría de los documentos.

5.4.1 Creando nuevos documentos

Como ya dijimos, el método que nos permite crear nuevos documentos se llama **loadComponentFromURL** y es un método del servicio “com.sun.star.frame.XComponentLoader”, veamos primero los ejemplos y después las explicaciones.

```
Option Explicit

Sub CreandoNuevosDocumentos1()
Dim sRuta As String
Dim mArg()
Dim oNuevoDocumento As Object

   sRuta = "private:factory/scalc"
   oNuevoDocumento = StarDesktop.loadComponentFromURL( sRuta, "_default", 0, mArg() )
```

```
End Sub
```

¿Ya la probaste?, muy fácil ¿no?, creamos una nueva hoja de calculo, y si, lo deduces bien, solo hay que hacer unos pequeños cambios para crear un documento de texto.

```
Sub CreandoNuevosDocumentos2()
Dim sRuta As String
Dim mArg()
Dim oNuevoDocumento As Object

   sRuta = "private:factory/swriter"
   oNuevoDocumento = StarDesktop.loadComponentFromURL( sRuta, "_default", 0, mArg() )

End Sub
```

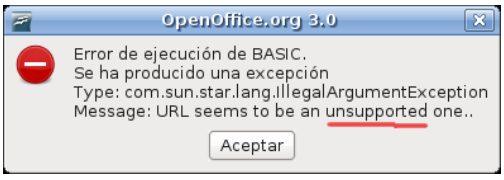
Y los demás tipos de documentos.

```
Sub CreandoNuevosDocumentos3()
Dim sRuta As String
Dim mArg()
Dim oNuevoDocumento As Object

   sRuta = "private:factory/simpress"
   oNuevoDocumento = StarDesktop.loadComponentFromURL( sRuta, "_default", 0, mArg() )

   sRuta = "private:factory/sdraw"
   oNuevoDocumento = StarDesktop.loadComponentFromURL( sRuta, "_default", 0, mArg() )

   sRuta = "private:factory/smath"
   oNuevoDocumento = StarDesktop.loadComponentFromURL( sRuta, "_default", 0, mArg() )

End Sub
```

¿Y una base de datos?, probemos.

```
Sub CreandoNuevosDocumentos4()
Dim sRuta As String
Dim mArg()
Dim oNuevoDocumento As Object

   sRuta = "private:factory/sbase"
   oNuevoDocumento = StarDesktop.loadComponentFromURL( sRuta, "_default", 0, mArg() )

End Sub
```

Nos da un error ¿verdad?



No esta soportado. Para saber por que nos da un error, intenta abrir una base de datos nueva, observa como primero, si queremos trabajar con ella, el asistente nos “obliga” primero a guardarla con un nombre y después ya podemos manipularla, esto, solo pasa con las bases de datos, de los demás documentos podemos crear tantos nuevos documentos como

queramos sin tener que guardar nada, entonces ¿como creamos una nueva base de datos?, la respuesta es sencilla, del mismo modo que lo hacemos desde la interfaz de usuario, es decir, tenemos que guardar primero la base de datos.

```
Option Explicit

Sub CreandoBaseDeDatos1()
Dim oBDServicio As Object
Dim oBDNueva As Object
Dim sRuta As String
Dim mArg()

   'Ruta y nombre donde quedara guardada la base de datos
   sRuta = "/home/mau/Mi_Base_Datos.odb"
   'Creamos el servicio que nos permite manipular bases de datos
   oBDServicio = createUnoService( "com.sun.star.sdb.DatabaseContext" )
   'Creamos una nueva instancia de una base de datos
   oBDNueva = oBDServicio.createInstance()
   'Establecemos el tipo de base
   oBDNueva.URL = "sdbc:embedded:hsqldb"
   'Y la guardamos
   oBDNueva.DatabaseDocument.storeAsURL( sRuta, mArg() )

End Sub
```

Toma en cuenta dos cosas muy importantes en la macro anterior, el nombre que uses, si ya existe, simplemente lo reemplazara, no te preguntara nada, así que ten cuidado de no usar una ruta o nombre de una base importante que tengas y segundo, la base de datos así creada, no quedara “registrada” en OpenOffice.org, con lo que no tendrás acceso desde otros programas de la aplicación, más adelante veremos como registrar una base de datos por código.

5.4.2 Rutas de archivos y directorios

OpenOffice.org, al ser multiplataforma, hace uso de las rutas de archivos y directorios, en el formato URL, por ejemplo.

file:///home/usuario/datos/miarchivo.ods
file:///d:/datos/miarchivo.ods

Toma en cuenta que este formato hace uso de los caracteres 0-9, a-z y A-Z, y cualquier otro carácter será convertido y mostrado con su respectivo código, por ejemplo, los espacios los reemplazara con %20. Siempre que le pases una ruta de archivo a cualquier servicio de OpenOffice.org, procura, casi como una regla, pasarle la ruta en formato Url, para ello, OOo Basic cuenta con dos funciones muy útiles que hacen el trabajo ConvertToUrl y ConvertFromUrl, veamos como se usan.

```
Sub RutasDeArchivos1()
Dim sRuta As String

   'Establecemos una ruta
   sRuta = "/home/mau/Mi archivo de Calc.ods"
   'Y la mostramos
   MsgBox sRuta

   'La convertimos al formato URL
   sRuta = ConvertToUrl( "/home/mau/Mi archivo de Calc.ods" )
   'Y vemos el resultado
   MsgBox sRuta
```

```
    'Regresamos al formato local
    sRuta = ConvertFromUrl( sRuta )
    'Y comprobamos que es correcto
    MsgBox sRuta

End Sub
```

Abre varios documentos existentes y ejecuta la siguiente macro desde cada uno:

```
Sub DocumentoActivoMostrarDatos()
Dim sRutaURL As String
Dim sRutaCompleta As String
Dim sRuta As String
Dim sNombre As String

   GlobalScope.BasicLibraries.LoadLibrary( "Tools" )
   'Referencia al documento activo
   sRutaURL = ThisComponent.getURL()
   'Convertimos la ruta URL en formato local
   sRutaCompleta = ConvertFromUrl( sRutaURL )
   'Obtenemos solo el nombre del archivo
   sNombre = FileNameOutOfPath( sRutaURL )
   'Obtenemos el directorio donde esta el archivo
   sRuta = DirectoryNameoutofPath(sRutaCompleta, GetPathSeparator())
   'Mostramos los resultados
   MsgBox sRutaCompleta & Chr(13) & Chr(13) & sNombre & Chr(13) & Chr(13) & sRuta

End Sub
```

El objeto **ThisComponent**, es una palabra clave especial que designa al documento desde el cual llamamos a nuestra macro, es buena practica de programación, comprobar siempre a que tipo de documento esta apuntando ThisComponent, no es lo mismo si ejecutas una macro desde la interfaz del usuario, a que lo hagas desde el IDE, y aun desde el IDE, dependerá si haces referencia a el desde un archivo cualquiera o desde el archivo especial Mis Macros, así que siempre verifica que el objeto apuntado con ThisComponent, sea el que esperas. Observa que hacemos uso de algunas funciones incorporadas de OpenOffice.org, estas son: FileNameOutOfPath, que te devuelve de la ruta pasada solo el nombre del archivo y DirectoryNameoutofPath, que te regresa solo el directorio de la ruta del archivo pasada. Estas y otras útiles funciones se encuentran dentro de la biblioteca Tools de OpenOffice.org que cargamos en la primer línea de nuestra macro de ejemplo.

En nuestro siguiente tema, veremos la importancia de usar las rutas de archivos y directorios en formato Url y recuerda que en S.O. Linux, se tiene en cuenta la diferencia entre mayúsculas y minúsculas.

5.4.3 Abriendo, guardando y cerrando documentos

Abrir archivos existentes es sumamente sencillo, usamos el mismo método usado para crear nuevos documentos, pero en el parámetro ruta, le especificamos la ubicación del archivo a abrir de la siguiente manera.

```
Sub AbriendoDocumentos1()
Dim sRuta As String
Dim mArg()
```

```
Dim oDocumento As Object

   'Reemplaza esta ruta por la ruta de tú archivo
   sRuta = ConvertToUrl( "/home/mau/Mi archivo de Calc.ods" )
   oDocumento = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mArg() )

End Sub
```

Si tu archivo existe, el código anterior lo abrirá, “siempre” utiliza ConvertToUrl cuando uses rutas de archivos o directorios en funciones de OpenOffice.org que las requieran

Si abres una plantilla, esta se comportara de forma normal, es decir, te abrirá una copia de dicha plantilla.

```
   sRuta = ConvertToUrl( "/home/mau/Mi Plantilla.ots" )
   oDocumento = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mArg() )
```

Pero también podrías querer abrir la plantilla para editarla, en este caso, le indicamos en las opciones de apertura esto, observa como se declara la variable mOpciones.

```
Sub AbriendoDocumentos3()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   mOpciones(0).Name = "AsTemplate"
   mOpciones(0).Value = False

   sRuta = ConvertToUrl( "/home/mau/miPlantilla.ots" )
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

End Sub
```

La estructura "com.sun.star.beans.PropertyValue" esta conformada por un par de valores, un nombre y un valor y es muy frecuentemente usada cuando se programa en OOo Basic, observa como la matriz de la variable mOpciones la hemos inicializado en 0, es decir, solo contendrá un par de valores, pero muy bien puede tener más opciones como veremos más adelante. En la macro anterior le estamos indicando la propiedad AsTemplate (que seria algo así como Es Plantilla), si el valor es falso (False), le estamos indicando que queremos abrir la plantilla para editarla, en caso contrario (True) abrirá una copia de la plantilla. Lo interesante de esta propiedad es que perfectamente la puedes aplicar a archivos existentes, es decir, archivos que “no” necesariamente sean plantillas, con esta propiedad podemos forzarlos a comportarse como plantillas, por ejemplo, prueba el siguiente código con algún archivo existente que no sea plantilla.

```
Sub AbriendoDocumentos4()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   mOpciones(0).Name = "AsTemplate"
   mOpciones(0).Value = True

   sRuta = ConvertToUrl( "/home/mau/Mi archivo de Writer.odt" )
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

End Sub
```

Observa como el archivo es un documento de texto normal (ODT), pero con la propiedad AsTemplate, forzamos su apertura como una plantilla. Otras propiedades interesantes son ReadOnly (Solo Lectura) para abrir un archivo como solo lectura y Password (Contraseña) para abrir archivos con contraseña.

```
Sub AbriendoDocumentos5()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   mOpciones(0).Name = "ReadOnly"
   mOpciones(0).Value = True

   sRuta = ConvertToUrl( "/home/mau/Mi archivo de Writer.odt" )
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

End Sub


Sub AbriendoDocumentos6()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   'La contraseña más segura
   mOpciones(0).Name = "Password"
   mOpciones(0).Value = "letmein"

   sRuta = ConvertToUrl( "/home/mau/Mi archivo de Draw.odg" )
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

End Sub
```

Por supuesto puedes combinar cualquier cantidad de opciones siempre y cuando no entren en conflicto.

```
Sub AbriendoDocumentos7()
Dim sRuta As String
Dim mOpciones(1) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   mOpciones(0).Name = "ReadOnly"
   mOpciones(0).Value = True
   mOpciones(1).Name = "Password"
   mOpciones(1).Value = "abrete"

   sRuta = ConvertToUrl( "/home/mau/Mi archivo de Draw.odg" )
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

End Sub
```

Prueba a poner la contraseña incorrecta y observa el mensaje, por supuesto, podemos solicitarle al usuario la contraseña y no esta demás verificar que el archivo exista.

```
Sub AbriendoDocumentos8()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object
Dim sContra As String

   sContra = Trim( InputBox( "Introduce la contraseña de apertura" ) )
```

```
    mOpciones(0).Name = "Password"
    mOpciones(0).Value = sContra

    sRuta = ConvertToUrl( "/home/mau/Mi archivo de Draw.odg" )

          'Verificamos que exista el archivo
    If Len(Dir(sRuta)) > 0 Then
          oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )
    Else
          MsgBox "El archivo no existe"
    End If

  End Sub
```

Mientras un formato de archivo sea soportado por OpenOffice.org, lo podrás abrir con el método loadComponentFromURL, en el siguiente ejemplo, abrimos un archivo tipo Doc, un Html y un Xls.

```
  Sub AbriendoDocumentos9()
  Dim sRuta As String
  Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
  Dim oDoc As Object

     sRuta = ConvertToUrl( "/home/mau/Campamentos.doc" )
     oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

     sRuta = ConvertToUrl( "/home/mau/Renacimiento.htm" )
     oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

     sRuta = ConvertToUrl( "/home/mau/datos.xls" )
     oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

  End Sub
```

Cuando abres un archivo por código y este contiene macros, tu puedes establecer si se abre con las macros activadas, de forma predeterminada lo abre con las macros desactivadas, así que solo nos resta saber como abrirlo con las macros activadas.

```
  Sub AbriendoDocumentos10()
  Dim sRuta As String
  Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
  Dim oDoc As Object

     mOpciones(0).Name = "MacroExecutionMode"
     mOpciones(0).Value = 4

     sRuta = ConvertToUrl( "/home/mau/Mi archivo de Calc.ods" )
     oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

  End Sub
```

Otra opción interesante, es la que nos permite abrir una presentación e iniciarla inmediatamente.

```
  Sub AbriendoDocumentos11()
  Dim sRuta As String
  Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
  Dim oDoc As Object

     mOpciones(0).Name = "StartPresentation"
```

```
    mOpciones(0).Value = True

    sRuta = ConvertToUrl( "/home/mau/Mi archivo de Impress.odp" )
    oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

End Sub
```

Puedes abrir un archivo como vista previa, útil para evitar que el usuario haga cambios en el, en este estado, los cambios que intente hacer el usuario no se guardaran.

```
Sub AbriendoDocumentos12()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   mOpciones(0).Name = "Preview"
   mOpciones(0).Value = True

   sRuta = ConvertToUrl( "/home/mau/Mi archivo de Writer.odt" )
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

End Sub
```

Por supuesto, la idea general de abrir un archivo es para obtener datos de el o para manipularlo, ya sea un documento nuevo o uno existente, veamos como guardar los cambios realizados y cerrar el archivo.

```
Sub AbriendoGuardandoDocumentos1()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   sRuta = ConvertToUrl( "/home/mau/Mi archivo de Calc.ods" )
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

   'AQUI va todo el código que quieras para manipular el archivo

   MsgBox "Archivo abierto y modificado correctamente, presione Aceptar para guardar y cerrar"

   'Guardamos los cambios
   oDoc.store()
   'Cerramos el archivo
   oDoc.close(True)

End Sub
```

El código anterior funcionara con un archivo existente, pero con uno nuevo no, te dará un error, por supuesto no me creas, compruébalo. Los documentos tienen una propiedad (**hasLocation**) que nos informan si un documento ya esta guardado o no.

```
Sub AbriendoGuardandoDocumentos2()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   sRuta = "private:factory/scalc"
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

   'Nos indica si el archivo esta guardado físicamente
   MsgBox oDoc.hasLocation()
```

```
    sRuta = ConvertToUrl( "/home/mau/Mi archivo de Calc.ods" )
    oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

    MsgBox oDoc.hasLocation()

End Sub
```

El primer mensaje debe mostrarte Falso (False) porque es un documento nuevo y el segundo Verdadero (True) por ser un archivo abierto del disco.

También tenemos una propiedad que nos ayuda a saber si un documento ha sido modificado o no, ejecuta la siguiente macro, llamándola desde diferentes documentos, unos ábrelos y modificalos antes de ejecutarla, otros, solo ábrelos y ejecutala para que notes la diferencia.

```
Sub EstaModificado()

    MsgBox ThisComponent.isModified()

End Sub
```

Y otra más que nos informa si el documento es de solo lectura.

```
Sub AbriendoGuardandoDocumentos3()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

    sRuta = "private:factory/scalc"
    oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

    MsgBox oDoc.isReadOnly()

    mOpciones(0).Name = "ReadOnly"
    mOpciones(0).Value = True

    sRuta = ConvertToUrl( "/home/mau/Mi archivo de Calc.ods" )
    oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

    MsgBox oDoc.isReadOnly()

End Sub
```

Ya te imaginaras que útil es saber, antes de guardar un archivo, si este es de solo lectura, si ha sido modificado o si ya ha sido guardado, antes de ver trabajando juntas todas estas propiedades, veamos como guardar un archivo nuevo con el método storeAsUrl.

```
Sub AbriendoGuardandoDocumentos4()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

    sRuta = "private:factory/scalc"
    oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

    'Establecemos la ruta y nombre donde se guardara el archivo
    sRuta = ConvertToUrl( "/home/mau/Archivo de Calc.ods" )
    'Guardamos el archivo
    oDoc.storeAsURL( sRuta, mOpciones() )
```

```
End Sub
```

Observa como requiere dos parámetros, la ruta del archivo y una matriz de propiedades, algunas de las cuales veremos a continuación. Toma nota de que si el archivo ya existe lo reemplazara sin avisarte por lo que es buena practica de programación verificar esto.

```
Sub AbriendoGuardandoDocumentos5()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   sRuta = "private:factory/scalc"
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

   sRuta = ConvertToUrl( "/home/mau/Archivo de Calc.ods" )
   If Len( Dir( sRuta )) = 0 Then
         'Guardamos el archivo
         oDoc.storeAsURL( sRuta, mOpciones() )
   Else
         MsgBox "El archivo ya existe, no se ha guardado el nuevo archivo"
   End If

End Sub
```

Te queda de tarea implementar la pregunta al usuario para reemplazar o no el archivo. Veamos algunas propiedades interesantes para guardar archivos, por ejemplo, guardar con contraseña.

```
Sub AbriendoGuardandoDocumentos6()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   sRuta = "private:factory/scalc"
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

   'Como vez no tengo mucha imaginación
   mOpciones(0).Name = "Password"
   mOpciones(0).Value = "abrete"
   sRuta = ConvertToUrl( "/home/mau/Archivo de Calc.ods" )
   oDoc.storeAsURL( sRuta, mOpciones() )

End Sub
```

Como sabes, OpenOffice.org, soporta la importación/exportación de múltiples formatos de archivos, veamos como guardar, por ejemplo, en formato DOC.

```
Sub AbriendoGuardandoDocumentos7()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   sRuta = "private:factory/swriter"
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

   'Establecemos el filtro a documentos tipo DOC
   mOpciones(0).Name = "FilterName"
   mOpciones(0).Value = "MS Word 97"
   sRuta = ConvertToUrl( "/home/mau/Archivo de Word.doc" )
```

```
   oDoc.storeAsURL( sRuta, mOpciones() )

End Sub
```

El método storeAsURL también lo puedes usar para abrir un archivo existente y guardarlo con otro nombre.

```
Sub AbriendoGuardandoDocumentos8()
Dim sRuta As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
Dim oDoc As Object

   sRuta = ConvertToUrl( "/home/mau/Mi archivo de Calc.ods" )
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )

   sRuta = ConvertToUrl( "/home/mau/Nuevo archivo Calc.ods" )
   oDoc.storeAsURL( sRuta, mOpciones() )

End Sub
```

Para terminar esta capitulo, vamos a hacer un poco más interactivo la apertura y guardado de archivos, permitamos que el usuario escoja la ruta y el archivo desde un cuadro de dialogo, para ello, usaremos el servicio com.sun.star.ui.dialogs.FilePicker, que nos permite hacer esto, veamos como.

```
Sub AbrirArchivo1()
Dim oDlgAbrirArchivo as Object
Dim mArchivo() As String
Dim mOpciones()
Dim sRuta As String
Dim oDoc As Object

   'Creamos el servicio necesario
   oDlgAbrirArchivo = CreateUnoService ("com.sun.star.ui.dialogs.FilePicker")
   'Establecemos el titulo del cuadro de dialogo
   oDlgAbrirArchivo.setTitle("Selecciona el archivo a abrir")
   'Con el método .Execute() mostramos el cuadro de dialogo
   'Si el usuario presiona Abrir el método devuelve 1 que podemos evaluar como Verdadero (True)
   'Si presiona Cancelar devuelve 0
   If oDlgAbrirArchivo.Execute() Then
         'De forma predeterminada, solo se puede seleccionar un archivo
         'pero devuelve una matriz de todos modos con la ruta completa
         'del archivo en formato URL
         mArchivo() = oDlgAbrirArchivo.getFiles()
         'El primer elemento de la matriz es el archivo seleccionado
         sRuta = mArchivo(0)
         'Y lo abrimos
         oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )
   Else
         'Si el usuario presiona Cancelar
         MsgBox "Proceso cancelado"
   End If

End Sub
```

Si quieres abrir más de un archivo a la vez, usa.

```
Sub AbrirArchivo2()
Dim oDlgAbrirArchivo as Object
Dim mArchivos() As String
```

```
Dim mOpciones()
Dim co1 As Integer

   oDlgAbrirArchivo = CreateUnoService ("com.sun.star.ui.dialogs.FilePicker")
   oDlgAbrirArchivo.setTitle("Selecciona los archivos a abrir")
   'Establecemos que se puedan seleccionar mas de un archivo
   oDlgAbrirArchivo.setMultiSelectionMode(True)

   If oDlgAbrirArchivo.Execute() Then
         mArchivos() = oDlgAbrirArchivo.getSelectedFiles()
         'Mostramos los archivos seleccionados
         For co1 = LBound( mArchivos() ) To UBound( mArchivos() )
               MsgBox ConvertFromUrl( mArchivos(co1) )
         Next
               'Te queda de tarea abrir todos los archivos
               'oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mOpciones() )
   Else
         MsgBox "Proceso cancelado"
   End If

End Sub
```

Ahora, mostramos el cuadro de dialogo Guardar Como, para permitir al usuario seleccionar carpeta y nombre de archivo donde guardar su archivo.

```
Sub GuardarArchivo1()
Dim oDlgGuardarArchivo as Object
Dim mDlgOpciones()
Dim mArchivo() As String
Dim mOpciones()

   'Usamos el mismo servicio para abrir archivos, pues es el mismo cuadro de dialogo
   oDlgGuardarArchivo = CreateUnoService ("com.sun.star.ui.dialogs.FilePicker")
   'Establecemos que nos muestre el cuadro de dialogo Guardar Como con las
   'casillas de verificación Autoextension y guardar con contraseña
   'com.sun.star.ui.dialogs.TemplateDescription.FILESAVE_AUTOEXTENSION_PASSWORD = 2
   mDlgOpciones() = Array(2)
   With oDlgGuardarArchivo
         'Iniciamos el cuadro de dialogo con las opciones seleccionadas
         .Initialize ( mDlgOpciones() )
         'Agregamos filtros de archivos
         .AppendFilter( "All files (*.*)", "*.*" )
         .AppendFilter( "Documento de Texto ODF (.odt)", "*.odt" )
         .AppendFilter( "Hoja de calculo ODF (.ods)", "*.ods" )
   End With

   'Lo mostramos
   If oDlgGuardarArchivo.Execute() Then
         mArchivo() = oDlgGuardarArchivo.getFiles()
         'Solo te muestro la ruta y nombre de archivo seleccionado
         'Solo te resta guardar el archivo
         MsgBox ConvertFromUrl( mArchivo(0) )
   Else
         MsgBox "Proceso cancelado"
   End If

End Sub
```

Los cuadros de diálogo que veas, estarán en función de la opción del menú Herramientas | Opciones... rama OpenOffice.org, subrama General, sección Abre/Guarda diálogos, casilla de verificación Usa los diálogos OpenOffice.org.

5.4.4 Exportando a PDF

Exportar a PDF desde OpenOffice.org es muy sencillo, lo es también con OOo Basic, prueba la siguiente macro desde diferentes documentos, por ahora, asegurate de que los archivos ya estén guardados antes de exportar, te queda de tarea intentar adaptarla para archivos nuevos.

```
Sub ExportarPDF()
Dim oDoc As Object
Dim sTipoDoc As String
Dim mOpciones(0) As New "com.sun.star.beans.PropertyValue"
dim sRuta As string

   BasicLibraries.LoadLibrary( "Tools" )
   'Referencia al documento desde donde se llama la macro
   oDoc = ThisComponent

   'Obtenemos el tipo de documento, esta función ya la usamos
   sTipoDoc = TipoDocumento( oDoc )

   'Solo se pueden exportar (por ahora)
   Select Case sTipoDoc
         'los siguiente documentos
         Case "Calc", "Writer", "Draw", "Impress", "Math"
               'Establecemos el tipo de filtro
               mOpciones(0).Name = "FilterName"
               'Construimos el filtro correcto PDF para cada aplicacion
               mOpciones(0).Value = LCase(sTipoDoc) & "_pdf_Export"

               'Construimos la ruta correcta, usamos el mismo directorio
               'y nombre del archivo, solo agregamos la extensión PDF
               sRuta = GetFileNameWithoutExtension( oDoc.getUrl ) & ".pdf"

               'Guardamos el archivo
               oDoc.storeToURL( sRuta, mOpciones() )
         Case Else
               MsgBox "Aplicación no soportada"
   End Select

End Sub
```

La función GetFileNameWithoutExtension viene integrada en las macros de OpenOffice.org, de la ruta pasada, te devuelve toda la ruta pero sin la extensión del archivo, observa como solo le agregamos la extensión PDF, también observa que esta vez “no” hemos usado storeAsUrl, sino storeToUrl para exportar el archivo.

Te habrás dado cuenta de que las variantes pueden ser inmensas, pero creo que tenemos los elementos para empezar a automatizar nuestras tareas diarias, en los siguientes capítulos, ahondaremos en las características especificas de cada tipo de documento.

5.4.5 Tareas comunes en documentos

Veamos algunas tareas comunes a la mayoría de los documentos de OOo, por ejemplo, para saber el titulo del documento activo, usamos.

```
Sub ControlarAplicacion1()
```

```
Dim oDoc As Object
Dim oControlador As Object

   'El documento desde donde se llama esta macro
   oDoc = ThisComponent
   'El controlador del documento actual
   oControlador = oDoc.getCurrentController
   'El titulo
   MsgBox oControlador.getTitle

End Sub
```

Para activar el documento que quieras, se usa el método *setFocus* de la siguiente manera, en donde activamos durante un segundo y medio cada documento de OOo abierto.

```
Sub ControlarAplicacion2()
Dim oDocumentos As Object
Dim oDocumento As Object
Dim oVentana As Object
Dim oEnumeraDocumentos As Object

   'Accedemos a todos los documentos actualmente abiertos
   oDocumentos = StarDesktop.getComponents()
   'Enumeramos cada uno
   oEnumeraDocumentos = oDocumentos.createEnumeration()
   'hasMoreElements devuelve verdadero (True) mientras haya elementos
   Do While oEnumeraDocumentos.hasMoreElements()
         'Nos desplazamos al siguiente elemento y lo asignamos
         oDocumento = oEnumeraDocumentos.nextElement()
         'Obtenemos acceso a la ventana
         oVentana = oDocumento.getCurrentController.getFrame.getComponentWindow()
         'Le enviamos el foco para activarla
         oVentana.setFocus()
         'Esperamos 1.5 segundos para pasar al siguiente documento
         Wait 1500
   Loop

End Sub
```

Una posibilidad interesante, es la de controlar la barra de estado de las aplicaciones, muy útil para mostrar mensajes al usuario o mostrar una barra de progreso, como en el siguiente ejemplo.

```
Sub ControlarAplicacion3()
Dim oBarraEstado As Object
Dim co1 As Integer

   'Referencia a la barra de estado del documento activo
   oBarraEstado = ThisComponent.getCurrentController.StatusIndicator
   'Establecemos el texto inicial y el limite de la barra de progreso
   oBarraEstado.start( "Contando ", 10 )
   For co1 = 1 To 10
         'Establecemos el valor de la barra de progreso
         oBarraEstado.setValue( co1 )
         'Esperamos un segundo
         Wait 1000
   Next
   'Es importante finalizar la barra de estado para devolverle el control a la aplicación
   oBarraEstado.end()

End Sub
```

La macro anterior, funcionará en todas las aplicaciones, excepto en Base, por lo que tienes que validar en su caso si es necesario, también puedes cambiar el texto durante el progreso para dar más información al usuario, como en.

```
Sub ControlarAplicacion4()
Dim oBarraEstado As Object
Dim co1 As Integer

   'Referencia a la barra de estado del documento activo
   oBarraEstado = ThisComponent.getCurrentController.StatusIndicator
   'Establecemos el texto inicial y el limite de la barra de progreso
   oBarraEstado.start( "Procesando Líneas ", 10 )
   For co1 = 1 To 10
         'Establecemos el valor de la barra de progreso
         oBarraEstado.setValue( co1 )
         'Y el texto
         oBarraEstado.setText( "Procesando la línea: " & co1 )
         'Esperamos un segundo
         Wait 1000
   Next
   'Es importante finalizar la barra de estado para devolverle el control a la aplicación
   oBarraEstado.end()

End Sub
```

Cuando hagas uso de la barra de estado para mostrar el progreso de una tarea, es recomendable hacerlo en ciclos determinados, para saber siempre donde terminará, esto te permite mostrar una avance “real” de tu proceso.

Otra opción es cambiar el zoom de la vista del documento.

```
Sub ControlarAplicacion5()
Dim oDoc As Object
Dim oDH As Object
dim mOpc(0) As New com.sun.star.beans.PropertyValue

   'Acceso al marco del documento
   oDoc = ThisComponent.CurrentController.Frame
   'El despachador de instrucciones a nivel usuario
   oDH = createUnoService("com.sun.star.frame.DispatchHelper")

   'El método que deseamos ejecutar
   mOpc(0).Name = "ZoomSlider.CurrentZoom"
   'El valor deseado, en este caso 150%
   mOpc(0).Value = 150
   'Ejecutamos la orden
   oDH.executeDispatch(oDoc, ".uno:ZoomSlider", "", 0, mOpc() )

End Sub
```

La opción anterior, solo funcionará en las aplicaciones que soporten zoom; Writer, Calc, Impress y Draw, Base y Math no lo implementan, pero no te dará ningún error si llamas a la macro desde estas aplicaciones, simplemente ignorará la instrucción.

6 Trabajando con hojas de calculo – Calc

En el capitulo anterior aprendiste a crear, abrir y guardar archivos de Calc y otros tipos de documentos, en este, aprenderemos más detalles del manejo de la hoja de calculo con código OOo Basic que se basan en el servicio: com.sun.star.sheet.SpreadsheetDocument, doy por sentado que eres un usuario medio de hoja de calculo, pues si no sabes que es un formato condicional, por citar solo un ejemplo, te será más complicado establecerlo por código, como aprenderemos en este capítulo, así que te invito a repasar tus apuntes de hoja de calculo.

6.1 Trabajando con hojas

Para acceder a todas las hojas de un documento de hoja de calculo, usamos.

```
Sub TodasLasHojas1()
Dim oDoc As Object
Dim oHojas As Object

   'Acceso al documento desde donde se llama a esta macro
   oDoc = ThisComponent
   'Nos aseguramos de que sea una hoja de calculo
   If oDoc.supportsService("com.sun.star.sheet.SpreadsheetDocument") Then
         'Referencia a TODAS las hojas del documento
         oHojas = oDoc.getSheets()
         'Mostramos cuantas son
         MsgBox oHojas.getCount()
   Else
         MsgBox "No es un documento de hoja de calculo"
   End If

End Sub
```

Asegurate de llamar a la macro anterior desde una hoja de calculo, si estas usando el archivo especial Mis Macros, te recomiendo siempre verificar que ThisComponent, apunta efectivamente a un documento de hoja de calculo como se vio más arriba, de ahora en adelante, daré por hecho que lo sabes y tú determinaras si haces la validación o no.

Podemos acceder a los nombres de todas las hojas.

```
Sub TodasLasHojas2()
Dim oDoc As Object
Dim oHojas As Object
Dim mNombresHojas() As String
Dim sMensaje As String

   GlobalScope.BasicLibraries.LoadLibrary( "Tools" )
   oDoc = ThisComponent
   oHojas = oDoc.getSheets()

   'Obtenemos una matriz con los nombres de todas las hojas
   mNombresHojas() = oHojas.getElementNames()

   'Construimos el mensaje
   sMensaje = "El archivo " & FileNameOutOfPath( oDoc.getLocation() ) & _
```

```
            Chr(13) & "tiene las siguientes hojas" & Chr(13) & Chr(13)
    sMensaje = sMensaje & Join( mNombresHojas(), Chr(13) )

    'Lo mostramos
    MsgBox sMensaje

End Sub
```

La función FileNameOutOfPath, viene incorporada en OpenOffice.org y nos devuelve solo el nombre del archivo de la ruta pasada.

También podemos mostrar los nombres de las hojas una a una.

```
Sub TodasLasHojas3()
Dim oDoc As Object
Dim oHojas As Object
Dim mNombresHojas() As String
Dim co1 As Integer

   oDoc = ThisComponent
   oHojas = oDoc.getSheets()

   'Obtenemos una matriz con los nombres de todas las hojas
   mNombresHojas() = oHojas.getElementNames()

   For co1 = LBound( mNombresHojas() ) To UBound( mNombresHojas() )
        MsgBox mNombresHojas(co1)
   Next

End Sub
```

Podemos devolver solo la hoja que nos interesa por su nombre.

```
Sub UnaHoja1()
Dim oDoc As Object
Dim oHoja As Object

   oDoc = ThisComponent
   'Accedemos a una hoja por su nombre
   oHoja = oDoc.getSheets.getByName("Datos Agosto")

   'Solo comprobamos que es la hoja correcta
   MsgBox oHoja.getName()

End Sub
```

Pero si la hoja no existe, la macro anterior te dará un error, el método hasByName es muy útil para saber si una hoja existe, lo cual es indispensable para acceder a ella.

```
Sub UnaHoja2()
Dim oDoc As Object
Dim oHojas As Object
Dim oHoja As Object
Dim sNombreHoja As String

   oDoc = ThisComponent
   oHojas = oDoc.getSheets()

   sNombreHoja = "Datos Agosto"
```

```
    'Comprobamos que la hoja exista para poder acceder a ella
    If oHojas.hasByName( sNombreHoja ) Then
          'Accedemos a una hoja por su nombre
          oHoja = oHojas.getByName( sNombreHoja )
          MsgBox oHoja.getName() & " - existe en el documento"
    Else
          MsgBox "La hoja -" & sNombreHoja & "- no existe"
    End If

End Sub
```

Nota que el método hasByName es un método del conjunto de las hojas (getSheets) y te devolverá verdadero (True) en caso de que la hoja exista y falso (False) en caso de que no. Este método no distingue entre mayúsculas y minúsculas.

Podemos acceder a una hoja por su índice, recuerda que los índices en OpenOffice.org empiezan en cero, en las hojas, la numeración empieza de izquierda a derecha.

```
Sub UnaHoja3()
Dim oDoc As Object
Dim oHojas As Object
Dim oHoja As Object

   oDoc = ThisComponent
   oHojas = oDoc.getSheets()

   'Accedemos a la hoja por el indice
   oHoja = oHojas.getByIndex( 1 )
   MsgBox oHoja.getName()

End Sub
```

Del mismo modo que por el nombre, si tratas de acceder a una hoja por un índice que no exista, te dará un error, lo podemos comprobar asegurándonos que el numero de índice a consultar siempre es menor al total de las hojas.

```
Sub UnaHoja4()
Dim oDoc As Object
Dim oHojas As Object
Dim oHoja As Object
Dim iNumeroHoja As Integer

   oDoc = ThisComponent
   oHojas = oDoc.getSheets()
   iNumeroHoja = 1

   'Comprobamos que la hoja exista
   If iNumeroHoja < oHojas.getCount() Then
         'Accedemos a la hoja por el índice
         oHoja = oHojas.getByIndex( iNumeroHoja )
         MsgBox oHoja.getName()
   Else
         MsgBox "Numero de hoja no existe"
   End If

End Sub
```

Por lo que podemos acceder a cada hoja de un documento, también por su índice.

```
Sub TodasLasHojas4()
Dim oDoc As Object
Dim oHojas As Object
Dim oHoja As Object
Dim co1 As Integer

   oDoc = ThisComponent
   oHojas = oDoc.getSheets()

   'Nota que el limite, es el total de hojas menos uno, por que comienza en 0
   For co1 = 0 To oHojas.getCount()-1
         oHoja = oHojas.getByIndex( co1 )
         MsgBox oHoja.getName()
   Next

End Sub
```

Toma en cuenta que si mueves una hoja de posición en relación con las demás, su índice cambiara, no así su nombre, pero el nombre es susceptible de ser cambiado por el usuario, así que siempre comprueba que exista una hoja antes de intentar acceder a ella.

Otra opción es devolver la hoja activa.

```
Public Sub HojaActiva()
Dim oHoja As Object

   'Hacemos una referencia a la hoja activa
   oHoja = ThisComponent.getCurrentController.getActiveSheet()
   Msgbox oHoja.getName()

End Sub
```

Ahora, ya puedes crearte tus útiles funciones para trabajar con hojas, por ejemplo, una función que nos devuelva falso o verdadero según exista o no el nombre de la hoja pasado como argumento, una primera aproximación podría ser.

```
Option Explicit

Sub SaberSiExisteHoja()

   MsgBox ExisteHoja( "Hoja3" )

End Sub


'Saber si una hoja existe
Function ExisteHoja(ByVal NombreHoja As String) As Boolean
Dim oHojas As Object

   oHojas = ThisComponent.getSheets()
   ExisteHoja = oHojas.hasByName(NombreHoja)

End Function
```

Podemos hacerla incluso más genérica pasándole el documento donde se desea comprobar la existencia de la hoja y comprobando que sea una hoja de calculo.

```
Option Explicit

Sub SaberSiExisteHoja2()
```

```
   MsgBox ExisteHoja2( ThisComponent, "Hoja3" )

End Sub

'Saber si una hoja existe
Function ExisteHoja2(Documento As Object, NombreHoja As String) As Boolean
Dim oHojas As Object

   'Si no es una hoja de calculo devuelve falso (False)
   If Documento.supportsService("com.sun.star.sheet.SpreadsheetDocument") Then
         oHojas = Documento.getSheets()
         ExisteHoja2 = oHojas.hasByName(NombreHoja)
   End If

End Function
```

Ahora devolvemos la hoja.

```
Option Explicit

Sub DevuelveReferenciaAHoja1()
dim oHoja As Object

   oHoja = DevuelveHoja1( "Datos Enero" )
   If IsNull(oHoja) Then
         MsgBox "La hoja no existe"
   Else
         MsgBox oHoja.getName()
   End If

End Sub


'Saber si existe la hoja y regresarla, si llamas a esta función,
'tienes que verificar que se devolvió algo con IsNull
Function DevuelveHoja1(ByVal NombreHoja As String) As Object
Dim oHojas As Object

   oHojas = ThisComponent.getSheets()
   If oHojas.hasByName(NombreHoja) Then
         DevuelveHoja1 = oHojas.getByName(NombreHoja)
   End If

End Function
```

De nuevo, si lo deseas, puedes pasarle el documento del cual te interesa devolver la hoja, observa como comprobamos también que la hoja exista, esto lo puedes hacer directamente o usando la función creada más arriba, queda a tu criterio.

```
Sub DevuelveReferenciaAHoja2()
dim oHoja As Object

   oHoja = DevuelveHoja2( ThisComponent, "Datos Enero" )
   If IsNull(oHoja) Then
         MsgBox "La hoja no existe"
   Else
         MsgBox oHoja.getName()
   End If

End Sub
```

```
'Saber si existe la hoja y regresarla, si llamas a esta función,
'tienes que verificar que se devolvió algo con IsNull
Function DevuelveHoja2(Documento As Object, NombreHoja As String) As Object
Dim oHojas As Object

   If Documento.supportsService("com.sun.star.sheet.SpreadsheetDocument") Then
         oHojas = Documento.getSheets()
         If oHojas.hasByName(NombreHoja) Then
               DevuelveHoja2 = oHojas.getByName(NombreHoja)
         End If
   End If

End Function
```

6.1.1 Insertando hojas

Para insertar nuevas hojas, usamos el método: insertNewByName(Nombre, Posición), en donde necesitamos el nombre de la nueva hoja a insertar y la posición donde la queremos, la siguiente macro agrega una hoja nueva al inicio de las demás.

```
Sub InsertarNuevaHoja1()
Dim oHojas As Object

   oHojas = ThisComponent.getSheets()
   oHojas.insertNewByName("Datos Sep", 0)

End Sub
```

Ejecuta la macro anterior dos veces y notaras que te dará un error, pues no puedes tener dos hojas con el mismo nombre, entonces, tenemos que verificar si la hoja existe o no.

```
Sub InsertarNuevaHoja2()
Dim oHojas As Object
Dim oHoja As Object
Dim sNombre As String

   'Solicitamos un nombre para la nueva hoja
   sNombre = Trim(InputBox("Nombre de la nueva hoja"))
   'Verificamos que no este vacio
   If sNombre <> "" Then
         'Referencia a todas las hojas
         oHojas = ThisComponent.getSheets()
         'Verificamos si ya existe la hoja
         If Not oHojas.hasByName(sNombre) Then
               'Si no existe la insertamos el inicio
               oHojas.insertNewByName(sNombre, 0)
         Else
               MsgBox "Esta hoja ya existe"
         End If

         'Referencia a la nueva hoja o a la existente
         oHoja = ThisComponent.getSheets.getByName(sNombre)
         'La activamos
         ThisComponent.getCurrentController.setActiveSheet(oHoja)
   End If
```

```
End Sub
```

Observa que si la hoja ya existe, solo hacemos una referencia a ella. La ultima linea para activar la hoja no es necesaria para que la manipules, solo en caso de que quieras que el usuario haga o manipule algo en ella, si no, puedes omitirla. Es sumamente frecuente en programadores noveles, querer "activar" toda hoja que se quiera manipular, esto no es necesariamente así, de hecho, la mayor parte de las veces, con la referencia es suficiente.

También podemos agregar la hoja al final de todas las demás.

```
Sub InsertarNuevaHoja3()
Dim oHojas As Object
Dim oHoja As Object
Dim sNombre As String

   sNombre = Trim(InputBox("Nombre de la nueva hoja"))
   If sNombre <> "" Then
         oHojas = ThisComponent.getSheets()
         If Not oHojas.hasByName(sNombre) Then
               'Si no existe la insertamos al final
               oHojas.insertNewByName( sNombre, oHojas.getCount() )
         End If
         oHoja = ThisComponent.getSheets.getByName(sNombre)
         ThisComponent.getCurrentController.setActiveSheet(oHoja)
   End If

End Sub
```

Observa entonces que el argumento posición puede tomar valores desde 0 y hasta el total de hojas devuelto por getCount() pero incluso si es mayor a este valor, no te dará error y la insertara al final de las demás hojas. Si quieres insertar la hoja antes o después de la hoja activa, primero tienes que encontrar el índice de la hoja activa.

```
Sub IndiceHojaActiva()
Dim co1 As Integer
Dim oHojaActiva As Object
dim oHoja As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   For co1 = 0 To ThisComponent.getSheets.getCount() - 1
         oHoja = ThisComponent.getSheets.getByIndex(co1)
         If oHoja.getName() = oHojaActiva.getName() then
               MsgBox "El indice de la hoja activa es: " & co1
               Exit For
         End If
   Next

End Sub
```

Que podemos convertir a una función.

```
Function BuscarIndiceHoja(ByVal NombreHoja As String) As Integer
Dim co1 As Integer
dim oHoja As Object

   For co1 = 0 To ThisComponent.getSheets.getCount() - 1
         oHoja = ThisComponent.getSheets.getByIndex(co1)
         If oHoja.getName() = NombreHoja then
               BuscarIndiceHoja = co1
```

```
            Exit Function
        End If
    Next
    BuscarIndiceHoja = -1

End Function
```

Observa que si no la encuentra el valor devuelto es -1, lo cual hay que evaluar en caso necesario. Para insertar antes de la hoja activa usamos.

```
Sub InsertarNuevaHoja4()
Dim oHojas As Object
Dim oHoja As Object
Dim oHojaActiva As Object
Dim sNombre As String
Dim pos As Integer

   sNombre = Trim(InputBox("Nombre de la nueva hoja"))
   If sNombre <> "" Then
        oHojas = ThisComponent.getSheets()
        oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
        If Not oHojas.hasByName(sNombre) Then
            'Buscamos el índice de la hoja activa
            pos = BuscarIndiceHoja(oHojaActiva.getName())
            oHojas.insertNewByName( sNombre, pos )
        End If
        oHoja = ThisComponent.getSheets.getByName(sNombre)
        ThisComponent.getCurrentController.setActiveSheet(oHoja)
   End If

End Sub
```

Para insertar después de la hoja activa, solo sumas uno al valor devuelto por la función BuscarIndiceHoja.

```
pos = BuscarIndiceHoja(oHojaActiva.getName()) + 1
```

Podemos aventurar una primera versión de una función genérica para insertar una nueva hoja de calculo donde quieras.

```
'Como función regresamos la nueva hoja insertada o la existente en su caso
'Posición:    1 = Inicio
'             2 = Final
'             3 = Antes de la hoja activa
'             4 = Después de la hoja activa
Function getNuevaHoja(NombreHoja As String, Posicion As Integer) As Object
Dim oHojas As Object
Dim oHojaActiva As Object
Dim iPos As Integer

   If NombreHoja <> "" Then
        oHojas = ThisComponent.getSheets()
        oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
        Select Case Posicion
            Case 1 : iPos = 0
            Case 2 : iPos = oHojas.getCount()
            Case 3 : iPos = oHojaActiva.getRangeAddress.Sheet
            Case 4 : iPos = oHojaActiva.getRangeAddress.Sheet + 1
            Case Else : iPos = 0
        End Select
```

```
        If Not oHojas.hasByName(NombreHoja) Then
            oHojas.insertNewByName(NombreHoja, iPos)
        End If
        getNuevaHoja = ThisComponent.getSheets.getByName(NombreHoja)
    End If

End Function
```

Nota como estamos obteniendo el índice de la hoja activa (oHoja.getRangeAddress.Sheet), con lo cual nos evitamos tener que hacer un ciclo por las hojas del documento. Puedes mejorar esta función para insertar la hoja en cualquier otro documento, así como ingeniártelas para pedirle al usuario el número de hojas nuevas que quiera insertar e insertarlas por supuesto, esa, es tu tarea.

6.1.2 Borrando hojas

Para borrar hojas, usamos el método removeByName(Nombre), donde Nombre es el nombre de la hoja que queremos borrar, no esta de más recomendarte usar con cuidado el borrado de hojas, aunque algo que me gusta mucho de OpenOffice.org, es que muchas de las acciones que hacemos por código, son susceptibles de deshacerse de forma normal con la barra de herramientas o con CTRL+Z, el borrado de hojas es una de ellas, compruébalo.

```
Sub BorrarHoja1()
Dim oHojas As Object

   oHojas = ThisComponent.getSheets()
   'Borramos la hoja por su nombre
   oHojas.removeByName( "Hoja11" )

End Sub
```

Por supuesto no puedes borrar una hoja que no exista, así que verifícalo.

```
Sub BorrarHoja2()
Dim oHojas As Object
Dim sNombre As String

   sNombre = Trim( InputBox( "Nombre de la hoja a borrar" ))
   If sNombre <> "" Then
        oHojas = ThisComponent.getSheets()
        If oHojas.hasByName( sNombre ) Then
            oHojas.removeByName( sNombre )
        Else
            MsgBox "La hoja no existe"
        End If
   End If

End Sub
```

Puedes borrar la hoja activa.

```
Sub BorrarHojaActiva()
Dim oHojas As Object
Dim oHojaActiva As Object
```

```
    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
    oHojas = ThisComponent.getSheets()
    oHojas.removeByName( oHojaActiva.getName() )

End Sub
```

Ejecuta la macro anterior hasta que no quede ni una, claro, no te dejara por que un documento de hoja de calculo por lo menos debe de tener una hoja, para evitar el error que te da al tratar de borrar la ultima hoja, valida que siempre quede más de una.

```
Sub BorrarHoja3()
Dim oHojas As Object
Dim sNombre As String

   sNombre = Trim( InputBox( "Nombre de la hoja a borrar" ))
   If sNombre <> "" Then
         oHojas = ThisComponent.getSheets()
         If oHojas.hasByName( sNombre ) And oHojas.getCount()>1 Then
               oHojas.removeByName( sNombre )
         Else
               MsgBox "La hoja no existe o solo queda una"
         End If
   End If

End Sub
```

6.1.3 Moviendo hojas

Para mover hojas usamos el método moveByName(NombreHoja, PosicionNueva), donde NombreHoja tiene que ser el nombre de una hoja existente.

```
Sub MoverHoja1()
Dim oHojas As Object

   oHojas = ThisComponent.getSheets()
   'Movemos la hoja especificada al inicio
   oHojas.moveByName( "Hoja2", 0 )

End Sub
```

Ahora, la movemos al final.

```
Sub MoverHoja2()
Dim oHojas As Object

   oHojas = ThisComponent.getSheets()
   'Movemos la hoja especificada al final
   oHojas.moveByName( "Hoja2", oHojas.getCount() )

End Sub
```

Si usas una versión anterior a la 3.1, ten cuidado con mover tus hojas “no más allá de la ultima hoja”, pues te puede provocar un error en toda la aplicación, este error (bug) se reportó en su momento (http://www.openoffice.org/issues/show_bug.cgi?id=92477), y ha sido arreglado a partir de esta versión (3.1), aun así, usa siempre getCount() para asegurarte.

Vamos a hacer algo muy divertido, como ya sabemos obtener los nombres e índices de cada hoja, y ahora hemos aprendido a mover hojas, hagamos una macro que nos ordene alfabéticamente nuestras hojas, para ello nos apoyaremos en una función que ordena la matriz que le pasemos.

```
Sub PruebaOrdenar1()
Dim mDatos()

   GlobalScope.BasicLibraries.LoadLibrary( "Tools" )
   mDatos() = Array("5","l","o","f","e","v","y","d","h","u",)

   'Mostramos los datos desordenados
   MsgBox Join( mDatos(), Chr(13) )

   'Ordenamos los datos
   mDatos() = BubbleSortList( mDatos() )

   'Mostramos los datos ordenados
   MsgBox Join( mDatos(), Chr(13) )

End Sub
```

La función BubbleSortList, viene incorporada a las macros de OpenOffice.org y usa el método de burbuja para ordenar la lista pasada, no es el algoritmo más eficiente, pero en listas pequeñas y dada su sencillez de implementación es perfectamente válida para nuestros fines, si quieres saber más de este algoritmo visita: http://es.wikipedia.org/wiki/Bubblesort

También puedes crearte tu versión de este algoritmo, aquí esta la mía, puedes ordenar de forma ascendente o descendente.

```
Sub PruebaOrdenar2()
Dim mDatos()

   mDatos() = Array("5","l","o","f","e","v","y","d","h","u",)
   'Mostramos los datos desordenados
   MsgBox Join( mDatos(), Chr(13) )

   'Ordenamos los datos
   Call OrdenarBurbuja( mDatos(), 1 )

   'Mostramos los datos ordenados
   MsgBox Join( mDatos(), Chr(13) )

End Sub


' Datos = matriz de datos a ordenar
' Orden =     1 ascendente
'             2 descendente
Sub OrdenarBurbuja(ByRef Datos() As Variant, ByVal Orden As Integer)
Dim NumDatos As Long
Dim co1 As Long, co2 As Long

   NumDatos = UBound(Datos)
   For co1 = 1 To NumDatos
         For co2 = NumDatos To co1 Step -1
               If Orden = 1 Then
                     If Datos(co2) < Datos(co2 - 1) Then
                           Call Intercambio Datos(co2), Datos(co2 - 1)
                     End If
               Else
```

```
                    If Not (Datos(co2) < Datos(co2 - 1)) Then
                         Call Intercambio Datos(co2), Datos(co2 - 1)
                    End If
               End If
          Next co2
     Next co1

End Sub


'Macro para intercambiar un par de valores
Sub Intercambio(ByRef Dato1 As Variant, ByRef Dato2 As Variant)
Dim sTmp As Variant

   sTmp = Dato1
   Dato1 = Dato2
   Dato2 = sTmp

End Sub
```

Ahora si, nuestra macro para ordenar hojas en acción, puedes usar la función o macro que prefieras para ordenar la matriz de nombres obtenida, incluso, crearte tu propia versión si así lo prefieres.

```
Option Explicit

'Ordenamos las hojas por orden alfabético
Sub OrdenarHojas()
Dim oHojas As Object
Dim mNombres() As Variant
Dim aTmp As Variant
Dim co1 As Integer
Dim oHoja As Object

   'Referencia a todas las hojas del documento
   oHojas = ThisComponent.getSheets()

   'Matriz con los nombres de todas las hojas
   mNombres() = oHojas.getElementNames()

   'Ordenamos la matriz
   Call OrdenarBurbuja( mNombres(), 1 )

   'Recorremos la matriz
   For co1 = LBound( mNombres() ) To UBound( mNombres() )

         'El índice en la matriz, sera el mismo de la posición
         oHojas.moveByName( mNombres(co1), co1 )

   Next co1

End Sub
```

6.1.4 Copiando hojas

Para copiar hojas usamos el método copyByName (NombreHoja, NombreNuevo, Posicion) de la siguiente manera.

```
Sub CopiarHoja1()
Dim oHojas As Object

   oHojas = ThisComponent.getSheets()

   'Copiamos la "hoja2" como "Nueva Hoja2" al inicio
   oHojas.copyByName( "Hoja2", "Nueva Hoja2", 0 )

End Sub
```

Por supuesto el nombre de la hoja a copiar “debe” de existir y el nombre nuevo de la hoja “no debe” existir, lo mejor, es evaluarlo antes.

```
Sub CopiarHoja2()
Dim oHojas As Object
Dim sNombre As String
Dim sNombreNuevo As String

   sNombre = "Datos"
   sNombreNuevo = "Datos nuevos"

   oHojas = ThisComponent.getSheets()

   'Solo copia la hoja, si la hoja a copiar existe y el nombre nuevo no existe
   If oHojas.hasByName(sNombre) And (Not oHojas.hasByName(sNombreNuevo)) Then
         oHojas.copyByName( sNombre, sNombreNuevo, oHojas.getCount() )
   Else
         MsgBox "No se copió la hoja"
   End If

End Sub
```

Podemos intentar copiar una hoja y asegurarnos de que el nombre no exista, tomando como base el nombre actual de la hoja, por ejemplo.

```
Sub CopiarHojaActiva()
Dim oHojas As Object
Dim oHoja As Object
Dim sNombre As String
Dim sNombre2 As String
Dim co1 As Long

   'Referencia a todas las hoja
   oHojas = ThisComponent.getSheets()

   'Nombre de la hoja activa
   sNombre = ThisComponent.getCurrentController.getActiveSheet.getName()

   'Contador para construir el nuevo nombre
   co1 = 1

   'El nuevo nombre es igual mas un guión bajo y un numero
   sNombre2 = sNombre & "_" & Format(co1)

   'Hace el ciclo mientras el nuevo nombre exista
   Do While oHojas.hasByName( sNombre2 )
         'Si ya existe incrementamos el contador
         co1 = co1 + 1
         'y construimos el nuevo nombre
         sNombre2 = sNombre & "_" & Format(co1)
   Loop
```

```
    'Sale del ciclo cuando el nuevo nombre no exista, entonces
    'podemos copiar la hoja al final (o donde quieras)
    oHojas.copyByName(sNombre, sNombre2, oHojas.getCount())

    'Referencia a la nueva hoja
    oHoja = ThisComponent.getSheets.getByName(sNombre2)

    'Y la activamos
    ThisComponent.getCurrentController.setActiveSheet(oHoja)

End Sub
```

Solo te resta preguntarle al usuario cuantas nuevas hojas quiere e insertar ese número de hojas nuevas, pero esa, es tu tarea.

6.1.5 Renombrando hojas

Para renombrar hojas usamos el método setName, de la siguiente manera.

```
Sub CambiarNombreHoja1()
Dim oHojas As Object

   oHojas = ThisComponent.getSheets()
   'Cambiamos el nombre de la hoja “Hola”, por “Prueba”
   oHojas.getByName("Hola").setName( "Prueba" )

End Sub
```

También puedes usar el índice para referirte a la hoja.

```
Sub CambiarNombreHoja2()
Dim oHojas As Object

   oHojas = ThisComponent.getSheets()
   oHojas.getByIndex( 0 ).setName( "Gastos" )

End Sub
```

Es recomendable validar que la hoja a renombrar exista y que el nuevo nombre no.

```
Sub CambiarNombreHoja3()
Dim oHojas As Object
Dim sNombreActual As String
Dim sNombreNuevo As String

   sNombreActual = "Resumen"
   sNombreNuevo = "Terminado"
   oHojas = ThisComponent.getSheets()

   'Validamos que la hoja exista y el nuevo nombre no
   If oHojas.hasByName(sNombreActual) And (Not oHojas.hasByName(sNombreNuevo)) Then
         oHojas.getByName( sNombreActual ).setName( sNombreNuevo )
   Else
```

```
        MsgBox "No se renombro la hoja"
    End If

End Sub
```

Solo para divertirnos, cambiamos los nombres de las hojas por números.

```
Sub CambiarNombreHoja4()
Dim oHojas As Object
Dim co1 As Integer

    oHojas = ThisComponent.getSheets()

    For co1 = 1 To oHojas.getCount()
        oHojas.getByIndex( co1-1 ).setName( co1 )
    Next

End Sub
```

Ahora por letras, el código siguiente podría fallarte si tienes más de 25 hojas en tu documento, tu tarea es decirme ¿porqué? y corregirlo, en algunos casos, “el cambio de nombre no tendrá efecto”, también, te toca averiguar ¿porqué?

```
Sub CambiarNombreHoja5()
Dim oHojas As Object
Dim co1 As Integer

    oHojas = ThisComponent.getSheets()

    For co1 = 1 To oHojas.getCount()
        oHojas.getByIndex( co1-1 ).setName( Chr( co1+64 ) )
    Next

End Sub
```

O los meses del año:

```
Sub CambiarNombreHoja6()
Dim oHojas As Object
Dim co1 As Integer
Dim Limite As Byte
Dim sMes As String

    oHojas = ThisComponent.getSheets()

    'Para que solo cambie las primeras 12
    If oHojas.getCount() > 12 Then
        Limite = 12
    Else
        'O las que haya si son menos de 12
        Limite = oHojas.getCount()
    End If

    For co1 = 1 To Limite
        'Obtenemos el nombre del mes
        sMes = Format( DateSerial(Year(Date),co1,1) ,"mmmm")
        oHojas.getByIndex( co1-1 ).setName( sMes )
    Next

End Sub
```

Te queda de tarea lograr completar los meses para que sean los doce del año, es decir, tienes que insertar los meses que te hagan falta si el documento tiene menos de las hojas necesarias, si tiene más de doce borra las sobrantes.

6.1.6 Ocultando y mostrando hojas

Mostrar y ocultar hojas es muy sencillo, solo hay que establecer su propiedad isVisible en verdadero (True) o falso (False) según se requiera de la siguiente manera,

```
Sub OcultarHoja1()
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   oHojaActiva.isVisible = False

End Sub
```

La macro anterior te ocultara la hoja activa, si la ejecutas varias veces te ira ocultando tus hojas hasta que solo quede una, si solo hay una no te dará error, pero la dejará visible por que, como sabes, tiene que haber al menos una hoja visible en un documento de Calc. También puedes ocultar una hoja por medio de su índice, como en.

```
Sub OcultarHoja2()
Dim oHoja As Object

   oHoja = ThisComponent.getSheets.getByIndex(1)

   oHoja.isVisible = False

End Sub
```

Toma en cuenta que: el índice de la hoja “debe” existir, sino te dará un error, así mismo, si ocultas una hoja, esta no cambia de índice por lo que puedes usar el mismo para acceder a ella, aunque este oculta, la siguiente macro alterna entre mostrar y ocultar la primer hoja del documento.

```
Sub OcultarHoja3()
Dim oHoja As Object

   oHoja = ThisComponent.getSheets.getByIndex(0)

   oHoja.isVisible = Not oHoja.isVisible

End Sub
```

Como ya habrás intuido, para mostrar una hoja oculta simplemente hay que establecer esta propiedad en verdadero (True).

```
Sub OcultarHoja4()
Dim oHoja As Object

   oHoja = ThisComponent.getSheets.getByIndex(1)
```

```
   oHoja.isVisible = True

End Sub
```

El siguiente código te oculta todas las hojas, excepto la hoja activa.

```
Sub OcultarHoja5()
Dim oHojaActiva As Object
Dim oHojas As Object
Dim co1 As Long

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oHojas = ThisComponent.getSheets()
   For co1 = 0 To oHojas.getCount() - 1
        If oHojas.getByIndex(co1).getName <> oHojaActiva.getName() Then
             oHojas.getByIndex(co1).isVisible = False
        End If
   Next

End Sub
```

Tu tarea es hacer la función inversa, muestra todas las hojas ocultas.

6.1.7 Protección y desprotección de hojas

Para terminar este capítulo, veamos como proteger una hoja, es decir, establecer una contraseña para evitar modificaciones a la misma, además, recuerda que para que la protección de celdas individuales sea efectiva, la hoja debe estar protegida. Para proteger una hoja, usamos el método *Protect*, pasándole como argumento, la contraseña que queremos establecer, por supuesto, puedes pasarle una contraseña vacía, con lo que la hoja no estará muy protegida que digamos, pero creeme, muchos usuarios no saben desproteger una hoja, aun sin contraseña.

```
Sub ProtegerHoja1()
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   'Para que veas que si me ocurren otras contraseñas
   oHojaActiva.Protect( "letmein" )

End Sub
```

Y para desproteger, usamos el método *unProtect*, si la hoja tiene contraseña hay que pasársela como argumento, si no es correcta, el método no te devolverá ningún error como en la interfaz del usuario que te avisa que la contraseña es incorrecta, para saber si tuvo éxito o no la desprotección, hay que verificar la propiedad *isProtected*, si es verdadera (True) la hoja sigue protegida, si es falsa (False), la hoja esta desprotegida.

```
Sub ProtegerHoja2()
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   If oHojaActiva.isProtected Then
        MsgBox "La hoja esta protegida"
        'Intentamos desprotegerla
```

```
        oHojaActiva.unProtect( "letmein" )
        'Verificamos si tuvo éxito la desprotección
        If oHojaActiva.isProtected Then
            MsgBox "La contraseña no es correcta"
        Else
            MsgBox "Hoja desprotegida correctamente"
        End If
    Else
        MsgBox "La hoja NO esta protegida"
    End If

End Sub
```

Te queda de tarea modificar la macro para solicitar al usuario la contraseña, verificar si es correcta o no y darle solo tres intentos para ingresarla. Fácil, ¿verdad?. Como comentarios finales, si intentas proteger una hoja que ya tiene contraseña, no obtendrás ningún error, pero la hoja permanecerá con la contraseña original, para cambiarla, primero tienes que desprotegerla y después cambiarla. Cuando proteges una hoja e intentas hacer modificaciones a esta, por ejemplo, escribir en una celda, tampoco retornará ningún error, pero no será efectiva la modificación, procura usar la propiedad para saber si una hoja esta o no protegida (*isProtected*), para actuar en consecuencia.

6.2 Referencia a rangos

Seguro que sabes, si eres usuario habitual de una hoja de calculo, que el trabajo con rangos es esencial en estos documentos, por lo mismo, el trabajo con rangos desde código OOo Basic es igualmente importante, ya vimos como aseguramos que estamos trabajando en una hoja de calculo, así que dejo a tu criterio esta validación. En la siguientes secciones nos centraremos en aprender como hacer referencia a distintos tipos de rangos para después poder manipularlos, darles formato o hacer con ellos lo que queramos.

6.2.1 Referencia a celdas individuales

Podemos acceder a las celdas de una hoja de calculo de varias maneras, principalmente por su nombre o por su posición, pero muy importante, primero tienes que acceder a la hoja donde están las celdas que te interesa manipular, como acceder a hojas es un tema que ya hemos tratado, pero en cada ejemplo podrás notar que repasamos estos conocimientos, la forma más simple de hacer referencia a una celda es por su nombre.

```
Sub AccesoCeldas1()
Dim oHojaActiva As Object
Dim oCelda As Object

   'Referencia a la hoja activa
   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Referencia a la celda E5
   oCelda = oHojaActiva.getCellRangeByName( "E5" )
```

```
    'Mostramos el contenido de la celda
    MsgBox oCelda.getString()

    'Mostramos la hoja, columna y fila correspondiente a esta celda
    MsgBox "Hoja: " & oCelda.getCellAddress.Sheet & Chr(13) & _
               "Columna: " & oCelda.getCellAddress.Column & Chr(13) & _
               "Fila: " & oCelda.getCellAddress.Row

End Sub
```

Observa como comprobamos en la ultima linea, que efectivamente hemos hecho referencia a la celda que nos interesa, es decir a la celda E5, que en columna y fila es la 4, por que recordamos que los números de columna y fila empiezan en 0, observa la estructura getCellAddress, esta, es muy importante pues a muchos métodos para manipular celdas, se les tienen que pasar estructuras como esta, solo tiene tres propiedades, la hoja (valor tipo integer, este, también empieza en 0) donde esta la celda referenciada, la columna (long) y la fila (long) de esta celda.

Ahora accedemos a una celda por su posición, recuerda que los índices de inicio desde código empiezan en 0, por lo que para hacer referencia a la celda E5, tenemos que poner la columna 4 y fila 4, el primer valor es para la columna y el segundo para la fila, no esta de más comentarte que tengas cuidado de no establecer una posición fuera de la hoja, pues te dará un error, por ejemplo, establecer el valor de la columna en 256 o superior si trabajos con la versión 2.x de OpenOffice.org, en la versión 3.x tenemos 1024 columnas para trabajar, por supuesto, si el valor de la fila y columna se la solicitas al usuario, “deberías” de validar que los valores proporcionados son correctos.

```
Sub AccesoCeldas2()
Dim oHojaActiva As Object
Dim oCelda As Object
Dim Col As Long
Dim Fil As Long

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Nos aseguramos que sea un valor con Val y que sea positivo con Abs
   Col = Abs(Val(InputBox("Dame la columna")))
   Fil = Abs(Val(InputBox("Dame la fila")))

   'Nos aseguramos que estén dentro de los rangos correctos
   If Col < oHojaActiva.Columns.Count And Fil < oHojaActiva.Rows.Count Then
         'Accedemos a la celda
         oCelda = oHojaActiva.getCellByPosition( Col,Fil )
         MsgBox oCelda.getString()
   Else
         MsgBox "Valores de celda incorrectos"
   End If

End Sub
```

Es frecuente que el acceso por nombre a una celda se use para establecer valores preestablecidos, como títulos de campos por ejemplo, y el acceso por posición es muy útil para realizar ciclos, como el ejemplo siguiente que inserta el año como titulo en la celda A1 y los meses del año de la celda A2 a la A13.

```
Sub AccesoCeldas3()
Dim oHojaActiva As Object
Dim oCelda As Object
Dim col As Integer
```

```
    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

    'Referencia a celda por nombre
    oCelda = oHojaActiva.getCellRangeByName( "A1" )
    oCelda.setString( Year(Date) )

    For co1 = 1 To 12
          'Referencia a celda por posicion
          oCelda = oHojaActiva.getCellByPosition( 0,co1 )
          oCelda.setString( Format( DateSerial(Year(Date),co1,1) ,"mmmm") )
    Next

End Sub
```

Observa que hemos estado usando el método getString() para obtener el contenido de una celda y setString(Valor As String) para establecerlo, más adelante veremos todas las posibilidades que tenemos para introducir u obtener datos de las celdas de nuestra hoja de calculo, así como sus diferencias.

6.2.2 Referencia a un rango de celdas

Podemos acceder a un rango de celdas por su nombre, usando el mismo método usado para acceder a una celda.

```
Sub AccesoRango1()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Referencia al rango A1:E5
   oRango = oHojaActiva.getCellRangeByName( "A1:E5" )

   'Y lo seleccionamos
   ThisComponent.getCurrentController.select(oRango)

End Sub
```

Para acceder a un rango de celdas por su posición, hay que usar un método diferente: getCellRangeByPosition, que requiere de cuatro argumentos.

```
Sub AccesoRango2()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Seleccionamos el rango B1:B10
   oRango = oHojaActiva.getCellRangeByPosition( 1,0,1,9 )

   'Y lo seleccionamos
   ThisComponent.getCurrentController.select(oRango)

End Sub
```

Observa que ahora usamos el método getCellRangeByPosition, los argumentos pasados a este método son cuatro, la columna y fila donde empieza nuestro rango y la columna y

fila donde termina, recuerda que los números de fila y columna empiezan en 0, algunos piensan que los dos últimos argumentos son el ancho y alto del rango a usar, no es así, estos argumentos también son números de índices de columna y fila respectivamente y tienes que tener la precaución de establecer los segundos iguales o mas grandes que los primeros, sino, te dará un error en tiempo de ejecución y por supuesto sin sobrepasar el máximo de filas y columnas de la hoja de calculo. Observa también, como en la ultima línea seleccionamos el rango referenciado.

Otra posibilidad, es usar nombres definidos de rangos, es decir, aquellos que establecemos desde el "Cuadro de nombre" en la hoja de calculo, ya sabes, ese cuadro de lista desplegable (combobox) que esta al lado de la barra de formulas, que también puedes establecer desde el menú I*nsertar* | *Nombres* | *Definir...* cuya teclas de acceso rápido son Ctrl+F3. En el siguiente ejemplo, seleccionamos el rango de celdas llamado Datos. Toma nota de que si el rango no existe en la hoja desde donde se intenta referenciar, te dará un error.

```
Sub AccesoRango3()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Seleccionamos el rango por su nombre definido
   oRango = oHojaActiva.getCellRangeByName( "Datos" )

   'Y lo seleccionamos
   ThisComponent.getCurrentController.select(oRango)

End Sub
```

Por supuesto, si el nombre del rango no existe, te dará un error en tiempo de ejecución, este método no es sensible a mayúsculas o minúsculas. Para hacer referencia a la hoja correcta donde exista el rango con nombre, observa como obtenemos la hoja donde se encuentra. Si el usuario es el que proporciona el nombre del rango, como siempre, es mejor validar que el rango exista.

```
Sub AccesoRango4()
Dim oHoja As Object
Dim oRango As Object
Dim oRangosPorNombre As Object
Dim sNombreRango As String

   'Referencia a todos los rangos con nombre en la hoja de calculo
   oRangosPorNombre = ThisComponent.NamedRanges()
   sNombreRango = Trim( InputBox( "Escribe el nombre del rango a seleccionar" ) )
   'Comprobamos que el rango exista
   If oRangosPorNombre.hasByName( sNombreRango ) Then
         oHoja =
ThisComponent.getSheets.getByIndex(oRangosPorNombre.getByName(sNombreRango).getReferredCells.getRangeA
ddress.Sheet)
         'Seleccionamos el rango por su nombre definido
         oRango = oHoja.getCellRangeByName( sNombreRango )
         'Y lo seleccionamos
         ThisComponent.getCurrentController.select(oRango)
   Else
         MsgBox "El rango " & sNombreRango & " no existe en el documento"
   End If

End Sub
```

No confundas estos nombres de rangos, con los que puedes establecer en el menú *Datos* | *Definir rango...*, ya que estos últimos se refieren a rangos considerados como una tabla de

datos, de hecho, puedes tener un mismo nombre para un rango de celdas y para un rango de datos, pero son dos cosas diferentes, los segundos, los veremos más adelante.

De los rangos de celdas, también es posible obtener información, para ello se hace uso de la estructura CellRangeAddress a través del método getRangeAddress que te devuelve información de: la hoja donde esta el rango, la columna y fila donde comienza y la columna y fila donde acaba.

```
Sub AccesoRango5()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oDirCelda As Object
Dim sTmp As String

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName( "Nombres" )

   'Obtenemos la información de la dirección
   oDirCelda = oRango.getRangeAddress()

   'Construimos el texto informativo
   sTmp = "El rango esta en la hoja: " & oDirCelda.Sheet & Chr(13) & _
              "Columna Inicio: " & oDirCelda.StartColumn & Chr(13) & _
              "Fila Inicio: " & oDirCelda.StartRow & Chr(13) & _
              "Columna Fin: " & oDirCelda.EndColumn & Chr(13) & _
              "Fila Fin: " & oDirCelda.EndRow

   MsgBox sTmp

End Sub
```

Esta estructura (getRangeAddress) también es usada por varios métodos para manipular rangos que veremos más adelante, por lo que es importante que la tengas presente.

6.2.3 Referencia a varios rangos de celdas

Cuando en la interfaz del usuario, hacemos la selección de un rango y mantenemos presionada la tecla Ctrl y hacemos un segunda selección de un rango y así sucesivamente, estamos hablando de un conjunto de rangos que tiene sus características particulares para su control y manejo. Para seleccionar varios rangos desde código, primero debemos crear el “contenedor” de estos rangos, veamos como.

```
Sub Acceso_A_Rangos1()
Dim oHojaActiva As Object
Dim oRangos As Object
Dim oRango As Object
Dim oDirRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Creamos el contender para los rangos
   oRangos = ThisComponent.createInstance("com.sun.star.sheet.SheetCellRanges")

   'Creamos la estructura CellRangeAddress necesaria
   oDirRango = oHojaActiva.getCellRangeByName( "A1:B2" ).getRangeAddress()
   'Y lo agregamos al contenedor de rangos
   oRangos.addRangeAddress( oDirRango ,False )

   'Aquí solo hacemos referencia al rango
   oRango = oHojaActiva.getCellRangeByName( "E1:G2" )
```

```
    'Y lo agregamos al contenedor de rangos, nota como tenemos que usar
    'el método getRangeAddress para pasarle el argumento correcto
    oRangos.addRangeAddress( oRango.getRangeAddress() ,False )

    'Aquí agregamos un rango directamente al contenedor
    'toma en cuenta que se van acumulando
    oRangos.addRangeAddress( oHojaActiva.getCellRangeByName( "D4:E5" ).getRangeAddress() ,False )

    'Comprobamos que están correctos seleccionándolos
    ThisComponent.getCurrentController.select(oRangos)

End Sub
```

No te confundas, los tres líneas donde se agrega el rango con el método addRangeAddress() son iguales, lo que cambia es la forma en que hacemos referencia a la dirección del rango, si vas empezando a programar, te recomiendo ir desglosando cada línea, con el tiempo y la experiencia, podrás concentrar código de forma más natural, por ahora, usa todas las líneas que consideres pertinentes para que tu código lo entiendas con solo verlo.

Por supuesto, también de estos rangos podemos obtener información, por ejemplo, los nombres de los rangos que contiene,

```
Sub Acceso_A_Rangos2()
Dim oHojaActiva As Object
Dim oRangos As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Creamos el contender para los rangos
   oRangos = ThisComponent.createInstance("com.sun.star.sheet.SheetCellRanges")

   'Agregamos los rangos que queramos
   oRangos.addRangeAddress( oHojaActiva.getCellRangeByName( "A1:A2" ).getRangeAddress() ,False )
   oRangos.addRangeAddress( oHojaActiva.getCellRangeByName( "B4:B5" ).getRangeAddress() ,False )
   oRangos.addRangeAddress( oHojaActiva.getCellRangeByName( "C7:C8" ).getRangeAddress() ,False )
   oRangos.addRangeAddress( oHojaActiva.getCellRangeByName( "D10:D11" ).getRangeAddress() ,False )
   oRangos.addRangeAddress( oHojaActiva.getCellRangeByName( "E13:E14" ).getRangeAddress() ,False )

   'Mostramos las direcciones de los rangos
   MsgBox oRangos.getRangeAddressesAsString()

End Sub
```

Nota como los rangos es una cadena larga de las direcciones de los rangos separados por un “ ; ”, si lo quieres mejor presentable, reemplaza los puntos y comas por un salto de línea con el siguiente código.

```
    'Lo único que hacemos es reemplazar los ; por saltos de linea
    sTmp = Join( Split(oRangos.getRangeAddressesAsString(),";"),Chr(13))
    'Mostramos el resultado
    MsgBox sTmp
```

Los rangos también se pueden remover de la colección.

```
Sub Acceso_A_Rangos4()
Dim oHojaActiva As Object
Dim oRangos As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Creamos el contender para los rangos
   oRangos = ThisComponent.createInstance("com.sun.star.sheet.SheetCellRanges")
```

```
    'Agregamos los rangos que queramos
    oRangos.addRangeAddress( oHojaActiva.getCellRangeByName( "A1:A2" ).getRangeAddress() ,False )
    oRangos.addRangeAddress( oHojaActiva.getCellRangeByName( "B4:B5" ).getRangeAddress() ,False )
    oRangos.addRangeAddress( oHojaActiva.getCellRangeByName( "C7:C8" ).getRangeAddress() ,False )
    oRangos.addRangeAddress( oHojaActiva.getCellRangeByName( "D10:D11" ).getRangeAddress() ,False )

    'Comprobamos que estén los rangos
    MsgBox oRangos.getRangeAddressesAsString()

    'Removemos el rango deseado
    oRangos.removeRangeAddress( oHojaActiva.getCellRangeByName( "D10:D11" ).getRangeAddress() )

    'Volvemos a verificar que se haya removido
    MsgBox oRangos.getRangeAddressesAsString()

End Sub
```

Cuidado, si el rango que quieres remover no existe en la colección, te dará un error en tiempo de ejecución, lo mejor es validar antes que existe...

```
    'Validamos que el rango a remover exista en la colección
    If oRangos.hasByName("Hoja1.D10:D11") Then
          'Removemos el rango deseado
          oRangos.removeRangeAddress(oHojaActiva.getCellRangeByName( "D10:D11" ).getRangeAddress())
    End If
```

El código anterior tiene un problema que tal vez te pueda causar un pequeño inconveniente, cuando consultamos con hasByName si el rango existe, observa que le pasamos el nombre del rango como texto ("Hoja1.D10:D11") y que incluye el nombre de la hoja donde esta el rango, pero observa como al método removeRangeAddress, le pasamos un objeto (oHojaActiva) que no necesariamente tiene que corresponder con el nombre de hoja (Hoja1), de tarea te queda garantizar que se corresponde uno con el otro, es decir, que el objeto que apunta a la hoja, efectivamente corresponda en nombre con el rango pasado.

6.2.4 Referencia a filas y columnas

Las filas y columnas de una hoja de calculo siguen siendo rangos de celdas, lo único que las caracteriza, en función de hacer referencia a ellas, es que abarcan la totalidad de celdas que contienen, como en.

```
Sub AccesoColumna1()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Referencia a la columna A
   oRango = oHojaActiva.getCellRangeByName( "A1:A65536" )

   'Y lo seleccionamos
   ThisComponent.getCurrentController.select(oRango)

End Sub
```

Nota como tomamos de la fila 1 y hasta la 65536 que es el número total de filas con que por ahora cuentan las hojas de calculo de Calc, si bien el número de filas cambia realmente

poco y hasta ahora, siempre hacia arriba, no es buena idea usar estos valores, es mejor usar un nombre y hacer referencia a la columna completa como veremos más adelante. Para acceder al nombre de una columna previamente establecido, usamos.

```
Sub AccesoColumna2()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Referencia a la columna A que se nombró previamente
   oRango = oHojaActiva.getCellRangeByName( "Claves" )

   'Y lo seleccionamos
   ThisComponent.getCurrentController.select(oRango)

End Sub
```

Podemos hacer lo mismo con las filas.

```
Sub AccesoFilas1()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Referencia a la fila 10
   oRango = oHojaActiva.getCellRangeByName( "A10:AMJ10" )

   'Y lo seleccionamos
   ThisComponent.getCurrentController.select(oRango)

End Sub


Sub AccesoFilas2()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Referencia a la fila 15 que se nombró previamente
   oRango = oHojaActiva.getCellRangeByName( "Registros" )

   'Y lo seleccionamos
   ThisComponent.getCurrentController.select(oRango)

End Sub
```

OOo Basic cuenta con métodos específicos para manipular filas y columnas, pero propiamente hablando, al acceder por medio de estos métodos, dejan de ser rangos de celdas y pasan a ser objetos ScTableRowsObj y ScTableColumnsObj respectivamente como lo demostramos a continuación.

```
Sub AccesoColumna3()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Referencia al rango A1:E1
```

```
    oRango = oHojaActiva.getCellRangeByName( "A1:E1" )
    'Mostramos el tipo de objeto
    MsgBox oRango.getImplementationName()
    'Renombramos el objeto oRango accediendo a sus columnas
    oRango = oRango.getColumns()
    'Verificamos el tipo de objeto
    MsgBox oRango.getImplementationName()

End Sub
```

Trata de seleccionar el rango tal y como queda al final del código y veras que te da un error, también, trata de acceder a la información de depuración como se vio en capítulos anteriores y nota como implementan métodos y propiedades diferentes. Es el mismo caso para las filas.

```
Sub AccesoFilas3()
Dim oHojaActiva As Object
Dim oRango As Object

    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

    'Referencia al rango A1:A10
    oRango = oHojaActiva.getCellRangeByName( "A1:A10" )
    'Mostramos el tipo de objeto
    MsgBox oRango.getImplementationName()
    'Renombramos el objeto oRango accediendo a sus filas
    oRango = oRango.getRows()
    'Verificamos el tipo de objeto
    MsgBox oRango.getImplementationName()

End Sub
```

Por lo anterior, la recomendación es que mantengas tu referencia original al rango y uses una segunda variable si necesitas acceder a las filas o columnas completas como te muestro en el siguiente ejemplo.

```
Sub AccesoFilasColumnas1()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oFil As Object
Dim oCol As Object

    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

    'Referencia al rango C5:E10
    oRango = oHojaActiva.getCellRangeByName( "C5:E10" )
    'Mostramos el tipo de objeto
    MsgBox oRango.getImplementationName()
    'Creamos una nueva referencia a las filas y columnas
    oCol = oRango.getColumns()
    oFil = oRango.getRows()

    'Verificamos el tipo de objeto
    MsgBox oCol.getImplementationName() & " - " & oFil.getImplementationName()

End Sub
```

Los métodos getColumns y getRows, siempre (aunque en el rango exista solo una fila o columna) te devolverá un conjunto de Columnas y Filas, para poder tener acceso a los métodos y propiedades disponibles en los rangos, así como para poder hacer selecciones,

tenemos que acceder a cada fila o columna de forma individual o construir un contenedor de rangos para acceder a varias filas o columnas, veamos los dos casos.

Para seleccionar la primer columna del rango.

```
Sub AccesoFilasColumnas2()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oCol As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Referencia al rango C5:E10
   oRango = oHojaActiva.getCellRangeByName( "C5:E10" )
   'Creamos una nueva referencia a la primer columna del rango la C
   oCol = oRango.getColumns.getByIndex(0)

   'Y lo seleccionamos
   ThisComponent.getCurrentController.select(oCol)

End Sub
```

Para seleccionar la primer fila del rango.

```
Sub AccesoFilasColumnas3()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oFil As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Referencia al rango C5:E10
   oRango = oHojaActiva.getCellRangeByName( "C5:E10" )
   'Creamos una nueva referencia a la segunda fila del rango la 6
   oFil = oRango.getRows.getByIndex(1)

   'Y lo seleccionamos
   ThisComponent.getCurrentController.select(oFil)

End Sub
```

Por supuesto, podemos seleccionar una fila y una columna al mismo tiempo.

```
Sub AccesoFilasColumnas4()
Dim oHojaActiva As Object
Dim oRangos As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Creamos el contender para los rangos
   oRangos = ThisComponent.createInstance("com.sun.star.sheet.SheetCellRanges")

   'Agregamos la columna E y la fila 10 al contenedor
   oRangos.addRangeAddress( oHojaActiva.getColumns().getByIndex(4).getRangeAddress(),False )
   oRangos.addRangeAddress( oHojaActiva.getRows().getByIndex(9).getRangeAddress(),False )

   'Comprobamos que están correctos seleccionándolos
   ThisComponent.getCurrentController.select(oRangos)

End Sub
```

Como ya notaste, podemos acceder directamente a la fila o columna que queramos de cualquier hoja por medio de la colección de estas (getColumns y getRows) y por medio del índice (getByIndex) a la fila o columna deseada.

```
    oHojaActiva.getColumns.getByIndex(4)    'Columna E

    oHojaActiva.getRows.getByIndex(9)       'Fila 10
```

Para seleccionar todas las columnas o filas del rango indicado, usamos.

```
Sub AccesoFilasColumnas5()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oRangos As Object
Dim oCol As Object
Dim co1 As Long

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Referencia al rango A2:A8
   oRango = oHojaActiva.getCellRangeByName( "A2:E8" )
   'Contenedor de rangos
   oRangos = ThisComponent.createInstance("com.sun.star.sheet.SheetCellRanges")

   For co1 = 0 To oRango.getColumns.getCount() - 1
         'Creamos una nueva referencia a cada columna
         oCol = oRango.getColumns.getByIndex(co1)
         'La agregamos al contenedor de rangos
         oRangos.addRangeAddress( oCol.getRangeAddress(),False )
   Next

   'Y lo seleccionamos
   ThisComponent.getCurrentController.select( oRangos )

End Sub
```

Con un poco de ingenio, puedes hacerte tus propias funciones que te devuelvan filas o columnas enteras, pero esa es tu tarea.

Por ahora hemos visto como hacer referencia a diferentes tipos de rangos, en capítulos posteriores, veremos como manipular estos rangos, mover, insertar, copiar, etc., para terminar este tema de hacer referencia a rangos, veamos uno muy importante, la selección actual.

6.2.5 Referencia a la selección actual

Trabajar con la selección actual, es decir, con lo que el usuario tenga seleccionado al momento de llamar a una macro, es una actividad muy común y también, muy propensa a errores (ya sabes como son los usuarios), trataremos de mantener el control de la selección, siempre que sea posible.

Para acceder a la selección actual dentro de nuestra hoja de calculo usamos el método getCurrentSelection() de la siguiente manera.

```
Sub SeleccionActual1()
Dim oSel As Object
```

```
    oSel = ThisComponent.getCurrentSelection()

    MsgBox oSel.getImplementationName()

End Sub
```

Pero cuidado, nuestra selección pueden ser muchas cosas, vuelve a ejecutar la macro anterior teniendo cada vez seleccionado los siguientes elementos:

1. Una celda
2. Un rango de celdas
3. Varios rangos de celdas
4. Un objeto de dibujo

Debes de obtener, si lo hiciste en el mismo orden que yo, la siguiente lista de mensajes:

1. ScCellObj
2. ScCellRangeObj
3. ScCellRangesObj
4. com.sun.star.drawing.SvxShapeCollection

Observa la diferencia entre una sola celda (1) y un rango (2 y 3), y nota también la diferencia en un rango (2) y varios (3), esto es muy importante para actuar en consecuencia por que son objetos diferentes y por lo tanto implementan métodos y propiedades diferentes, por ejemplo, vamos a tratar de ver el contenido de una celda, ejecuta la macro siguiente, teniendo seleccionado cada uno de los tres primeros rangos mencionados.

```
Sub SeleccionActual2()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   MsgBox oSel.getString()

End Sub
```

Si seleccionaste una sola celda y esta contenía algo, la macro anterior te tiene que mostrar ese contenido, en las dos siguientes selecciones, de un rango y varios rangos, te tuvo que haber dado el error “Propiedad o método no encontrado”, por que efectivamente, el método getString() solo esta implementado en celdas individuales, por ello es muy importante discriminar la selección que haya hecho el usuario, primer intento.

```
Sub SeleccionActual3()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   Select Case oSel.getImplementationName()
         Case "ScCellObj"
               MsgBox "Muy bien seleccionaste solo una celda"
         Case "ScCellRangeObj"
               MsgBox "Te pasaste un poco"
         Case "ScCellRangesObj"
               MsgBox "No tienes remedio"
         Case Else
               MsgBox "No se ni que seleccionaste"
   End Select

End Sub
```

Ahora ya sabemos que es lo que el usuario selecciono, pero dependerá de que queramos hacer con la selección para actuar en consecuencia, por ejemplo, supongamos que nuestra macro requiere que el usuario seleccione una y solo una celda, podemos solo informarle como en el siguiente ejemplo.

```
Sub SeleccionActual4()
Dim oSel As Object
Dim oCelda As Object

   oSel = ThisComponent.getCurrentSelection()

   Select Case oSel.getImplementationName()
         Case "ScCellObj"
               oCelda = oSel
               MsgBox oCelda.getString()
         Case Else
               MsgBox "Se requiere seleccionar solo UNA celda"
   End Select

End Sub
```

O podemos tomar solo la primer celda del rango o de los rangos que haya seleccionado como en.

```
Sub SeleccionActual5()
Dim oSel As Object
Dim oCelda As Object

   oSel = ThisComponent.getCurrentSelection()

   Select Case oSel.getImplementationName()
         Case "ScCellObj"
               oCelda = oSel
         Case "ScCellRangeObj"
               'Si es un solo rango, accedemos a la primer celda
               oCelda = oSel.getCellByPosition( 0,0 )
         Case "ScCellRangesObj"
               'Si son varios rangos, primero accedemos al primer rango
               'con getByIndex(0) y despues a la primer celda
               oCelda = oSel.getByIndex(0).getCellByPosition( 0,0 )
         Case Else
               MsgBox "Se requiere seleccionar una celda"
   End Select

   If Not IsNull(oCelda) Then
         MsgBox oCelda.getString()
   End If

End Sub
```

Nota que aun tenemos que evaluar que oCelda, apunte a un rango válido, recuerda que en la interfaz del usuario, puede haber más elementos seleccionables, como recomendación malévola, nunca confíes en el usuario, siempre válida sus datos. Es importante que recuerdes que en el caso de un rango, la primer celda siempre será la superior izquierda, y en un grupo de rangos, el rango 0 también siempre será el que este más arriba y a la izquierda, no importa si fue el ultimo rango en ser seleccionado. También, recuerda que la posición 0,0 en relación con la hoja siempre se refiere a la celda A1, pero en relación al rango seleccionado, puede ser cualquier otra.

En algunos casos, tal vez sea valido cualquiera de los tres tipos de rangos, es decir, que el método a usar esta implementado en los tres, por ejemplo.

```
Sub SeleccionActual6()
Dim oSel As Object
Dim oRango As Object

   oSel = ThisComponent.getCurrentSelection()

   Select Case oSel.getImplementationName()
         Case "ScCellObj", "ScCellRangeObj", "ScCellRangesObj"
               oRango = oSel
         Case Else
               MsgBox "Se requiere seleccionar un rango"
   End Select

   If Not IsNull(oRango) Then
         'Borramos el contenido de las celdas
         oRango.clearContents(31)
   End If

End Sub
```

El método clearContents, esta implementado en los tres tipos de rangos, por ello podemos llamarlo sin problemas, más adelante veremos a detalle este método.

Lo que he querido ilustrarte, es que, dependiendo de tus necesidades, tienes que discriminar una u otra cosa, pero siempre, y disculpa la necedad, siempre válida los datos.

6.2.6 Obteniendo información de rangos

En diversas ocasiones, es necesario saber donde estamos ubicados, es decir, en que hoja, fila y columna, los rangos cuentas con métodos y estructuras con esta información, algunas de ellas ya las hemos usado, también, muchos métodos de manipulación de rangos, requieren se les pasen las estructuras correctas de la dirección de los rangos, por ello es importante saber cuales son y como están estructuradas, veamos cuales son.

```
Sub InfoRangos1()
Dim oSel As Object
Dim oDir As Object
Dim sTmp As String
Dim col As Integer

   oSel = ThisComponent.getCurrentSelection()

   Select Case oSel.getImplementationName()
         Case "ScCellObj"
               'Obtenemos la dirección de la celda
               oDir = oSel.getCellAddress()
               'Mostramos sus datos, observa como hacemos uso del índice de la hoja
               'oDir.Sheet para obtener el nombre de la hoja
               MsgBox "Hoja: " & ThisComponent.getSheets().getByIndex(oDir.Sheet).getName() & Chr(13)
& "Columna: " & oDir.Column & Chr(13) & "Fila: " & oDir.Row

         Case "ScCellRangeObj"
               'Si es un solo rango, obtenemos sus datos
               oDir = oSel.getRangeAddress()
               'Construimos el texto informativo
               sTmp = "El rango esta en la hoja: " & oDir.Sheet & Chr(13) & _
                        "Columna Inicio: " & oDir.StartColumn & Chr(13) & _
```

```
                        "Fila Inicio: " & oDir.StartRow & Chr(13) & _
                        "Columna Fin: " & oDir.EndColumn & Chr(13) & _
                        "Fila Fin: " & oDir.EndRow
                MsgBox sTmp

        Case "ScCellRangesObj"
                'Si son varios rangos, podemos obtener los datos de cada rango
                For co1 = 0 To oSel.getCount()-1
                        oDir = oSel.getByIndex(co1).getRangeAddress()
                        sTmp = "El rango " & co1 & " esta en la hoja: " & oDir.Sheet & Chr(13) & _
                                "Columna Inicio: " & oDir.StartColumn & Chr(13) & _
                                "Fila Inicio: " & oDir.StartRow & Chr(13) & _
                                "Columna Fin: " & oDir.EndColumn & Chr(13) & _
                                "Fila Fin: " & oDir.EndRow
                        MsgBox sTmp
                Next
                'O podemos acceder a las direcciones de todos los rangos
                sTmp = Join( Split(oSel.getRangeAddressesAsString(),";"),Chr(13))
                'Mostramos el resultado
                MsgBox sTmp

        Case Else
                MsgBox "Se requiere seleccionar un rango de celdas"
    End Select

End Sub
```

Observa que si es una sola celda se tiene que usar getCellAddress que obtiene: la hoja donde esta la celda, la columna y fila, no se te olvide que todo empieza en 0, cuando es un rango de celdas se usa getRangeAddress, que contiene, la hoja por índice donde esta el rango, la columna y fila donde empieza y la columna y fila donde termina, reitero, todo desde 0. Observa como en el caso de varios rangos, podemos acceder a cada uno, pero estos si cuentan con una propiedad de texto (getRangeAddressesAsString) que directamente nos devuelve las direcciones de los rangos en un formato claro para “casi” por cualquier usuario, por ejemplo “Datos.A2:E5”. No he encontrado, desconozco si existe, una propiedad o método similar para los casos de una celda y un rango, pero podemos crearnos nuestra propia función que lo haga por nosotros.

```
Sub InfoRangos2()
Dim oSel As Object

    oSel = ThisComponent.getCurrentSelection()
    'Usamos nuestra función personalizada
    MsgBox DireccionRango(oSel)

End Sub


'Que linda quedo esta función
Function DireccionRango(Rango As Object) As String
Dim oSFA As Object
Dim oDir As Object
Dim mDatos

    'Nos apoyamos en la funcion de Calc DIRECCION (ADDRESS)
    oSFA = createUnoService( "com.sun.star.sheet.FunctionAccess" )
    'Validamos el tipo de rango pasado
    Select Case Rango.getImplementationName()
        Case "ScCellObj"
            'Si es una sola celda usamos getCellAddress
            oDir = Rango.getCellAddress
            'Construimos la matriz de datos para la función
            mDatos = Array( oDir.Row+1, oDir.Column+1,4,1,Rango.getSpreadsheet.getName() )
            DireccionRango = oSFA.callFunction("ADDRESS",mDatos())
```

```
        Case "ScCellRangeObj"
              'Si es un rango de celdas usamos getRangeAddress
              oDir = Rango.getRangeAddress()
              'Obtenemos la celda superior izquierda del rango
              mDatos =Array(oDir.StartRow+1,oDir.StartColumn+1,4,1,Rango.getSpreadsheet.getName())
              DireccionRango = oSFA.callFunction("ADDRESS",mDatos()) & ":"
              'Ahora la celda inferior derecha del rango
              mDatos = Array( oDir.EndRow+1, oDir.EndColumn+1, 4 )
              DireccionRango = DireccionRango & oSFA.callFunction("ADDRESS",mDatos())
        Case "ScCellRangesObj"
              'Esta ya la vimos
              DireccionRango = Join( Split(Rango.getRangeAddressesAsString(),";"),Chr(13))
   End Select

End Function
```

La función DIRECCION (*ADDRESS*) de Calc, nos devuelve en formato de texto, la referencia a la celda, indicándole la fila, columna, si queremos la referencia absoluta o relativa y la hoja, si usamos las funciones de Calc desde OOo Basic, el nombre de estas tenemos que indicarlas en ingles y pasarle los argumentos de la función en forma de matriz de datos, es muy importante pasarle, mínimo, los argumentos requeridos y estos, deben estar en el formato (string, integer, etc.) que “espera” la función, sino, tal y como sucede en la hoja de calculo, la función te retornara un error. Más adelante veremos un poco más detallado el uso de funciones incorporadas de la hoja de calculo, directamente en nuestras macros.

Una segunda versión de esta función, puede ser la siguiente, nota que no hacemos uso de ninguna función de hoja de calculo, sino que obtenemos los datos solo con propiedades del rango, puedes usar indistintamente cualquiera de las dos.

```
Function DireccionRango2(Rango As Object) As String
Dim sTmp As String

   Select Case Rango.getImplementationName()
        Case "ScCellObj"
              sTmp = Rango.getSpreadsheet.getName() & "." & _
                    Rango.getColumns().getByIndex(0).getName() & _
                    Rango.getCellAddress.Row + 1
        Case "ScCellRangeObj"
              sTmp = Rango.getSpreadsheet.getName() & "." & _
                    Rango.getColumns().getByIndex(0).getName() & _
                    Rango.getRangeAddress.StartRow + 1 & ":" & _
                    Rango.getColumns().getByIndex(Rango.getColumns().getCount()-1).getName() & _
                    Rango.getRangeAddress.EndRow + 1
        Case "ScCellRangesObj"
              sTmp = Join( Split(Rango.getRangeAddressesAsString(),";"),Chr(13) )
   End Select
   DireccionRango2 = sTmp

End Function
```

Por ultimo, podemos saber el número de filas y columnas que tiene un rango usando los métodos getRows y getColumns respectivamente, estos métodos solo están disponibles cuando el rango es una sola celda y un rango, cuando son varios rangos tienes que ingeniarteles que no es complicado.

```
Sub InfoRangos3()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
```

```
    Select Case oSel.getImplementationName()
          Case "ScCellObj","ScCellRangeObj"
                MsgBox "Filas = " & oSel.getRows().getCount() & Chr(13) & _
                      "Columnas = " & oSel.getColumns().getCount()
    End Select

End Sub
```

Los rangos, también, cuentan con una propiedad que nos devuelve la dirección del rango seleccionado, se llama AbsoluteName y el resultado como su nombre lo indica es con las referencias absolutas.

```
Sub InfoRangos4()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   MsgBox oSel.AbsoluteName

End Sub
```

6.2.7 Trabajando con Cursores

Un cursor, es un objeto que nos permite movernos por la hoja entera o dentro de un rango especificado sin afectar o modificar al rango a partir del cual se crea. Lo más usual es crearlo a partir de un rango para desplazarse dentro de el, pero además tiene métodos muy útiles para expandir por ejemplo a la región actual, que como sabes, es el rango de celdas “con datos” delimitado por filas y columnas en blanco, la región actual es un concepto sumamente, útil y poderoso cuando se programa, por ello le daremos un poco de énfasis, supongamos la siguiente tabla de datos.

	A	B	C	D	E	F	G
1							
2		0.41	0.39	0.73	0.76	0.08	
3		0.31	0.21	0.12	0.91	0.27	
4		0.3	0.99	0.61	0.25	0.61	
5		0.18	0.8	0.83	0.12	0.01	
6		0.26	0.73	0.45	0.14	0.05	
7		0.72	0.97	0.88	0.48	0.71	
8		0.65	0.89	0.1	0.38	0.65	
9		0.19	0.69	0.86	0.3	0.6	
10		0.13	0.6	0.6	0.73	0.85	
11		0.21	0.91	0.65	0.04	0.04	
12							
13							

Observa que el cursor esta en la celda D8 y que la región actual de este rango de datos es B2:F11, ahora, ejecuta la siguiente macro donde obligamos al usuario a seleccionar solo una celda, lo cual sabes, no necesariamente es así y solo es para fines didácticos, nota que usaremos una función creada en tema anterior que se llama DireccionRango, pero muy importante, cuando le pasemos como argumento una celda, esta trabajara bien, pero cuando intentemos pasarle un cursor, al ser un objeto diferente, te devolverá una cadena vacía, para solucionar esto, tenemos que agregar el tipo de objeto "ScCellCursorObj", busca y corrige la siguiente línea en dicha función personalizada.

```
Case "ScCellRangeObj", "ScCellCursorObj"
```

Observa que ahora evaluara si es un rango de celdas o un cursor, ahora si, ejecuta la siguiente macro.

```
Sub Cursores1()
Dim oSel As Object
Dim oCursor As Object

   'Partimos de la selección
   oSel = ThisComponent.getCurrentSelection()
   'Solo si es una sola celda
   If oSel.getImplementationName() = "ScCellObj" Then
         'Mostramos la direccion de la celda seleccionada
         MsgBox DireccionRango( oSel )

         'Creamos un cursor a partir de esta celda
         oCursor = oSel.getSpreadSheet().createCursorByRange( oSel )
         'Verificamos que apunten a la misma celda
         MsgBox DireccionRango( oCursor )

         'Nos movemos al inicio de la region actual
         oCursor.gotoStart()
         MsgBox DireccionRango( oCursor )

         'Nos movemos al final
         oCursor.gotoEnd()
         MsgBox DireccionRango( oCursor )

         'Expandimos el cursor a toda la región actual
         oCursor.collapseToCurrentRegion()
         MsgBox DireccionRango( oCursor )

         'Mostramos de nuevo la dirección de la celda seleccionada
         'observa que esta no se a movido
         MsgBox DireccionRango( oSel )
   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Los cursores, al compartir la mayoría de los servicios de las celdas y los rangos, heredan la mayoría de sus métodos y propiedades, pero sus métodos particulares son lo que los hacen especiales e interesantes. Una pregunta recurrente en las listas de correo, es ¿como encuentro la siguiente fila libre?, con un cursor, esto es muy sencillo.

```
Sub Cursores2()
Dim oSel As Object
Dim oCursor As Object

   'Partimos de la selección
   oSel = ThisComponent.getCurrentSelection()
   'Solo si es una sola celda
   If oSel.getImplementationName() = "ScCellObj" Then
         'Creamos un cursor a partir de esta celda
         oCursor = oSel.getSpreadSheet().createCursorByRange( oSel )
         'Nos movemos al final
         oCursor.gotoEnd()
         'La siguiente fila libre, sera esta fila mas 1
         MsgBox oCursor.getRangeAddress().EndRow + 1
   Else
```

```
        MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Por supuesto, algo muy practico es tener una función personalizada que nos encuentre la siguiente fila libre, algo como.

```
Function FilaLibre( Rango As Object) As Long
Dim oCursor As Object

   oCursor = Rango.getSpreadSheet.createCursorByRange( Rango )
   oCursor.gotoEnd()
   FilaLibre = oCursor.getRangeAddress.EndRow + 1

End Function
```

Y que usamos de una forma muy sencilla.

```
Sub Cursores3()
Dim oSel As Object
Dim oCursor As Object

   'Partimos de la selección
   oSel = ThisComponent.getCurrentSelection()

   'Solo funciona con celdas individuales o rangos de celdas
   Select Case oSel.getImplementationName()
        Case "ScCellObj", "ScCellRangeObj"
             MsgBox "La siguiente fila libre es la: " & FilaLibre( oSel )
   End Select

End Sub
```

Otros métodos con que cuentan los cursores son.

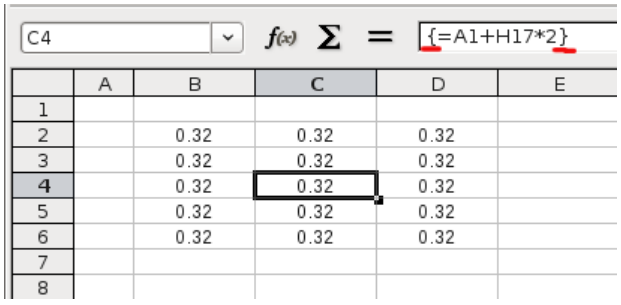
```
Sub Cursores4()
Dim oSel As Object
Dim oCursor As Object

   oSel = ThisComponent.getCurrentSelection()
   If oSel.getImplementationName() = "ScCellObj" Then

        oCursor = oSel.getSpreadSheet().createCursorByRange( oSel )

        'Nos movemos a la siguiente celda
        oCursor.gotoNext()
        MsgBox DireccionRango( oCursor )

        'Nos movemos a la celda anterior
        oCursor.gotoPrevious()
        MsgBox DireccionRango( oCursor )

        'Nos movemos dos columnas a la derecha y tres filas abajo
        oCursor.gotoOffset(2,3)
        MsgBox DireccionRango( oCursor )

   Else
        MsgBox "Selecciona solo una celda"
   End If
```

```
End Sub
```

Estos métodos hay que usarlos con cuidado, pues puedes “salirte” del rango donde estas trabajando y ya no podrás moverte con otros métodos como gotoStart o gotoEnd, por ejemplo, si te mueves al final de un rango (gotoEnd) y después te desplazas con gotoOffset(3,3), o sea tres columnas a la derecha y tres filas abajo, te saldrás del rango y ya no podrás regresar al inicio (gotoStart) como lo demostramos en el siguiente ejemplo, usa estos métodos con precaución.

```
Sub Cursores5()
Dim oSel As Object
Dim oCursor As Object

   oSel = ThisComponent.getCurrentSelection()
   If oSel.getImplementationName() = "ScCellObj" Then

         oCursor = oSel.getSpreadSheet().createCursorByRange( oSel )

         oCursor.gotoStart()
         'Toma nota de esta dirección
         MsgBox DireccionRango( oCursor )

         oCursor.gotoEnd()
         MsgBox DireccionRango( oCursor )

         'Nos movemos tres columnas a la derecha y tres filas abajo
         oCursor.gotoOffset(3,3)
         MsgBox DireccionRango( oCursor )

         'Intentamos regresar al inicio, nota como NO es la
         'misma de la primer linea
         oCursor.gotoStart()
         MsgBox DireccionRango( oCursor )

   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

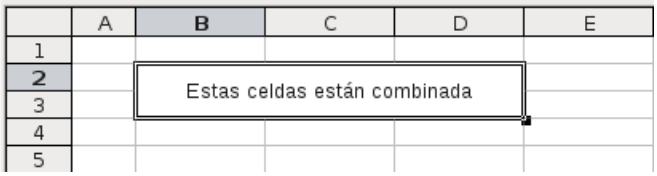
Observa atentamente la siguiente imagen, nota en la barra de formulas que la formula mostrada esta encerrada en unas llaves, como sabes, estas llaves nos indican que dicha formula forma parte de una formula matricial, como no es el tema de este libro las formulas matriciales no entraremos en detalle alguno, pero como sabes, estas se introducen con la combinación de teclas CTRL+SHIFT+ENTER y que ya creada, no puedes modificar solo una celda de dichas matrices, sino tienes que hacerlo con toda la matriz.



Con un cursor, podemos saber cual es el rango de una formula matricial, como lo demostramos en el siguiente ejemplo, probado en los datos mostrados en la imagen anterior.

```
Sub Cursores6()
Dim oSel As Object
Dim oCursor As Object

   oSel = ThisComponent.getCurrentSelection()
   If oSel.getImplementationName() = "ScCellObj" Then

         oCursor = oSel.getSpreadSheet.createCursorByRange( oSel )
         'Expandimos el cursor a la matriz actual
         oCursor.collapseToCurrentArray()
         MsgBox DireccionRango( oCursor )

   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

En la siguiente imagen, observa las celdas combinadas, cuando establecemos el cursor en celdas combinadas y tratamos de obtener dicha dirección, solo te devolverá la celda superior izquierda de dicho rango, pero con un cursor podemos saber el rango completo de estas celdas combinadas.




```
Sub Cursores7()
Dim oSel As Object
Dim oCursor As Object

   oSel = ThisComponent.getCurrentSelection()
   If oSel.getImplementationName() = "ScCellObj" Then

         oCursor = oSel.getSpreadSheet().createCursorByRange( oSel )

         'Comprobamos que solo devuelve una celda
         MsgBox DireccionRango( oSel )
         'Expandimos el cursor a todas las celdas combinadas
         oCursor.collapseToMergedArea()
         MsgBox DireccionRango( oCursor )

   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Podemos expandir el cursor para que abarque todas las columnas de la región actual.

```
Sub Cursores8()
Dim oSel As Object
Dim oCursor As Object

   oSel = ThisComponent.getCurrentSelection()
   If oSel.getImplementationName() = "ScCellObj" Then
```

```
        oCursor = oSel.getSpreadSheet.createCursorByRange( oSel )

        'Expandimos el cursor a la región actual
        oCursor.collapseToCurrentRegion()
        MsgBox DireccionRango( oCursor )

        'Y luego a todas las columnas
        oCursor.expandToEntireColumns()
        MsgBox DireccionRango( oCursor )

    Else
        MsgBox "Selecciona solo una celda"
    End If

End Sub
```

Intuirás que también es posible hacerlo en las filas.

```
        'Expandimos el cursor a la región actual
        oCursor.collapseToCurrentRegion()
        MsgBox DireccionRango( oCursor )

        'Y luego a todas las columnas
        oCursor.expandToEntireRows()
        MsgBox DireccionRango( oCursor )
```

Ten cuidado cuando uses los dos métodos anteriores, si los ejecutas uno detrás de otro, tendrás un cursor con tu hoja completa, que no descartes usar un día, lo importante es que siempre tengas control y sepas lo que estas haciendo.

```
Sub Cursores10()
Dim oSel As Object
Dim oCursor As Object

    oSel = ThisComponent.getCurrentSelection()
    If oSel.getImplementationName() = "ScCellObj" Then

        oCursor = oSel.getSpreadSheet().createCursorByRange( oSel )

        'Expandimos el cursor a la region actual
        oCursor.collapseToCurrentRegion()
        MsgBox DireccionRango( oCursor )

        'Y luego a todas las filas
        oCursor.expandToEntireRows()
        MsgBox DireccionRango( oCursor )

        'Nota como devuelve la hoja entera
        oCursor.expandToEntireColumns()
        MsgBox DireccionRango( oCursor )

    Else
        MsgBox "Selecciona solo una celda"
    End If

End Sub
```

También puedes ampliar tu cursor el número de columnas y filas que necesites, toma en cuenta que la celda superior izquierda no cambia.

```
Sub Cursores11()
Dim oSel As Object
Dim oCursor As Object

   oSel = ThisComponent.getCurrentSelection()
   If oSel.getImplementationName() = "ScCellObj" Then

         oCursor = oSel.getSpreadSheet().createCursorByRange( oSel )

         'Expandimos el cursor a la región actual
         oCursor.collapseToCurrentRegion()
         MsgBox DireccionRango( oCursor )

         'Y luego cambiamos el cursor a 2 columnas y 5 filas
         oCursor.collapseToSize( 2,5 )
         MsgBox DireccionRango( oCursor )

   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Nota que estamos “cambiando” el tamaño del cursor, si lo que quieres realmente es expandirlo, tienes que sumarle el ancho y alto de la región actual, como en el siguiente ejemplo.

```
Sub Cursores12()
Dim oSel As Object
Dim oCursor As Object

   oSel = ThisComponent.getCurrentSelection()
   If oSel.getImplementationName() = "ScCellObj" Then

         oCursor = oSel.getSpreadSheet.createCursorByRange( oSel )

         'Expandimos el cursor a la región actual
         oCursor.collapseToCurrentRegion()
         MsgBox DireccionRango( oCursor )

         'Y luego ampliamos el cursor 1 columna y 5 filas
         oCursor.collapseToSize( oCursor.getColumns.getCount()+1, oCursor.getRows.getCount()+5 )
         MsgBox DireccionRango( oCursor )

   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Nota como hemos usado algunos métodos vistos en el tema anterior (getColumns.getCount), como ya lo mencionamos, esto es posible por que los cursores comparten la mayoría de los servicios con los rangos. Casi para finalizar, veamos dos métodos más de los cursores.

```
Sub Cursores13()
Dim oSel As Object
Dim oCursor As Object

   oSel = ThisComponent.getCurrentSelection()
   If oSel.getImplementationName() = "ScCellObj" Then

         oCursor = oSel.getSpreadSheet().createCursorByRange( oSel )
```

```
        'Vamos al inicio del área usada, el argumento False, evita que se
        'expanda la selección del rango
        oCursor.gotoStartOfUsedArea( False )
        MsgBox DireccionRango( oCursor )

        'Ahora vamos al final
        oCursor.gotoEndOfUsedArea( False )
        MsgBox DireccionRango( oCursor )
   Else
        MsgBox "Selecciona solo una celda"
   End If

End Sub
```

En una hoja de calculo, el “área de usuario”, es aquel rango “efectivamente” usado, es muy dinámico pues varia de acuerdo a su uso y eliminación, en una hoja nueva el inicio y el final de esta área siempre será el mismo, la celda A1, si tienes “una” sola celda usada, esta, será el inicio y el fin de tu área de usuario, la cosa cambia en cuanto tienes dos o más celdas usadas, la regla es: el inicio de tu área de usuario será la celda (“con datos”) que este más cercana al extremo superior izquierdo de la hoja y el final será la celda (“con datos”) que este más cercana al extremo inferior derecho de dicha hoja, y digo que es dinámica, por que si eliminas datos o celdas, esta área se ira ajustando conforme a estos cambios.

Ahora si, para terminar, un cursor también lo puedes usar para establecer una selección, como en el ejemplo siguiente donde seleccionamos la región actual del rango.

```
Sub Cursores14()
Dim oSel As Object
Dim oCursor As Object

   oSel = ThisComponent.getCurrentSelection()
   If oSel.getImplementationName() = "ScCellObj" Then

        oCursor = oSel.getSpreadSheet().createCursorByRange( oSel )

        oCursor.collapseToCurrentRegion()
        ThisComponent.getCurrentController.select( oCursor )

   Else
        MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Con lo cual observamos que estos cursores con bastante versátiles.

6.2.8 Selecciones especiales

Los rangos cuentan con unos métodos bastante interesantes (y útiles) para hacer algunas selecciones especiales, por ejemplo, podemos seleccionar las celdas vacías de un rango como en.

```
Sub SeleccionesEspeciales1()
Dim oSel As Object
```

```
Dim oSelEsp As Object

   'Accedemos a la selección actual
   oSel = ThisComponent.getCurrentSelection()
   'Recuperamos las celdas vacías
   oSelEsp = oSel.queryEmptyCells()
   'Y las seleccionamos
   ThisComponent.getCurrentController.select( oSelEsp )
End Sub
```

Como hemos visto a lo largo de estas notas, **no es siempre necesario que selecciones, puedes manipular el rango obtenido sin que esto sea necesario**, si una celda tiene aunque sea solo un espacio o algún carácter especial no visible (tabuladores, saltos de página), esta, ya no será vacía y no la obtendrás por este método. Un punto importante, es que este método siempre retorna un conjunto de rangos (*ScCellRangesObj*), los cuales tienes que manipular como ya lo hemos aprendido.

También podemos seleccionar celdas de acuerdo a su contenido, en el siguiente ejemplo seleccionamos las celdas con texto.

```
Sub SeleccionesEspeciales2()
Dim oSel As Object
Dim oSelEsp As Object

   'Accedemos a la seleccion actual
   oSel = ThisComponent.getCurrentSelection()
   'Recuperamos las celdas con texto
   oSelEsp = oSel.queryContentCells( 4 )
   'Y las seleccionamos
   ThisComponent.getCurrentController.select( oSelEsp )

End Sub
```

Ahora, las celdas con formulas.

```
Sub SeleccionesEspeciales3()
Dim oSel As Object
Dim oSelEsp As Object

   oSel = ThisComponent.getCurrentSelection()
   'Recuperamos las celdas con formulas
   oSelEsp = oSel.queryContentCells( com.sun.star.sheet.CellFlags.FORMULA )
   ThisComponent.getCurrentController.select( oSelEsp )

End Sub
```

Los valores posibles para el argumento de este método son los siguientes, valores que también se usan para borrar datos como se ve en el tema 6.4.3.Borrando datos

Constante	*Valor*
com.sun.star.sheet.CellFlags.VALUE	1
com.sun.star.sheet.CellFlags.DATETIME	2
com.sun.star.sheet.CellFlags.STRING	4
com.sun.star.sheet.CellFlags.ANNOTATION	8
com.sun.star.sheet.CellFlags.FORMULA	16
com.sun.star.sheet.CellFlags.HARDATTR	32

Constante	*Valor*
com.sun.star.sheet.CellFlags.STYLES	64
com.sun.star.sheet.CellFlags.OBJECT	128
com.sun.star.sheet.CellFlags.EDITATTR	256
com.sun.star.sheet.CellFlags.FORMATTED	512

Puedes usar de forma indistinta el valor numérico o su constante como se ve en los dos ejemplos anteriores y también sumar los valores para tener múltiples combinaciones, como en el ejemplo siguiente donde seleccionamos las celdas con texto y valores.

```
Sub SeleccionesEspeciales4()
Dim oSel As Object
Dim oSelEsp As Object

   oSel = ThisComponent.getCurrentSelection()
   'Recuperamos las celdas con texto y valores
   oSelEsp = oSel.queryContentCells( 5 )
   ThisComponent.getCurrentController.select( oSelEsp )

End Sub
```

Como ya se mencionó, estos métodos siempre devuelven un conjunto de rangos (*ScCellRangesObj*), por lo que es muy sencillo saber si hay o no resultados, en el siguiente ejemplo, ya no seleccionamos, pero informamos si hubo o no celdas con el criterio especificado, para este ejemplo, las celdas que tengan notas, observa que usamos una función personalizada (*DireccionRango2*), que ya hemos usado anteriormente.

```
Sub SeleccionesEspeciales5()
Dim oSel As Object
Dim oSelEsp As Object

   oSel = ThisComponent.getCurrentSelection()
   'Recuperamos las celdas con notas
   oSelEsp = oSel.queryContentCells( 8 )
   If oSelEsp.getCount = 0 Then
         MsgBox "No hay celdas con notas"
   Else
         'Mostramos la dirección de los rangos encontrados
         MsgBox DireccionRango2( oSelEsp )
   End If

End Sub
```

Ya vimos como seleccionar formulas, pero también podemos seleccionar formulas de acuerdo a su resultado, por ejemplo, seleccionemos las celdas que contengan error.

```
Sub SeleccionesEspeciales6()
Dim oSel As Object
Dim oSelEsp As Object

   oSel = ThisComponent.getCurrentSelection()
   'Recuperamos las celdas con formulas cuyo resultado sea error
   oSelEsp = oSel.queryFormulaCells( 4 )
   If oSelEsp.getCount = 0 Then
         MsgBox "No hay celdas con errores"
   Else
         MsgBox DireccionRango2( oSelEsp )
   End If
```

```
End Sub
```

El argumento de este método, solo acepta los tres valores siguiente.

Constante	*Valor*
com.sun.star.sheet.FormulaResult.VALUE	1
com.sun.star.sheet.FormulaResult.STRING	2
com.sun.star.sheet.FormulaResult.ERROR	4

Los cuales, podemos combinar.

```
Sub SeleccionesEspeciales7()
Dim oSel As Object
Dim oSelEsp As Object

   oSel = ThisComponent.getCurrentSelection()
   'Recuperamos las celdas con formulas cuyo resultado sea texto o valor
   oSelEsp = oSel.queryFormulaCells( 3 )
   If oSelEsp.getCount = 0 Then
         MsgBox "No hay celdas con texto o valor"
   Else
         MsgBox DireccionRango2( oSelEsp )
   End If

End Sub
```

Los siguiente métodos requieren una atención especial, supongamos la siguiente lista de datos.

	A	**B**	**C**	**D**	**E**
1	3	3	5	3	4
2	5	2	1	2	4
3	3	1	2	5	5
4	1	1	3	4	5
5	1	5	5	5	4
6	4	1	4	1	3
7	1	3	1	1	5
8	5	1	1	5	3
9	5	3	3	2	3
10	3	2	4	5	1

Selecciona el rango A1:A10 y ejecuta la siguiente macro:

```
Sub SeleccionesEspeciales8()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oSelEsp As Object
Dim oCelda As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   'El rango a evaluar
   oRango = oHojaActiva.getCellRangeByName("A1:A10")
   'La celda de comparación
```

```
    oCelda = oHojaActiva.getCellRangeByName("A1")
    'Buscamos las celdas diferentes por columna
    oSelEsp = oRango.queryColumnDifferences( oCelda.getCellAddress )
    If oSelEsp.getCount = 0 Then
          MsgBox "No se encontraron celdas"
    Else
          'Las seleccionamos
          ThisComponent.getCurrentController.select( oSelEsp )
    End If

End Sub
```

Que te tiene que seleccionar.

	A	B	C	D	E
1	3	3	5	3	4
2	5	2	1	2	4
3	3	1	2	5	5
4	1	1	3	4	5
5	1	5	5	5	4
6	4	1	4	1	3
7	1	3	1	1	5
8	5	1	1	5	3
9	5	3	3	2	3
10	3	2	4	5	1

Observa como seleccionó las celdas con un valor diferente al de la celda de comparación, es muy importante entender que, de la celda de comparación, **"únicamente toma la fila"** de la dirección para la comparación, intenta pasarle a esta misma macro la celda C1, observa el resultado, después pasale la celda C2 y observa el resultado. Con el siguiente ejemplo, creo, queda bastante claro como trabaja este método, observa el rango pasado y la celda de comparación, el valor comparado es el valor de la fila respectiva para cada columna.

```
Sub SeleccionesEspeciales9()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oSelEsp As Object
Dim oCelda As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A1:C10")
   oCelda = oHojaActiva.getCellRangeByName("B10")
   oSelEsp = oRango.queryColumnDifferences( oCelda.getCellAddress )
   If oSelEsp.getCount = 0 Then
         MsgBox "No se encontraron celdas"
   Else
         ThisComponent.getCurrentController.select( oSelEsp )
   End If

End Sub
```

Que nos devuelve.

	A	B	C	D	E
1	3	3	5	3	4
2	5	2	1	2	4
3	3	1	2	5	5
4	1	1	3	4	5
5	1	5	5	5	4
6	4	1	4	1	3
7	1	3	1	1	5
8	5	1	1	5	3
9	5	3	3	2	3
10	3	2	4	5	1

Por ultimo, ejecuta la macro, llamando al rango completo de pruebas, con la celda de comparación que quieras, como en.

```
Sub SeleccionesEspeciales10()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oSelEsp As Object
Dim oCelda As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A1:E10")
   oCelda = oHojaActiva.getCellRangeByName("A3")
   oSelEsp = oRango.queryColumnDifferences( oCelda.getCellAddress )
   If oSelEsp.getCount = 0 Then
         MsgBox "No se encontraron celdas"
   Else
         ThisComponent.getCurrentController.select( oSelEsp )
   End If

End Sub
```

Y comprueba que es correcto el resultado. El mismo tipo de comparación lo podemos hacer pero por filas, como en el siguiente ejemplo, usando la misma tabla de datos.

```
Sub SeleccionesEspeciales11()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oSelEsp As Object
Dim oCelda As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A10:E10")
   oCelda = oHojaActiva.getCellRangeByName("C10")
   'Buscamos las celdas diferentes por filas
   oSelEsp = oRango.queryRowDifferences( oCelda.getCellAddress )
   If oSelEsp.getCount = 0 Then
         MsgBox "No se encontraron celdas"
   Else
         ThisComponent.getCurrentController.select( oSelEsp )
   End If

End Sub
```

Y el resultado.

	A	B	C	D	E
1	3	3	5	3	4
2	5	2	1	2	4
3	3	1	2	5	5
4	1	1	3	4	5
5	1	5	5	5	4
6	4	1	4	1	3
7	1	3	1	1	5
8	5	1	1	5	3
9	5	3	3	2	3
10	3	2	4	5	1

La lógica es exactamente la misma de hacerlo por columnas, solo que ahora, **"solo toma la columna"** de la celda de comparación, así que solo hagamos un ejemplo más con el rango completo de datos.

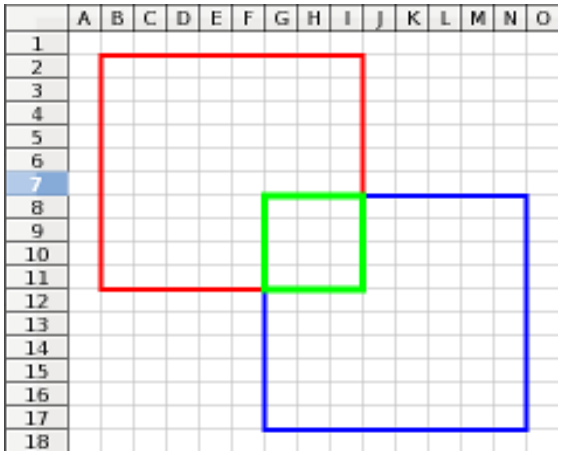
```
Sub SeleccionesEspeciales12()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oSelEsp As Object
Dim oCelda As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A1:E10")
   oCelda = oHojaActiva.getCellRangeByName("D1")
   'Buscamos las celdas diferentes por filas
   oSelEsp = oRango.queryRowDifferences( oCelda.getCellAddress )
   If oSelEsp.getCount = 0 Then
         MsgBox "No se encontraron celdas"
   Else
         ThisComponent.getCurrentController.select( oSelEsp )
   End If

End Sub
```

Que nos selecciona.

	A	B	C	D	E
1	3	3	5	3	4
2	5	2	1	2	4
3	3	1	2	5	5
4	1	1	3	4	5
5	1	5	5	5	4
6	4	1	4	1	3
7	1	3	1	1	5
8	5	1	1	5	3
9	5	3	3	2	3
10	3	2	4	5	1

Lo interesante de estos métodos, es que los valores de las celdas, no tienen que ser necesariamente texto, puede ser cualquier tipo de dato, incluyendo formulas, en cuyo caso, se tomará el tipo de resultado de esta.

El siguiente método nos sirve para saber el rango de celdas comunes a dos rangos, observa la siguiente imagen, la macro de ejemplo, nos tiene que seleccionar el rango en verde.



```
Sub SeleccionesEspeciales13()
Dim oHojaActiva As Object
Dim oRango1 As Object
Dim oRango2 As Object
Dim oSelEsp As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango1 = oHojaActiva.getCellRangeByName("B2:I11")
   oRango2 = oHojaActiva.getCellRangeByName("G8:N17")
   'Buscamos las celdas comunes
   oSelEsp = oRango1.queryIntersection( oRango2.getRangeAddress )
   If oSelEsp.getCount = 0 Then
         MsgBox "No hay nada en común"
   Else
         ThisComponent.getCurrentController.select( oSelEsp )
   End If

End Sub
```

Para terminar este tema, veamos un método muy interesante, nos permite seleccionar, del rango invocado, solo las celdas visibles. Observa la siguiente imagen.

	A	B	C	D	E	F
1	N°	Titulo	Director	Genero	Año	País
6	110	Al otro lado	Gustavo Loza	Drama	2005	México
10	17	Amores Perros	Alejandro González Iñárritu	Drama	2000	México
11	118	Ánimas Trujano (El hombre impo	Ismael Rodríguez	Drama	1962	México
17	50	Bajo California: el límite del tiem	Carlos Bolado	Drama	1999	México
20	52	Batalla en el cielo	Carlos Reygadas	Drama / Erotismo	2005	México
30	182	Cilantro y perejil	Rafael Montero	Comedia / Roman	1996	México
32	26	Como agua para chocolate	Alfonso Arau	Melodrama / Rom	1992	México
33	222	Cortometraje V-6 – Más que un i	Varios	Varios	2005	México
39	101	De la calle	Gerardo Tort	Drama	2001	México
44	89	Digna – Hasta el ultimo aliento	Felipe Cazals	Documental	2004	México
48	145	El	Luis Buñuel	Drama	1953	México
53	102	El Ceniciento	Gilberto Martínez	Comedia	1952	México
77	119	El mago	Jaime Aparicio	Drama	2004	México
87	16	El Tigre de Santa Julia	Alejandro Gamboa	Acción / Drama	2002	México
90	21	El topo	Alejandro Jodorowsky	Western / Fantasía	1970	México
95	144	Ensayo de un crimen	Luis Buñuel	Drama	1955	México
99	20	Fando y Lis	Alejandro Jodorowsky	Drama / Fantasía	1968	México
110	48	Hijas de su madres Las Buenros	Busi Cortés	Comedia / Drama	2005	México
115	51	Japón	Carlos Reygadas	Drama	2002	México
138	22	La montaña sagrada	Alejandro Jodorowsky	Drama / Fantasía	1973	México
158	132	Los albañiles	Jorge Fons	Drama	1976	México
163	85	Los ladrones viejos	Everardo González	Documental	2007	México
167	143	Los olvidados	Luis Buñuel	Drama	1950	México
186	242	Morricone dirige a Morricone		Musical	2005	México
191	146	Nazarín	Luis Buñuel	Drama	1958	México
209	31	Sangre	Amat Escalante	Drama	2005	México
218	90	Temporada de patos	Fernando Eimbcke	Comedia / Drama	2004	México
241	28	Y tu mamá también	Alfonso Cuarón	Comedia / Drama	2001	México

Nota en los encabezados de fila que tenemos muchas filas ocultas, con la siguiente macro, mostramos la dirección de los rangos visibles.

```
Sub SeleccionesEspeciales14()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oVisibles As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A1:G243")
   'Buscamos las celdas visibles
   oVisibles = oRango.queryVisibleCells()
   If oVisibles.getCount = 0 Then
         MsgBox "No hay celdas ocultas"
   Else
         'Mostramos la dirección de los rangos visibles
         MsgBox DireccionRango2( oVisibles )
   End If

End Sub
```

Cuando haces un filtro automático o especial y cuando calculas subtotales e intentas hacer operaciones de copiado, la copia resultante, solo contendrá las celdas visibles, cuando agrupas u ocultas filas o columnas manualmente, la copia incluirá aun las celdas ocultas, pero con este método siempre obtendrás las celdas visibles, que, en conjunto con los conocimientos del siguiente capitulo, puedes hacer tu propia versión de copiar solo las filas visibles.

6.3 Manipulando rangos

En el capitulo anterior aprendimos a referenciar cualquier rango que nos interese, ahora, aprenderemos a manipular estos rangos, a moverlos, insertarlos, eliminarlos y copiarlos.

6.3.1 Moviendo rangos

Para mover un rango, usamos el método moveRange de las hojas de calculo, este método requiere de dos argumentos, la celda superior izquierda (CellAddress) destino donde se moverá el rango origen (CellRangeAddress)

```
Hoja.moveRange( Destino As CellAddress, Origen As CellRangeAddress )
```

El siguiente ejemplo mueve el rango A1:B5 de la hoja activa, a la celda D10 de la misma hoja.

```
Sub MoverRangos1()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   'Rango a mover
   oOrigen = oHojaActiva.getCellRangeByName( "A1:B5" )
   'Celda destino
   oDestino = oHojaActiva.getCellRangeByName( "D10" )

   'Movemos el rango
   oHojaActiva.moveRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )

End Sub
```

Toma en cuenta que el rango origen “tiene” que ser un rango de celdas, que la celda destino “tiene” que ser una celda individual, que los datos del rango origen se borran y los datos del rango destino son “**reemplazados sin preguntarte**” por los datos origen y muy importante, si el rango contiene formulas, estas, “**no se actualizan a la nueva posición**” aun y cuando las referencias sean relativas, también, el rango origen cambia para adaptarse al rango destino, es decir, la referencia a dicho rango se actualizara automáticamente como lo puedes comprobar si muestras la dirección del rango origen, antes y después de moverse.

```
   MsgBox DireccionRango( oOrigen )

   'Movemos el rango
   oHojaActiva.moveRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )

   MsgBox DireccionRango( oOrigen )
```

Recuerda que la función DireccionRango, es personalizada y la hemos usado en los últimos temas vistos. Cuando usas getCellRangeByName, aun y cuando solo hagas referencia a una celda, puedes tener acceso a su propiedad geCellAddress, por lo que no tienes problemas en mover solo una celda como se ve en el siguiente ejemplo.

```
Sub MoverRangos2()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   'Nota como hacemos la referencia para que sea un rango
   oOrigen = oHojaActiva.getCellRangeByName( "B2" )
   'Esta tiene que seguir siendo una sola celda
   oDestino = oHojaActiva.getCellRangeByName( "E5" )

   'Movemos el rango
   oHojaActiva.moveRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )

End Sub
```

Por supuesto podemos validar que por lo menos el origen y el destino sean efectivamente los argumentos que necesita este método.

```
Sub MoverRangos3()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oOrigen = oHojaActiva.getCellRangeByName( "C5" )
   oDestino = oHojaActiva.getCellRangeByName( "F15" )

   'Validamos que los rangos sean correctos
   If oOrigen.getImplementationName() = "ScCellRangeObj" And oDestino.getImplementationName() =
"ScCellObj" Then
         oHojaActiva.moveRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )
   Else
         MsgBox "Los rangos no son correctos"
   End If

End Sub
```

Esta validación es más útil cuando tomamos el rango a mover a partir de la selección actual del usuario, para que funcione el siguiente código, tienes que seleccionar más de una celda es decir, un rango de celdas, por supuesto, te queda de tarea hacer que funcione, aun y con solo seleccionar una celda.

```
Sub MoverRangos4()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oOrigen = ThisComponent.getCurrentSelection()
   'Validamos que los rangos sean correctos
   If oOrigen.getImplementationName() = "ScCellRangeObj" Then
         'Dejamos una columna y una fila en blanco
         oDestino = oHojaActiva.getCellByPosition( oOrigen.getRangeAddress.EndColumn + 2,
oOrigen.getRangeAddress.EndRow + 2 )
         oHojaActiva.moveRange( oDestino.getCellAddress, oOrigen.getRangeAddress() )
   Else
         MsgBox "El rango Origen no es correcto"
   End If

End Sub
```

Nota como solo validamos el origen, por que el destino lo construimos a partir del este, observa que cuando mueves un rango, la selección actual no cambia, se queda en el rango origen, si seleccionamos el rango cuando se ha movido, podremos ir moviendo el rango, tantas veces como llames a la macro, modifica la macro anterior para que quede así.

```
Sub MoverRangos5()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oOrigen = ThisComponent.getCurrentSelection()
   If oOrigen.getImplementationName() = "ScCellRangeObj" Then
         oDestino = oHojaActiva.getCellByPosition( oOrigen.getRangeAddress().EndColumn + 2,
oOrigen.getRangeAddress().EndRow + 2 )
         oHojaActiva.moveRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )
         'Seleccionamos el nuevo origen
         ThisComponent.getCurrentController.select( oOrigen )
   Else
         MsgBox "El rango Origen no es correcto"
   End If

End Sub
```

Y llamala (ejecutala) varias veces, notaras como se va moviendo el rango, pero cuidado, puede pasar que en algún momento te un error, ¿cuando?, muy bien, cuando el rango destino quede fuera de la hoja de calculo, por lo que tenemos que evaluar también que esto no suceda.

```
Sub MoverRangos6()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object
Dim lLimiteCol As Long

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oOrigen = ThisComponent.getCurrentSelection()
   If oOrigen.getImplementationName() = "ScCellRangeObj" Then
         'Garantizamos que no sobre pase el máximo de columnas en la hoja de calculo
         lLimiteCol = oOrigen.getRangeAddress.EndColumn + 2 + oOrigen.getColumns.getCount()
         If lLimiteCol <= oHojaActiva.getColumns.getCount() Then
               oDestino = oHojaActiva.getCellByPosition( oOrigen.getRangeAddress.EndColumn + 2,
oOrigen.getRangeAddress.EndRow + 2 )
               oHojaActiva.moveRange( oDestino.getCellAddress, oOrigen.getRangeAddress() )
               ThisComponent.getCurrentController.select( oOrigen )
         Else
               MsgBox "Se llegó al limite de la hoja"
         End If
   Else
         MsgBox "Los rangos no son correctos"
   End If

End Sub
```

Nota que con oHojaActiva.getColumns().getCount(), obtenemos el total de columnas de la hoja activa, no importa si esta tiene 256 como en Openoffice.org 2.x o 1024 como en Openoffice.org 3.x o las que lleguen a tener más delante, con lo que este código funcionara en las dos versiones, de hecho, en cualquier versión que implemente estas propiedades, y si, claro que tienes que evaluar también que no pases el limite de filas, pero esa, es tu tarea.

Hasta ahora hemos movido rangos dentro de la misma hoja, pero de forma muy sencilla podemos mover rangos entre hojas, solo hay que establecer el destino correctamente y el método se encargará del resto.

```
Sub MoverRangos7()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object
Dim lLimiteCol As Long

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oOrigen = ThisComponent.getCurrentSelection()
   If oOrigen.getImplementationName() = "ScCellRangeObj" Then
         oDestino = ThisComponent.getSheets.getByIndex(0).getCellByPosition( 0, 0 )
         oHojaActiva.moveRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )
         ThisComponent.getCurrentController.select( oOrigen )
   Else
         MsgBox "El rango Origen no es correcto"
   End If

End Sub
```

Observa como el origen lo establecemos en la hoja activa, pero el destino lo referenciamos a la primer hoja del documento, claro que puedes hacerlo a cualquier otra, por índice o por nombre como ya lo hemos aprendido. Nota que el método moveRange, lo llamamos desde la hoja activa, pero también lo puedes llamar desde cualquier hoja, lo importante es que los argumentos, es decir, el origen y destino estén correctos, observa en el siguiente ejemplo, como llamamos al método moveRange desde la hoja del rango destino.

```
Sub MoverRangos8()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object
Dim lLimiteCol As Long

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oOrigen = ThisComponent.getCurrentSelection()
   If oOrigen.getImplementationName() = "ScCellRangeObj" Then
         oDestino = ThisComponent.getSheets.getByIndex(0).getCellByPosition( 0, 0 )
         oDestino.getSpreadSheet.moveRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )
         ThisComponent.getCurrentController.select( oOrigen )
   Else
         MsgBox "El rango Origen no es correcto"
   End If

End Sub
```

También podemos mover una columna completa, recuerda que una columna no es más que un rango de celdas.

```
Sub MoverRangos9()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Referenciamos como origen la columna E
   oOrigen = oHojaActiva.getColumns.getByIndex(4)
   'Como destino una columna a la derecha
   oDestino = oHojaActiva.getCellByPosition( oOrigen.getRangeAddress().EndColumn + 1, 0 )
   oHojaActiva.moveRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )
```

```
    ThisComponent.getCurrentController.select( oOrigen )

End Sub
```

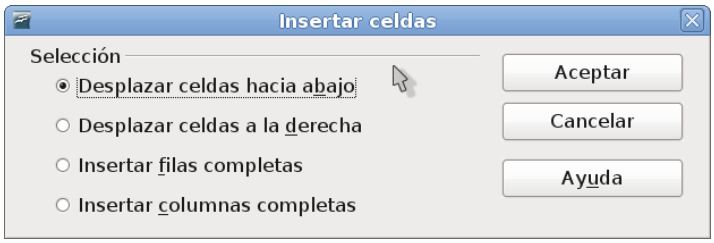
Y ya encarrerados, movemos filas también.

```
Sub MoverRangos10()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object

    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
    'Referenciamos como origen la fila 5
    oOrigen = oHojaActiva.getRows.getByIndex(4)
    'Como destino la fila 10
    oDestino = oHojaActiva.getCellByPosition( 0, 9 )
    oHojaActiva.moveRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )
    ThisComponent.getCurrentController.select( oOrigen )

End Sub
```

Cuando muevas columnas y filas completas, ten en cuenta las mismas consideraciones vistas en los demás ejemplos, de las cuales las principales son que los datos se mueven del origen al destino sin preguntar, es decir, no hay ninguna confirmación de sobreescritura en caso de que el destino no este vacío, por lo que tú tienes que implementar esta validación y tener cuidado de no salirse de los limites de la hoja de calculo.

6.3.2 Insertando rangos

En la interfaz del usuario, cuando insertamos un rango, recordaras que Calc, nos muestra un cuadro de dialogo preguntándonos como desplazar las celdas adyacentes, esta misma consideración hay que tener cuando lo hacemos por código, veamos como.




```
Hoja.insertCells( Celdas As CellRangeAddress, Modo As CellInsertMode)
```

```
Sub InsertarRangos1()
Dim oHojaActiva As Object
Dim oSel As Object

    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
    oSel = ThisComponent.getCurrentSelection()

    'Insertamos un rango de celdas igual a la selección actual
    'y movemos las celdas hacia abajo
```

```
    oHojaActiva.insertCells(oSel.getRangeAddress(), com.sun.star.sheet.CellInsertMode.DOWN)

End Sub
```

El rango a insertar no tiene por que ser a partir de la selección actual, puedes crear una estructura CellRangeAddress vacía del tamaño que quieras como en el siguiente ejemplo.

```
Sub InsertarRangos2()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Creamos una estructura vacía de dirección de un rango
   oRango = CreateUnoStruct( "com.sun.star.table.CellRangeAddress" )

   'Establecemos los valores del rango, tres filas por tres columnas
   With oRango
         .Sheet = oHojaActiva.getRangeAddress().Sheet
         .StartColumn = 1
         .StartRow = 1
         .EndColumn = 3
         .EndRow = 3
   End With

   'Insertamos y desplazamos hacia la derecha
   oHojaActiva.insertCells(oRango, com.sun.star.sheet.CellInsertMode.RIGHT)

End Sub
```

Es muy importante que establezcas la propiedad Sheet (hoja) de esta estructura, si no lo haces, el valor predeterminado es 0, con lo que el rango insertado “siempre” lo hará en la primer hoja del documento. En este segundo ejemplo hemos desplazado las celdas a la derecha, las demás opciones de este método son insertar filas completas o columnas completas, en la siguiente tabla resumimos los cuatro valores posibles para este método.

Constante	*Valor*
com.sun.star.sheet.CellInsertMode.DOWN	1
com.sun.star.sheet.CellInsertMode.RIGHT	2
com.sun.star.sheet.CellInsertMode.ROWS	3
com.sun.star.sheet.CellInsertMode.COLUMNS	4

Puedes usar indistintamente la constante o el valor de esta como en el siguiente ejemplo donde insertamos filas completas.

```
Sub InsertarRangos3()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oRango = CreateUnoStruct( "com.sun.star.table.CellRangeAddress" )

   With oRango
         .Sheet = oHojaActiva.getRangeAddress().Sheet
         .StartColumn = 1
         .StartRow = 1
         .EndColumn = 3
         .EndRow = 3
```

```
    End With

    'Insertamos filas completas
    oHojaActiva.insertCells( oRango, 3 )

End Sub
```

Cuando quieras insertar filas o columnas completas, puedes usar los métodos vistos hasta ahora o, puedes usar los métodos específicos del conjunto de filas y columnas como en el siguiente ejemplo donde insertamos 2 columnas a partir de la columna E.

```
Sub InsertarRangos4()
Dim oHojaActiva As Object

    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

    'Insertamos 2 columnas a partir de la columna E
    oHojaActiva.getColumns.insertByIndex( 4, 2 )

End Sub
```

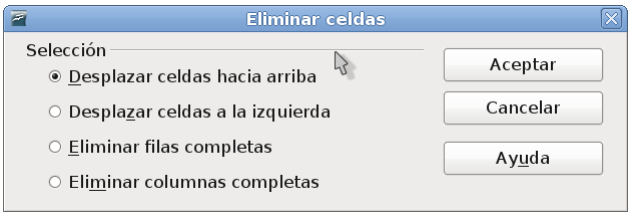
Nota que ahora usamos el método **insertByIndex**, que es especifico del conjunto de columnas, por ello primero llamados a **getColumns**(), el primer argumento de este método es el índice de la columna donde comenzara la inserción y el segundo es el número de columnas que deseamos insertar. El método para insertar filas es exactamente igual, excepto por que lo llamamos desde el conjunto de filas (**getRows**) como en.

```
Sub InsertarRangos5()
Dim oHojaActiva As Object

    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

    'Insertamos 5 filas a partir de la fila 3
    oHojaActiva.getRows.insertByIndex( 2, 5 )

End Sub
```

Para terminar este tema, recuerda que no puedes desplazar celdas fuera del rango de la hoja de calculo, por lo que tienes que evaluar que tengas suficiente espacio para la inserción, de lo contrario, te dará un error en tiempo de ejecución.

6.3.3 Eliminando rangos

Eliminar rangos, es la operación inversa a insertarlos, en este caso, tenemos que decidir que hacer con las celdas adyacentes al rango, es decir, como se desplazarán, en nuestro primer ejemplo, movemos las celdas hacia arriba.




```
Hoja.removeRange( Celdas As CellRangeAddress, Modo As CellDeleteMode)
```

```
Sub BorrarRangos1()
Dim oHojaActiva As Object
Dim oSel As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oSel = ThisComponent.getCurrentSelection()

   'Borramos la selección actual y movemos las celdas hacia arriba
   oHojaActiva.removeRange( oSel.getRangeAddress(), com.sun.star.sheet.CellDeleteMode.UP )

End Sub
```

Cuando se hace cualquier operación de borrado o eliminación, es una buena practica de programación que confirmes esta acción con el usuario, sobre todo, con aquellas acciones que no es posible deshacer, es casi una regla que lo hagas. En el siguiente ejemplo, desplazamos hacia la izquierda, después de confirmar la eliminación.

```
Sub BorrarRangos2()
Dim oHojaActiva As Object
Dim oRango As Object
Dim iRes As Integer

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango = CreateUnoStruct( "com.sun.star.table.CellRangeAddress" )

   'Establecemos el rango E8:F21
   oRango = oHojaActiva.getCellRangeByName( "E8:F21" )

   'Confirmamos la eliminacion
   iRes = MsgBox( "Estas seguro de borrar el rango", 4 + 32, "Borrar rango" )

   'Solo borramos si el usuario respondió SI
   If iRes = 6 Then
        'Borramos el rango y movemos las celdas hacia la izquierda
        oHojaActiva.removeRange( oRango.getRangeAddress, com.sun.star.sheet.CellDeleteMode.LEFT )
   End If

End Sub
```

En la siguiente tabla puedes ver las posibilidades de este método, así como sus valores que puedes usar en vez de ellas.

Constante	*Valor*
com.sun.star.sheet.CellDeleteMode.UP	1
com.sun.star.sheet.CellDeleteMode.LEFT	2

Constante	*Valor*
com.sun.star.sheet.CellDeleteMode.ROWS	3
com.sun.star.sheet.CellDeleteMode.COLUMNS	4

Podemos borrar columnas completas, en el siguiente ejemplo usamos el valor de la constante en vez de esta, ya no pedimos confirmación, pero te recomiendo siempre hacerla.

```
Sub BorrarRangos3()
Dim oHojaActiva As Object
Dim oRango As Object
Dim iRes As Integer

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango = CreateUnoStruct( "com.sun.star.table.CellRangeAddress" )

   'Establecemos el rango B2:D2
   oRango = oHojaActiva.getCellRangeByName( "B2:D2" )

   'Borramos las columnas completas
   oHojaActiva.removeRange( oRango.getRangeAddress, 4 )

End Sub
```

Al igual que con la inserción, para la eliminación de filas y columnas completas, se cuenta con métodos alternativos accesibles desde el conjunto de filas (getRows) y columnas (getColumns), veamos como.

```
Sub BorrarRangos4()
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()

   'Borramos 3 filas a partir de la fila 10
   oHojaActiva.getRows.removeByIndex( 9, 3 )

   'Borramos 2 columnas a partir de la A
   oHojaActiva.getColumns.removeByIndex( 0, 2 )

End Sub
```

No se te olvide siempre confirmar las eliminaciones.

6.3.4 Copiando rangos

Copiar rangos es muy similar a moverlos, se usan los mismos argumentos, un rango origen y una celda destino, claro, cambia el método usado, al igual que cuando movemos, el destino será reemplazado con el origen sin ningún tipo de confirmación, pero como ya sabes implementarla, no tienes problemas con ello, ¿verdad?

```
Hoja.copyRange( Destino As CellAddress, Origen As CellRangeAddress)
```

```
Sub CopiarRangos1()
```

```
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   'Rango a copiar
   oOrigen = oHojaActiva.getCellRangeByName( "A1:B3" )
   'Celda destino
   oDestino = oHojaActiva.getCellRangeByName( "D10" )

   'Copiamos el rango
   oHojaActiva.copyRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )

End Sub
```

A diferencia de cuando movemos, el origen permanece tal cual y aquí si, si el rango contiene formulas, las referencias relativas se ajustarán automáticamente a la nueva posición, además de que el rango origen permanece con la referencia original.

```
Sub CopiarRangos2()
Dim oHojaActiva As Object
Dim oOrigen As Object
Dim oDestino As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oOrigen = ThisComponent.getCurrentSelection()

   oDestino = oHojaActiva.getCellByPosition( oOrigen.getRangeAddress.EndColumn + 1,
oOrigen.getRangeAddress.EndRow + 1 )
   oHojaActiva.copyRange( oDestino.getCellAddress, oOrigen.getRangeAddress() )

   ThisComponent.getCurrentController.select( oOrigen )

End Sub
```

Para copiar en una hoja diferente, solo tienes que establecer el destino correctamente en dicha hoja, el siguiente ejemplo, copiamos el rango B2:D5 de la ultima hoja del documento a la celda A1 de la primera.

```
Sub CopiarRangos3()
Dim oHojaOrigen As Object
Dim oHojaDestino As Object
Dim oOrigen As Object
Dim oDestino As Object

   oHojaOrigen = ThisComponent.getSheets.getByIndex( ThisComponent.getSheets.getCount() - 1 )
   oHojaDestino = ThisComponent.getSheets.getByIndex( 0 )

   'Rango a copiar
   oOrigen = oHojaOrigen.getCellRangeByName( "B2:D5" )
   'Celda destino
   oDestino = oHojaDestino.getCellRangeByName( "A1" )

   'Copiamos el rango
   oHojaDestino.copyRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )

   ThisComponent.getCurrentController.select( oDestino )

End Sub
```

Puedes copiar columnas o filas completas.

```
Sub CopiarRangos4()
Dim oHojaOrigen As Object
Dim oHojaDestino As Object
Dim oOrigen As Object
Dim oDestino As Object

   'Primer hoja
   oHojaOrigen = ThisComponent.getSheets.getByIndex( 0 )
   'Rango a copiar columna B
   oOrigen = oHojaOrigen.getColumns.getByIndex( 1 )
   'Segunda Hoja
   oHojaDestino = ThisComponent.getSheets.getByIndex( 1 )
   oDestino = oHojaDestino.getCellRangeByName( "E1" )

   oHojaDestino.copyRange( oDestino.getCellAddress(), oOrigen.getRangeAddress() )

   ThisComponent.getCurrentController.select( oDestino )

End Sub
```

Hagamos algo muy interesante, como sabemos, cuando copiamos un rango de celdas y este rango tiene filas o columnas ocultas manualmente o por estar agrupadas, la copia resultante, contendrá aun, las filas y columnas ocultas, con el método para seleccionar solo las celdas visibles (*queryVisibleCells*) aprendido en el capitulo anterior y el método para copiar rangos, podemos intentar hacer nuestra versión de una herramienta que llamaremos “Copiar solo visibles”, ¿te parece?. Antes de que sigas leyendo, te invito a que intentes resolverlo por ti mismo primero, después checa mi versión y compáralas, claro, solo con fines didácticos puesto que estamos aprendiendo.

```
Sub CopiarSoloVisibles1()
Dim oSel As Object
Dim oCursor As Object
Dim oVisibles As Object
Dim oHojaOrigen As Object
Dim oHojaDestino As Object
Dim oRangoOrigen As Object
Dim oCeldaDestino As New com.sun.star.table.CellAddress
Dim col As Long
Dim mDir

   oHojaOrigen = ThisComponent.getCurrentController.getActiveSheet()
   'Referencia a la selección actual
   oSel = ThisComponent.getcurrentSelection()
   'Si es una sola celda regresamos la región actual
   If oSel.getImplementationName() = "ScCellObj" Then
         oCursor = oSel.getSpreadSheet.createCursorByRange( oSel )
         oCursor.collapseToCurrentRegion()
         'Y las celdas visibles
         oVisibles = oCursor.queryVisibleCells()
   ElseIf oSel.getImplementationName() = "ScCellRangesObj" Then
         oVisibles = oSel
   End If

   'si no hay celdas visibles
   If IsNull( oVisibles ) Then
         MsgBox "No hay celdas ocultas"
   Else
         'Agregamos una nueva hoja
         oHojaDestino = getNuevaHoja( ThisComponent, oHojaOrigen )
         'Obtenemos una matriz con las direcciones de todos los rangos
         mDir = oVisibles.getRangeAddresses()
         'Iteramos en cada dirección
```

```
        For co1 = LBound(mDir) To UBound(mDir)
            oRangoOrigen = mDir( co1 )
            'La celda destino tendrá la misma dirección del rango pero en la hoja destino
            oCeldaDestino.Sheet = oHojaDestino.getRangeAddress.Sheet
            oCeldaDestino.Column = oRangoOrigen.StartColumn
            oCeldaDestino.Row = oRangoOrigen.StartRow
            'Copiamos el rango
            oHojaDestino.copyRange( oCeldaDestino, oRangoOrigen )
        Next co1
        'Seleccionamos la nueva hoja con los datos copiados
        ThisComponent.getCurrentController.setActiveSheet( oHojaDestino )
        MsgBox "Rangos copiados"
    End If

End Sub


'Devuelve una nueva hoja en Documento, a la derecha del argumento Hoja
Function getNuevaHoja( Documento As Object, Hoja As Object ) As Object
Dim oHojas As Object
Dim co1 As Integer
Dim sNombre As String

    oHojas = Documento.getSheets()
    sNombre = "Rangos Copiados"
    Do While oHojas.hasByName( sNombre )
        co1 = co1 + 1
        sNombre = sNombre & " " & Format(co1)
    Loop
    oHojas.insertNewByName( sNombre, Hoja.getRangeAddress.Sheet+1 )
    getNuevaHoja = Documento.getSheets.getByName( sNombre )

End Function
```

Seguro que ya viste que no trabaja muy bien que digamos, claro, no es nada practico que use la misma dirección del origen en el destino pues los datos quedan todos dispersos. Vamos a mejorarla, de nuevo, intenta corregirla tu y después continuas.

```
Sub CopiarSoloVisibles2()
Dim oSel As Object
Dim oCursor As Object
Dim oVisibles As Object
Dim oHojaOrigen As Object
Dim oHojaDestino As Object
Dim oRangoOrigen As Object
Dim oRangoAnterior As Object
Dim oCeldaDestino As New com.sun.star.table.CellAddress
Dim co1 As Long, Fil As Long, Col As Long
Dim mDir

    oHojaOrigen = ThisComponent.getCurrentController.getActiveSheet()
    oSel = ThisComponent.getcurrentSelection()
    Select Case oSel.getImplementationName
        Case "ScCellObj"
            oCursor = oSel.getSpreadSheet.createCursorByRange( oSel )
            oCursor.collapseToCurrentRegion()
            oVisibles = oCursor.queryVisibleCells()
        Case "ScCellRangeObj", "ScCellRangesObj"
            oVisibles = oSel.queryVisibleCells()
    End Select

    If IsNull( oVisibles ) Then
        MsgBox "No hay celdas ocultas o no es un rango de celdas"
    Else
```

```
        Fil = 0
        Col = 0
        oHojaDestino = getNuevaHoja( ThisComponent, oHojaOrigen  )
        mDir = oVisibles.getRangeAddresses()
        'Copiamos el primer rango
        oRangoOrigen = mDir( 0 )
        oCeldaDestino.Sheet = oHojaDestino.getRangeAddress.Sheet
        'En la celda A1
        oCeldaDestino.Column = 0
        oCeldaDestino.Row = 0
        oHojaDestino.copyRange( oCeldaDestino, oRangoOrigen )
        'Si tenemos más rangos
        If oVisibles.getCount() > 1 then
              For co1 = 1 To UBound(mDir)
                    oRangoOrigen = mDir( co1 )
                    oRangoAnterior = mDir( co1-1 )
                    'Vamos sumando cada ancho y alto de cada rango, solo cuando cambien
                    If oRangoAnterior.StartColumn = oRangoOrigen.StartColumn Then
                          oCeldaDestino.Row = oCeldaDestino.Row + oRangoAnterior.EndRow -
oRangoAnterior.StartRow + 1
                    Else
                          oCeldaDestino.Column = Col + oRangoAnterior.EndColumn -
oRangoAnterior.StartColumn + 1
                          oCeldaDestino.Row = Fil
                          Col = oCeldaDestino.Column
                    End If
                    oHojaDestino.copyRange( oCeldaDestino, oRangoOrigen )
              Next co1
        End If
        ThisComponent.getCurrentController.setActiveSheet( oHojaDestino )
   End If

End Sub
```

Ahora si, trabaja mucho mejor, pero, todavía tiene un “pequeño” detalle, hay un caso particular donde el rango no contenga celdas ocultas y nuestra macro no lo informe, y digo que es un detalle por que no te dará ningún error y seguirá funcionando, puedes considerar evaluarlo o dejarla así, pero eso si, tu tarea es encontrar este caso particular. También, podrías mejorar esta macro para que copie solo datos o resultados de formulas, esto lo podrás hacer cuando adquieras los conocimientos del próximo capitulo.

6.4 Manipulando datos

Ya vamos llegando a temas más interesantes y divertidos, donde veremos como empezar a interactuar con las celdas de nuestra hoja de calculo. Para el mejor aprovechamiento de este capitulo, tienes que tener presente los diferentes tipos de datos que soporta Calc.

6.4.1 Obteniendo datos

La forma más simple y directa de obtener el contenido de una celda, es usando el método **getString**, que te devuelve el contenido de la celda, tal y como se ve en la interfaz del usuario, es decir, si el contenido de la celda es texto, te lo muestra tal cual, si contiene una formula, te mostrara el resultado de dicha formula y si es fecha y tiene formato, te la devolverá como se ve en pantalla.

```
Sub Datos1()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellObj" Then
         'Mostramos el contenido de la celda
         MsgBox oSel.getString()
   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Si lo que deseas es devolver el valor de la celda, se usa el método **getValue**, si la celda contiene texto, este método te devolverá 0, si tiene un valor, dicho valor, si la celda contiene una formula, dependerá del tipo de resultado de esta formula, si es texto de nuevo te devolverá 0, si es un valor, este valor, si tiene un error de nuevo será 0, si la celda contiene una fecha, te devolverá el número de serie de esta fecha.

```
Sub Datos2()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellObj" Then
         'Mostramos el valor de la celda
         MsgBox oSel.getValue()
   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

También puedes obtener la formula que tenga la celda con el método **getFormula**, si la celda contiene texto, te devolverá este texto, si es un valor, ese valor, de una fecha te devuelve el número de serie y si es formula, te devuelve dicha formula, incluyendo el signo de igual (=) con que empieza toda formula aun y cuando la formula tenga como resultado un error, también, toma en cuenta que si la formula devuelta contiene alguna función incorporada de Calc, como SUMA, BUSCARV, etc, este método te devolverá el nombre de esta función en ingles por ejemplo: SUM, VLOOKUP, etc.

```
Sub Datos3()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellObj" Then
         'Mostramos la formula de la celda
         MsgBox oSel.getFormula()
   Else
         MsgBox "Selecciona solo una celda"
   End If
```

```
End Sub
```

Si deseas obtener la formula tal y como se ve en la barra de formulas, entonces usa la propiedad FormulaLocal, que se comporta de forma muy similar a getString, excepto en las formulas, donde te las devuelve como aparecen en la barra de formulas, como en.

```
Sub Datos4()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellObj" Then
         'Mostramos la formula local de la celda
         MsgBox oSel.FormulaLocal
   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Si quieres ver las diferencias, captura valores en varias celdas, texto, números, fechas y formulas, y prueba la siguiente macro, toma en cuenta que estos métodos solo los puedes usar en celdas individuales, por ello hacemos la validación, en una de las formulas usa la función =ALEATORIO(), veras claramente las diferencias entres estos métodos.

```
Sub Datos5()
Dim oSel As Object
Dim sTmp As String

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellObj" Then
         'Mostramos las diferencias entre los métodos
         sTmp = "getString : " & oSel.getString & Chr(13)
         sTmp = sTmp & "getValue : " & oSel.getValue & Chr(13)
         sTmp = sTmp & "getFormula : " & oSel.getFormula & Chr(13)
         sTmp = sTmp & "FormulaLocal : " & oSel.FormulaLocal
         'Mostramos el resultado
         MsgBox sTmp
   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

También puedes obtener el tipo de contenido de la celda con **getType**, que te devolverá un entero dependiendo del contenido de la celda, vacía (0), valor (1), texto (2) o formula (3), de nuevo, este método solo esta disponible es una sola celda.

```
Sub Datos6()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   MsgBox oSel.getType()

End Sub
```

Si la celda contiene una formula y esta devuelve un error, puedes saber que error es con el método **getError**, si la celda no contiene una formula, este método siempre te devolverá 0.

```
Sub Datos7()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   MsgBox oSel.getError()

End Sub
```

El error división entre cero, devuelve el error 532, establece este error en una formula cualquiera para que lo compruebes, también puedes probar con el error de cuando a una formula le faltan argumentos, normalmente da el error 511.

Podríamos aventurar una primera forma de usar el método correcto, de acuerdo al contenido de la celda.

```
Sub Datos8()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   Select Case oSel.getType()
         Case 0
               MsgBox "La celda esta vacia"
         Case 1
               MsgBox oSel.getValue()
         Case 2
               MsgBox oSel.getString()
         Case 3
               If oSel.getError() = 0 Then
                     MsgBox oSel.FormulaLocal
               Else
                     MsgBox oSel.getError()
               End If
   End Select

End Sub
```

Y si se convierte en función.

```
Function ContenidoCelda(Celda As Object)
Dim tmp

   Select Case Celda.getType()
         Case 0 : tmp = "La celda esta vacía"
         Case 1 : tmp = Celda.getValue()
         Case 2 : tmp = Celda.getString()
         Case 3
               If Celda.getError() = 0 Then
                     tmp = Celda.FormulaLocal
               Else
                     tmp = Celda.getError()
               End If
   End Select

   ContenidoCelda = tmp

End Function
```

Toma en cuenta que una celda te puede devolver 532 y estar correcto, es decir que sea por ejemplo el número de alguna calle o ser el valor de alguna acción y seguir siendo correcto

o ser el error 532 y como información, seguir siendo correcto, así que no tomes la función anterior como definitiva, sino tan solo como una guía para lo que realmente necesites.

Ahora solo nos resta poder llamarla desde la celda.

```
Sub Datos9()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   MsgBox ContenidoCelda ( oSel )

End Sub
```

Y con un poco de ingenio desde cualquier rango de celdas.

```
Sub Datos10()
Dim oSel As Object
Dim fil As Long, col As Long

   oSel = ThisComponent.getCurrentSelection()
   Select Case oSel.getImplementationName()
         Case "ScCellObj"
               MsgBox ContenidoCelda ( oSel )
         Case "ScCellRangeObj"
               For fil = 0 To oSel.getRows().getCount() - 1
                     For col = 0 To oSel.getColumns().getCount() - 1
                           MsgBox ContenidoCelda ( oSel.getCellByPosition(col,fil) )
                     Next col
               Next fil
   End Select

End Sub
```

Toma en cuenta que getCellByPosition, obtiene una referencia a una sola celda, y esta, es en **“referencia”** a la selección original, te queda de tarea, obtener lo mismo pero con relación a la hoja completa, te doy dos pistas, usa el mismo método (getCellByPosition), pero toma como base de información para los ciclos la que te devuelve el método getRangeAddress.

En todos los casos anteriores, los métodos usados solo funcionan cuando hacemos referencia a una sola celda, si quieres obtener el contenido de un rango de celdas, usaremos el método getData, que te devuelve una matriz de matrices con el contenido de las celdas, pero cuidado, getData solo te devuelve las celdas con valores, además, en las celdas vacías devuelve un valor un tanto “extraño”, puedes verificarlo con el siguiente ejemplo, procura seleccionar un rango pequeño porque te mostrará el valor de cada celda.

```
Sub Datos11()
Dim oSel As Object
Dim mDatos
Dim mTmp
Dim co1 As Long, co2 As Long

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellRangeObj" Then
         'Obtenemos SOLO VALORES
         mDatos = oSel.getData()
         For co1 = LBound(mDatos) to UBound(mDatos)
               'Asignamos la matriz interna a una temporal
               mTmp = mDatos(co1)
```

```
            For co2 = LBound(mTmp) to UBound(mTmp)
                  MsgBox mTmp(co2)
            Next
      Next
   End If

End Sub
```

En vez de usar una matriz de apoyo, es más transparente obtener el número de filas y columnas del rango, un rango de celdas siempre es rectangular por lo que podemos garantizar que los índices de las matrices siempre serán correctos.

```
Sub Datos12()
Dim oSel As Object
Dim mDatos
Dim co1 As Long, co2 As Long
Dim Fil As Long, Col As Long

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellRangeObj" Then
         'Obtenemos SOLO VALORES
         mDatos = oSel.getData()
         'Obtenemos el número de filas y columnas
         Fil = oSel.getRows.getCount() - 1
         Col = oSel.getColumns.getCount() - 1

         For co1 = 0 To Fil
               For co2 = 0 to Col
               'Es más claro el acceso a la matriz
               MsgBox mDatos (co1) (co2)
               Next
         Next
   End If

End Sub
```

Si quieres obtener el contenido sea cual sea, usa la misma estructura, pero en vez de usar getData, usa getDataArray como en.

```
Sub Datos13()
Dim oSel As Object
Dim mDatos
Dim co1 As Long, co2 As Long
Dim Fil As Long, Col As Long

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellRangeObj" Then
         'Obtenemos todos los datos
         mDatos = oSel.getDataArray()
         'Obtenemos el número de filas y columnas
         Fil = oSel.getRows.getCount() - 1
         Col = oSel.getColumns.getCount() - 1

         For co1 = 0 To Fil
               For co2 = 0 to Col
               'Es más claro el acceso a la matriz
               MsgBox mDatos (co1) (co2)
               Next
         Next
   End If
```

```
End Sub
```

Y si lo que quieres es el contenido de las celdas, como si hubieses usado en cada una el método getFormula, usas getFormulaArray.

```
Sub Datos14()
Dim oSel As Object
Dim mDatos
Dim co1 As Long, co2 As Long
Dim Fil As Long, Col As Long

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellRangeObj" Then
         'Obtenemos todos los datos
         mDatos = oSel.getFormulaArray()
         'Obtenemos el número de filas y columnas
         Fil = oSel.getRows.getCount() - 1
         Col = oSel.getColumns.getCount() - 1

         For co1 = 0 To Fil
               For co2 = 0 to Col
                     'Es más claro el acceso a la matriz
                     MsgBox mDatos (co1) (co2)
               Next
         Next
   End If

End Sub
```

De la combinación de los métodos vistos en este tema, puedes acceder a cualquier información de una celda o un rango de celdas, veamos ahora como introducir información.

6.4.2 Introduciendo datos

Los métodos para introducir datos en celdas, son más o menos los mismos que para obtenerlos, pero en vez de obtenerlos (get) los establecemos (set).

Para el caso de cadenas de texto, usamos setString, toma en cuenta que este método reemplazará el contenido de la celda sin preguntarte nada.

```
Sub Introducir1()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellObj" Then
         'Insertamos una cadena en la celda
         oSel.setString( "Nadie esta más perdido que quien no sabe a donde va" )
   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Para valores usamos setValue, del mismo modo que setString, simplemente reemplazará el contenido de la celda sin consultarte.

```
Sub Introducir2()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellObj" Then
         'Insertamos un valor en la celda
         oSel.setValue( 34 )
   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Ya lo habrás intuido, para las formulas podemos usar setFormula, la formula debe estar como una cadena de texto y debe ser una formula válida.

```
Sub Introducir3()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellObj" Then
         'Insertamos una formula
         oSel.setFormula( "=A1+A5" )
   Else
         MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Si introduces una formula no válida, este método no te dará ningún error, pero si lo obtendrás en la interfaz del usuario, como en el siguiente ejemplo.

```
Sub Introducir4()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellObj" Then
         'Insertamos una formula no valida
         oSel.setFormula( "=C2-C3+" )
   End If

End Sub
```

Si hace uso de funciones incorporadas de OpenOffice.org, tienes que usar el nombre en ingles de la función a usar, también, los argumentos deben estar correctamente establecidos, así como los tipos de estos para que no te devuelva ningún error.

```
Sub Introducir5()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName() = "ScCellObj" Then
         'Insertamos una formula con función
```

```
        oSel.setFormula( "=SUM(A1:A10)" )
    End If

End Sub
```

Si prefieres usar los nombres de las funciones en español, en vez de este método, usa la propiedad FormulaLocal, el siguiente ejemplo hace lo mismo que el anterior.

```
Sub Introducir6()
Dim oSel As Object

    oSel = ThisComponent.getCurrentSelection()

    If oSel.getImplementationName() = "ScCellObj" Then
        'Insertamos una formula con función
        oSel.FormulaLocal = "=SUMA(A1:A10)"
    End If

End Sub
```

En el Apéndice te muestro una lista de equivalencias de las formulas que incorpora Calc en español e ingles, así puedes usar el método que quieras.

Cuando queramos introducir valores en rangos de celdas, hay que tener la precaución de establecer el rango destino, “exactamente” del mismo ancho y alto de la matriz origen, de lo contrario te dará un error en tiempo de ejecución.

```
Sub Introducir7()
Dim oHojaActiva As Object
Dim oRango As Object
Dim mDatos(4)

    mDatos(0) = Array(1,2,3)
    mDatos(1) = Array(4,5,6)
    mDatos(2) = Array(7,8,9)
    mDatos(3) = Array(10,11,12)
    mDatos(4) = Array(13,14,15)

    oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
    'El rango donde se insertaran los valores, "debe"
    'ser del mismo tamaño en ancho y alto de la matriz
    oRango = oHojaActiva.getCellRangeByName("A1:C5")
    'Insertamos la matriz completa
    oRango.setData( mDatos )

End Sub
```

También, toma nota de que con el método setData, solo puedes introducir valores, si la matriz lleva alguna cadena de texto, este método la reemplazara por 0, para valores y cadenas, debes usar setDataArray.

```
Sub Introducir8()
Dim oHojaActiva As Object
Dim oRango As Object
Dim mDatos(4)

    mDatos(0) = Array("No","Nombre","Tel")
    mDatos(1) = Array(1,"Gloria",12345678)
    mDatos(2) = Array(1,"Paola",23456789)
    mDatos(3) = Array(3,"Lidia",34567891)
```

```
    mDatos(4) = Array(4,"Lizette",87654321)

    oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
    'El rango donde se insertaran los valores, "debe"
    'ser del mismo tamaño en ancho y alto de la matriz
    oRango = oHojaActiva.getCellRangeByName("A1:C5")
    'Insertamos la matriz completa
    oRango.setDataArray( mDatos )

End Sub
```

Si tus datos a introducir incluyen formulas, es mejor que uses setFormulaArray.

```
Sub Introducir9()
Dim oHojaActiva As Object
Dim oRango As Object
Dim mDatos(4)

   mDatos(0) = Array("No","Nombre","Tel")
   mDatos(1) = Array(1,"Gloria","=RAND()")
   mDatos(2) = Array(1,"Paola","=A3")
   mDatos(3) = Array(3,"Lidia","=SUM(A2:A4)")
   mDatos(4) = Array(4,"Lizette","=RAND()")

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A1:C5")
   'Insertamos la matriz completa
   oRango.setFormulaArray( mDatos )

End Sub
```

6.4.3 Borrando datos

Para borrar datos de celdas usamos el método clearContents, este método requiere un entero indicándole el tipo de contenido que deseamos borrar, por ejemplo valores (1), textos (4) o formulas (16), este método esta presente en los tres tipos de rangos vistos, por lo que solo tienes que asegurarte que efectivamente sea un rango de celdas.

En el siguiente ejemplo, se borran solo las celdas con valores, textos y formulas, de tres rangos diferentes.

```
Sub Borrando1()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A1:C5")
   'Borramos solo los valores
   oRango.clearContents( 1 )

   oRango = oHojaActiva.getCellRangeByName("D2:E10")
   'Borramos solo los texto
   oRango.clearContents( 4 )

   oRango = oHojaActiva.getCellRangeByName("G1:K100")
   'Borramos solo las formulas
   oRango.clearContents( 16 )
```

```
End Sub
```

Puedes sumar los valores de los tipos a borrar, en el siguiente ejemplo, se borra los textos y las formulas del rango seleccionado.

```
Sub Borrando2()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   'Borramos los textos (4) + formulas (16)
   oSel.clearContents( 20 )

End Sub
```

Los valores permitidos para este método, están condicionados por las constantes del grupo com.sun.star.sheet.CellFlags, cuyos valores se listan en la siguiente tabla.

Constante	*Valor*
com.sun.star.sheet.CellFlags.VALUE	1
com.sun.star.sheet.CellFlags.DATETIME	2
com.sun.star.sheet.CellFlags.STRING	4
com.sun.star.sheet.CellFlags.ANNOTATION	8
com.sun.star.sheet.CellFlags.FORMULA	16
com.sun.star.sheet.CellFlags.HARDATTR	32
com.sun.star.sheet.CellFlags.STYLES	64
com.sun.star.sheet.CellFlags.OBJECT	128
com.sun.star.sheet.CellFlags.EDITATTR	256
com.sun.star.sheet.CellFlags.FORMATTED	512

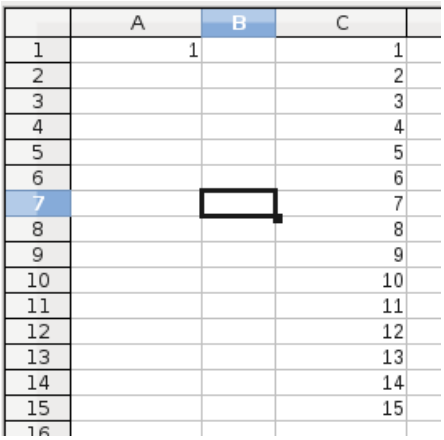
Puedes usar de forma indistinta las constantes o los valores de estas, así como cualquier combinación de ellas. El siguiente ejemplo borra “todo” te deja la selección como nueva.

```
Sub Borrando3()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   'Borramos todo
   oSel.clearContents( 1023 )

End Sub
```

6.4.4 Llenando series de datos

Llenar series es un método muy divertido de introducir datos en nuestra hoja de calculo, existen dos formas de hacerlo, en general la forma automática funcionara si se establecen

correctamente los valores iniciales, observa la siguiente imagen, en la celda A1 tenemos solo un valor y en la columna C como queremos que quede nuestra serie.



El código para lograr esto es el siguiente, observa que sencillo.

```
Sub LlenandoSeries1()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A1:A15")

   'Llenamos hacia abajo
   oRango.fillAuto( 0, 1 )

End Sub
```

El método fillAuto, solo requiere dos argumentos, el primero es para indicarle la dirección del llenado en nuestro ejemplo, abajo (TO_BOTTOM = 0) y un entero que le indica, cuantas celdas del rango tomara como "guías" para determinar el algoritmo de llenado. En la siguiente tabla están las cuatro posibilidades del argumento "dirección" de este método.

Constante	*Valor*
com.sun.star.sheet.FillDirection.TO_BOTTOM	0
com.sun.star.sheet.FillDirection.TO_RIGHT	1
com.sun.star.sheet.FillDirection.TO_TOP	2
com.sun.star.sheet.FillDirection.TO_LEFT	3

Si quisiéramos llenar el rango con números pares, lo haríamos así.

```
Sub LlenandoSeries2()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A1:A15")

   'Establecemos los valores guias
   oHojaActiva.getCellRangeByName("A1").setValue( 2 )
   oHojaActiva.getCellRangeByName("A2").setValue( 4 )

   'Llenamos hacia abajo, nota que ahora tomamos dos celdas como guías
```

```
   oRango.fillAuto( 0, 2 )

End Sub
```

Por supuesto no solo con números es posible, el siguiente código establece el rango A1:L1 con los meses del año.

```
Sub LlenandoSeries3()
Dim oHojaActiva As Object
Dim oRango As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A1:L1")

   'Establecemos el valor guía
   oHojaActiva.getCellRangeByName("A1").setString( "enero" )

   'Llenamos a la derecha
   oRango.fillAuto( 1, 1 )

End Sub
```

Esto es posible por que esta lista viene de forma predeterminada en OpenOffice.org, esta y otras listas, las puedes personalizar en el menú *Herramientas* | *Opciones...*, dentro de la rama OpenOffice.org Calc, subrama Ordenar listas.

Donde más me gusta el llenado de series, es en la forma transparente de copiar o llenar un rango con una formula deseada, por ejemplo, observa la siguiente imagen

	A	B	C	D	E
1	Nº	Nombre	Año	Edad	
2	1	Edgar	1971		
3	2	Gloria	1977		
4	3	Antonio	1965		
5	4	Lidia	1969		
6	5	Paola	1972		
992	991	Lizette	1980		
993	992	Edgar	1970		
994	993	Gloria	1971		
995	994	Antonio	1969		
996	995	Lidia	1967		
997	996	Paola	1976		
998	997	Vanessa	1975		
999	998	Lizette	1969		
1000	999	Edgar	1972		
1001	1000	Gloria	1975		

El fin es establecer la formula para calcular la edad de cada persona, para fines didácticos daremos por buena la edad, solo restando el año actual de la de nacimiento, observa que los datos terminan en la fila 1001 pero eso no importa, pueden terminar en la 1000 o la que sea pues esto lo determinaremos por código, de la siguiente forma.

```
Sub LlenandoSeries4()
Dim oHojaActiva As Object
Dim oRango As Object
Dim oCursor As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName("A1")
```

```
    'Establecemos la primer formula
    oHojaActiva.getCellRangeByName("D2").setFormula( "=YEAR(NOW())-C2" )

    'Creamos un cursor a partir de la primer celda
    oCursor = oHojaActiva.createCursorByRange( oRango )
    'Expandimos a la región actual
    oCursor.collapseToCurrentRegion()

    'Construimos el rango a llenar
    oRango = oHojaActiva.getCellRangeByName( "D2:D" & CStr(oCursor.getRows().getCount()) )

    'Llenamos hacia abajo
    oRango.fillAuto( 0, 1 )

End Sub
```

Para establecer la formula, como ya lo vimos, puedes usar también las propiedades FormulaLocal, nuestra formula de ejemplo quedaría así.

```
oHojaActiva.getCellRangeByName("D2").FormulaLocal = "=AÑO(AHORA())-C2"
```

Observa como obtenemos la ultima fila del rango usado con la ayuda del cursor y el método oCursor.getRows().getCount(), y con una sola línea más, llenamos todo el rango.

6.4.5 Buscar y reemplazar

La búsqueda y reemplazo en OpenOffice.org es una herramienta muy poderosa, si bien tiene tantas variantes como las que puedes ver en: *Editar | Buscar y reemplazar...*, nos limitaremos a las opciones más comunes y generales, quedándote de tarea la investigación del resto. Usaremos para nuestras pruebas la siguiente tabla de datos, que, no es restrictiva, puedes usar la que gustes, pero esta pequeña nos permite comprobar rápidamente los resultados para verificar que funciona correctamente, después, puedes usar listados del tamaño que quieras y los recursos de tu equipo te lo permitan.

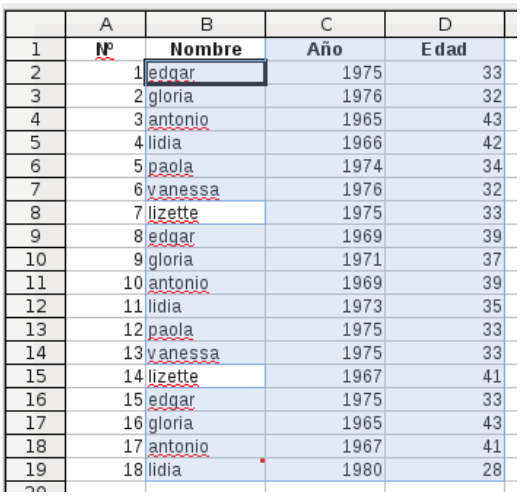
Nº	Nombre	Año	Edad
1	edgar	1975	33
2	gloria	1976	32
3	antonio	1965	43
4	lidia	1966	42
5	paola	1974	34
6	vanessa	1976	32
7	lizette	1975	33
8	edgar	1969	39
9	gloria	1971	37
10	antonio	1969	39
11	lidia	1973	35
12	paola	1975	33
13	vanessa	1975	33

14	lizette	1967	41
15	edgar	1975	33
16	gloria	1965	43
17	antonio	1967	41
18	lidia	1980	28

En las búsquedas, se usa un “descriptor de búsqueda” que no es otra cosa que una estructura donde le indicamos las características de la búsqueda y lo que estamos buscando y un lugar para buscar, que es el rango donde queremos hacer la búsqueda, por supuesto puedes buscar en toda la hoja, incluso en todas las hojas, la forma más sencilla de una búsqueda es la siguiente.

```
Sub Buscar1()
Dim oHojaActiva As Object
Dim oBuscarEn As Object
Dim oEncontrado As Object
Dim oSD As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Establecemos el rango donde buscaremos
   oBuscarEn = oHojaActiva.getCellRangeByName( "A1:D19" )

   'Creamos el descriptor de busqueda
   oSD = oHojaActiva.createSearchDescriptor

   'Establecemos lo que estamos buscando
   oSD.setSearchString( "33" )

   'Realizamos la busqueda de TODAS las coincidencias
   oEncontrado = oBuscarEn.findAll( oSD )

   'Si no hay coincidencias oEncontrado = Null
   If Not IsNull( oEncontrado ) Then
         'Si encuentra algo lo seleccionamos
         ThisComponent.getCurrentController.select( oEncontrado )
   Else
         MsgBox "No se encontraron coincidencias"
   End If

End Sub
```

Y nuestra primera búsqueda nos devuelve el mensaje “No se encontraron coincidencias”, ¿por qué si aparentemente si tenemos valores 33 en nuestros datos?, la respuesta es que la búsqueda predeterminada se hace buscando dentro de las formulas, y todos los 33 que ves en la lista son el “resultado” de una formula, dicha formula es: =AÑO(AHORA())-C2, ahora, establece el valor buscado en “a” y realiza la búsqueda: oSD.setSearchString("a").



	A	B	C	D
1	Nº	Nombre	Año	Edad
2	1	edgar	1975	33
3	2	gloria	1976	32
4	3	antonio	1965	43
5	4	lidia	1966	42
6	5	paola	1974	34
7	6	vanessa	1976	32
8	7	lizette	1975	33
9	8	edgar	1969	39
10	9	gloria	1971	37
11	10	antonio	1969	39
12	11	lidia	1973	35
13	12	paola	1975	33
14	13	vanessa	1975	33
15	14	lizette	1967	41
16	15	edgar	1975	33
17	16	gloria	1965	43
18	17	antonio	1967	41
19	18	lidia	1980	28

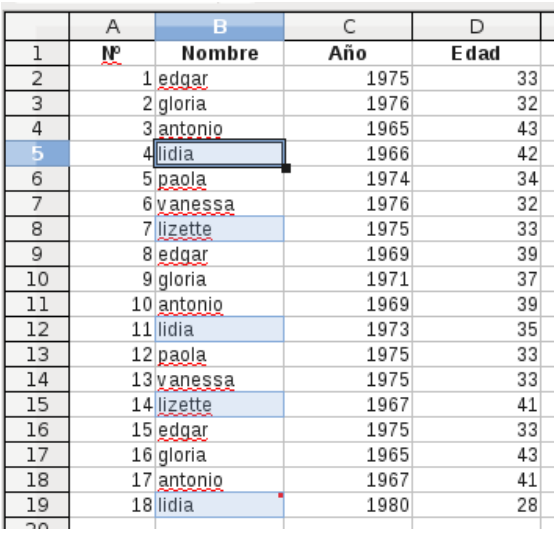
Ahora nos pasa al revés, devolvemos demasiados resultados, ¿verdad?, la razón es la misma, la búsqueda predeterminada se hace por formulas, observa que selecciono todas las celdas de la columna B que tienen la letra “a” y todas las celdas de la columna C y D, la formula de la columna D ya la vimos y contiene la letra buscada, la formula de la columna C es: =ALEATORIO.ENTRE(1965;1980), como vez, también tienen la letra “a” entre su texto, por ello, la búsqueda te devuelve todo, lo cual es correcto, la búsqueda esta bien, lo que tenemos que hacer es discriminar un poco más con los criterios de búsqueda, no necesariamente tiene que ser más restrictiva, sino acorde a lo que “estas buscando”, y claro, lo que esperas devolver, regresemos a nuestro primer ejemplo y agreguemos una propiedad a la búsqueda, ahora, le diremos que busque por valores, para que busque en el “resultado” de las formulas, no “dentro” de las formulas, esto lo logramos con la propiedad: searchType.

```
Sub Buscar3()
Dim oHojaActiva As Object
Dim oBuscarEn As Object
Dim oEncontrado As Object
Dim oSD As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oBuscarEn = oHojaActiva.getCellRangeByName( "A1:D19" )

   'Creamos el descriptor de busqueda
   oSD = oHojaActiva.createSearchDescriptor
   'Buscamos por valores
   oSD.searchType = 1
   'Establecemos lo que estamos buscando
   oSD.setSearchString( "33" )
   'Realizamos la busqueda de TODAS las coincidencias
   oEncontrado = oBuscarEn.findAll( oSD )
   'Si no hay coincidencias oEncontrado = Null
   If Not IsNull( oEncontrado ) Then
         'Si encuentra algo lo seleccionamos
         ThisComponent.getCurrentController.select( oEncontrado )
   Else
         MsgBox "No se encontraron coincidencias"
   End If

End Sub
```

Mucho mejor ¿verdad?, ahora si, solo las celdas con el valor 33 son seleccionadas, prueba a establecer la búsqueda con la letra “a” o cualquiera que tu quieras y observa los resultados, es muy importante que sepas que el resultado es una colección de rangos, aun,

cuando y solo te devuelva un solo rango, este, formará parte de dicha colección, en el tema Referencia a varios rangos, tratamos como manejar este tipo de rangos.

Como siguiente prueba, establece la búsqueda en oSD.setSearchString("li") y observa el resultado.



	A	B	C	D
1	Nº	Nombre	Año	Edad
2	1	edgar	1975	33
3	2	gloria	1976	32
4	3	antonio	1965	43
5	4	lidia	1966	42
6	5	paola	1974	34
7	6	vanessa	1976	32
8	7	lizette	1975	33
9	8	edgar	1969	39
10	9	gloria	1971	37
11	10	antonio	1969	39
12	11	lidia	1973	35
13	12	paola	1975	33
14	13	vanessa	1975	33
15	14	lizette	1967	41
16	15	edgar	1975	33
17	16	gloria	1965	43
18	17	antonio	1967	41
19	18	lidia	1980	28

Nota que las búsquedas las puede hacer en partes del texto, pero podemos restringirla a palabras completas con la propiedad searchWords, como en.

```
Sub Buscar4()
Dim oHojaActiva As Object
Dim oBuscarEn As Object
Dim oEncontrado As Object
Dim oSD As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oBuscarEn = oHojaActiva.getCellRangeByName( "A1:D19" )

   oSD = oHojaActiva.createSearchDescriptor
   oSD.searchType = 1
   'Buscamos por palabras completas
   oSD.searchWords = True
   oSD.setSearchString( "li" )

   oEncontrado = oBuscarEn.findAll( oSD )
   If Not IsNull( oEncontrado ) Then
         ThisComponent.getCurrentController.select( oEncontrado )
   Else
         MsgBox "No se encontraron coincidencias"
   End If

End Sub
```

La búsqueda anterior te tiene que devolver "No se encontraron coincidencias", por que no hay palabras completas que sean “li”, prueba a reemplazar la cadena buscada por el nombre que más de guste de los existentes en nuestra lista, por ejemplo “lizette” y obtendremos.

	A	B	C	D
1	Nº	Nombre	Año	Edad
2	1	edgar	1975	33
3	2	gloria	1976	32
4	3	antonio	1965	43
5	4	lidia	1966	42
6	5	paola	1974	34
7	6	vanessa	1976	32
8	7	lizette	1975	33
9	8	edgar	1969	39
10	9	gloria	1971	37
11	10	antonio	1969	39
12	11	lidia	1973	35
13	12	paola	1975	33
14	13	vanessa	1975	33
15	14	lizette	1967	41
16	15	edgar	1975	33
17	16	gloria	1965	43
18	17	antonio	1967	41
19	18	lidia	1980	28

Por que estamos haciendo la búsqueda por palabras completas. Podemos hacerla aun más restrictiva, si establecemos que distinga entre mayúsculas y minúsculas con la propiedad SearchCaseSensitive de la siguiente manera.

```
Sub Buscar5()
Dim oHojaActiva As Object
Dim oBuscarEn As Object
Dim oEncontrado As Object
Dim oSD As Object

   oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
   oBuscarEn = oHojaActiva.getCellRangeByName( "A1:D19" )

   oSD = oHojaActiva.createSearchDescriptor
   oSD.searchType = 1
   oSD.searchWords = False
   'Distinguimos mayusculas de minusculas
   oSD.SearchCaseSensitive = True
   oSD.setSearchString( "A" )
   oEncontrado = oBuscarEn.findAll( oSD )
   If Not IsNull( oEncontrado ) Then
         ThisComponent.getCurrentController.select( oEncontrado )
   Else
         MsgBox "No se encontraron coincidencias"
   End If

End Sub
```

Observa que hemos vuelto a establecer searchWords en False para que encuentre partes de las palabras. En la búsqueda anterior, únicamente te tiene que regresar la celda C1 que contiene la palabra “Año”. Incluso, puedes realizar búsquedas dentro de las notas de las celdas si estableces la propiedad searchType = 2.

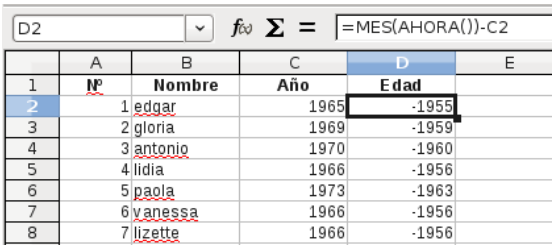
El reemplazo es la continuación de la búsqueda, es decir, para reemplazar algo, primero se tiene que buscar, veamos un ejemplo sencillo y después la explicación, continuamos haciendo uso de nuestra tabla usada en la búsqueda.

```
Sub Reemplazar1()
Dim oHojaActiva As Object
Dim oBuscarEn As Object
Dim lReemplazados As Long
Dim oRD As Object
```

```
    oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
    oBuscarEn = oHojaActiva.getCellRangeByName( "A1:D19" )

    'Creamos el descriptor de reemplazo
    oRD = oHojaActiva.createReplaceDescriptor
    'Texto a buscar
    oRD.setSearchString( "=año" )
    'Reemplazado por
    oRD.setReplaceString( "=mes" )
    'El método devuelve el numero de reemplazos que tuvieron éxito
    lReemplazados = oBuscarEn.replaceAll( oRD )

    If lReemplazados = 0 Then
          MsgBox "No se encontraron coincidencias"
    Else
          MsgBox "Se realizaron " & lReemplazados & " reemplazos"
    End If

End Sub
```

Observa que ahora, en ves de un descriptor de búsqueda (createSearchDescriptor), creamos un descriptor de reemplazo (createReplaceDescriptor), establecemos el valor buscado (setSearchString) y el valor por el que se reemplazara (setReplaceString), este método (replaceAll) devuelve un entero largo (long) con el número total de reemplazos que tuvieron éxito, si es cero no encontró coincidencias. En la columna D de nuestros datos de ejemplo, teníamos la formula =AÑO(AHORA())-C2, después de ejecutar la macro anterior, deberíamos tener en esta columna la formula =MES(AHORA())-C2, como lo podemos comprobar en la siguiente imagen.

Ahora, reemplazaremos palabras completas, con el siguiente código, mi amigo Edgar



	A	B	C	D	E
1	Nº	Nombre	Año	Edad	
2	1	edgar	1965	-1955	
3	2	gloria	1969	-1959	
4	3	antonio	1970	-1960	
5	4	lidia	1966	-1956	
6	5	paola	1973	-1963	
7	6	vanessa	1966	-1956	
8	7	lizette	1966	-1956	

no me reclamará nada si lo cambio por Nikole, de hecho me felicitará.

```
Sub Reemplazar2()
Dim oHojaActiva As Object
Dim oBuscarEn As Object
Dim lReemplazados As Long
Dim oRD As Object

    oHojaActiva = ThisComponent.getCurrentController().getActiveSheet()
    oBuscarEn = oHojaActiva.getCellRangeByName( "A1:D19" )

    oRD = oHojaActiva.createReplaceDescriptor
    oRD.setSearchString( "edgar" )
    oRD.setReplaceString( "nikole" )
    'Le indicamos que busque palabras completas
    oRD.searchWords = True

    lReemplazados = oBuscarEn.replaceAll( oRD )

    If lReemplazados = 0 Then
          MsgBox "No se encontraron coincidencias"
    Else
          MsgBox "Se realizaron " & lReemplazados & " reemplazos"
```

```
    End If

End Sub
```

A veces, es conveniente hacer primero una búsqueda y comprobar que esta correcta para después hacer el reemplazo, si tus búsquedas están bien establecidas, “casi” puedes estar seguro de que el reemplazo lo estará.

6.4.6 Trabajando con notas

Las notas, en las celdas de una hoja de calculo, son muy fáciles de manejar, en el siguiente ejemplo, insertamos una nueva nota en la celda E7.

```
Sub Notas1()
Dim oHojaActiva As Object
Dim oNotas As Object
Dim oDirCelda As New com.sun.star.table.CellAddress

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Obtenemos la colección de notas de la hoja activa
   oNotas = oHojaActiva.getAnnotations()
   'La celda donde insertaremos la nota
   oDirCelda.Column = 4
   oDirCelda.Row = 6
   'Insertamos la nota
   oNotas.insertNew( oDirCelda, "Presupuesto aprobado" )

End Sub
```

Para saber cuantas notas hay en la hoja activa.

```
Sub Notas2()
Dim oHojaActiva As Object
Dim oNotas As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Obtenemos la colección de notas de la hoja activa
   oNotas = oHojaActiva.getAnnotations()
   'Mostramos el total de notas en la hoja
   MsgBox oNotas.getCount()

End Sub
```

Para mostrar la dirección de cada una de las celdas con notas y su contenido.

```
Sub Notas3()
Dim oHojaActiva As Object
Dim oNotas As Object
Dim oNota As Object
Dim col As Long

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oNotas = oHojaActiva.getAnnotations()
   If oNotas.getCount() > 0 Then
         For col = 0 To oNotas.getCount - 1
```

```
            'Obtenemos una nota
            oNota = oNotas.getByIndex( col )
            'Mostramos su posición y su contenido
            MsgBox DireccionCelda( oNota.getPosition ) & ": " & oNota.getString()
        Next col
    Else
        MsgBox "No hay notas en esta hoja"
    End If

End Sub


Function DireccionCelda( DirCelda As Object ) As String
Dim oSFA As Object
Dim mDatos

    'Nos apoyamos en la función de Calc DIRECCION (ADDRESS)
    oSFA = createUnoService( "com.sun.star.sheet.FunctionAccess" )
    'Construimos la matriz de datos para la función
    mDatos = Array( DirCelda.Row+1, DirCelda.Column+1, 4 )
    DireccionCelda = oSFA.callFunction("ADDRESS",mDatos())

End Function
```

Puedes acceder a una celda primero y después a su nota para modificarla.

```
Sub Notas4()
Dim oHojaActiva As Object
Dim oCelda As Object
Dim oNota As Object

    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
    oCelda = oHojaActiva.getCellRangeByName("A1")
    'Accedemos a la nota de la celda
    oNota = oCelda.getAnnotation
    'Le cambiamos su contenido
    oNota.setString( "Cambio de texto en nota" )

End Sub
```

Si la celda referenciada no tiene una nota, el cambio no se verá reflejado en la interfaz del usuario, por lo que tienes que asegurarte que la celda contiene una nota, puedes usar la longitud del contenido de la nota para saberlo. Si no tiene nota, tienes que insertarla primero.

```
Sub Notas5()
Dim oHojaActiva As Object
Dim oCelda As Object
Dim oNota As Object

    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
    oCelda = oHojaActiva.getCellRangeByName("A1")
    'Accedemos a la nota de la celda
    oNota = oCelda.getAnnotation

    If Len( oNota.getString() ) = 0 Then
        MsgBox "La celda NO tiene nota"
    Else
        MsgBox "La celda tiene nota"
    End If

End Sub
```

También puede recorrer el conjunto de notas y comparar la dirección.

```
Sub Notas6()
Dim oHojaActiva As Object
Dim oCelda As Object
Dim oNotas As Object
Dim oNota As Object
Dim col As Long

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oCelda = oHojaActiva.getCellRangeByName("E7")
   oNotas = oHojaActiva.getAnnotations()
   If oNotas.getCount() > 0 Then
         For col = 0 To oNotas.getCount - 1
               oNota = oNotas.getByIndex( col )
               'Comparamos las direcciones
               If oNota.getPosition.Column = oCelda.getCellAddress.Column And oNota.getPosition.Row =
oCelda.getCellAddress.Row Then
                     MsgBox "La celda tiene nota"
                     Exit Sub
               End If
         Next col
         MsgBox "La celda NO tiene nota"
   Else
         MsgBox "No hay notas en esta hoja"
   End If

End Sub
```

Podemos hacer visible una nota.

```
Sub Notas7()
Dim oHojaActiva As Object
Dim oCelda As Object
Dim oNota As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oCelda = oHojaActiva.getCellRangeByName("E7")
   oNota = oCelda.getAnnotation
   'La hacemos visible
   oNota.setIsVisible( True )

End Sub
```

De nuevo, si la celda no tiene nota, el código anterior no hará nada, tampoco te dará ningún error, el siguiente ejemplo te intercambia la visibilidad de las notas de la hoja, es decir, si esta oculta la muestra y si esta visible la oculta.

```
Sub Notas8()
Dim oHojaActiva As Object
Dim oNotas As Object
Dim oNota As Object
Dim col As Long

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oNotas = oHojaActiva.getAnnotations()
   If oNotas.getCount() > 0 Then
         For col = 0 To oNotas.getCount - 1
               oNota = oNotas.getByIndex( col )
               'Intercambiamos su visibilidad
               oNota.setIsVisible( Not oNota.getIsVisible )
         Next col
```

```
    Else
          MsgBox "No hay notas en esta hoja"
    End If

End Sub
```

Nota como hacemos el intercambio, podemos saber si la nota de la celda es visible (*getIsVisible*), esta propiedad nos devuelve falso (*False*) o verdadero (*True*) según este o no visible la nota, con el operador de negación (Not) invertimos este valor y por consiguiente la visibilidad.

Para borrar una nota, es preciso saber el índice de esta, por lo que hay que iterar entre el conjunto de notas.

```
Sub Notas9()
Dim oHojaActiva As Object
Dim oCelda As Object
Dim oNotas As Object
Dim oNota As Object
Dim co1 As Long

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oCelda = oHojaActiva.getCellRangeByName("A1")
   oNotas = oHojaActiva.getAnnotations()
   If oNotas.getCount() > 0 Then
         For co1 = 0 To oNotas.getCount - 1
               oNota = oNotas.getByIndex( co1 )
               'Comparamos las direcciones
               If oNota.getPosition.Column = oCelda.getCellAddress.Column And oNota.getPosition.Row =
oCelda.getCellAddress.Row Then
                     'Borramos la nota por su índice
                     oNotas.removeByIndex( co1 )
                     Exit Sub
               End If
         Next co1
         MsgBox "La celda NO tiene nota"
   Else
         MsgBox "No hay notas en esta hoja"
   End If

End Sub
```

También es posible acceder a la autoforma (*Shape*) de la nota para manipularla.

```
Sub Notas10()
Dim oHojaActiva As Object
Dim oCelda As Object
Dim oNota As Object
Dim oForma As Object
Dim oTam As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oCelda = oHojaActiva.getCellRangeByName("E7")
   'Accedemos a la nota
   oNota = oCelda.getAnnotation
   'La hacemos visible
   oNota.setIsVisible( True )
   'Accedemos a la forma
   oForma = oNota.getAnnotationShape
   'Obtenemos el tamaño actual
   oTam = oForma.getSize
   'Lo duplicamos
```

```
    oTam.Width = oTam.Width * 2
    oTam.Height = oTam.Height * 2
    'Establecemos el nuevo tamaño
    oForma.setSize( oTam )
    'Cambiamos el color de fondo de forma aleatoria
    oForma.FillColor = RGB(Rnd()*255,Rnd()*255,Rnd()*255)

End Sub
```

Esta autoforma, tiene decenas de propiedades para manipularse, por lo que volveremos a ellas en el capítulo correspondiente a autoformas. Por ahora, tienes todos los elementos para manipular las notas de celdas a tu antojo. De tarea, crea una macro que nos haga un informe, en una hoja nueva, de todas las notas existentes en la hoja activa, nos tiene que devolver la dirección completa y su contenido, ¿vale?.

6.5 Dando formato

No se si estarás de acuerdo conmigo, pero a veces, dando formato a los datos se consume una gran cantidad de tiempo y ya sabes, uno que es bastante flojo esta encantado que se haga de forma automática, veamos las opciones más comunes de formato por código. En la mayoría de los ejemplos trataremos de trabajar sobre la selección, pero no se te olvide que siempre es bueno comprobar que estas trabajando en una hoja de calculo, así como en un rango de celdas cuando sea necesario, es decir, casi siempre.

6.5.1 Formato de celdas

Primero las características de formato más usuales, tipo de fuente, tamaño, negritas, cursiva y subrayado.

```
Sub FormatoCeldas1()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   With oSel
         'Tipo de letra
         .CharFontName = "Liberation Sans"
         'Tamaño de letra
         .CharHeight = 20
         'Negritas
         .CharWeight = 150
         'Cursiva o italica
         .CharPosture = 2
         'Subrayado
         .CharUnderline = 1
   End With
```

```
End Sub
```

Para saber la lista completa de fuentes (CharFontName) que tienes disponibles en OpenOffice.org, usa las macros disponibles en: Listar fuentes en un archivo de Calc

El tamaño de la fuente (CharHeight) se establece en puntos, recordando que un punto es igual a 1/72 de pulgada, aproximadamente 0.3528 mm.

El estilo negritas se establece por cualquiera de las siguiente constantes, de las cuales, excepto los extremos, no hay muchas variantes entre ellas.

Constante	*Valor*
com.sun.star.awt.FontWeight.DONTKNOW	0
com.sun.star.awt.FontWeight.THIN	50
com.sun.star.awt.FontWeight.ULTRALIGHT	60
com.sun.star.awt.FontWeight.LIGHT	75
com.sun.star.awt.FontWeight.SEMILIGHT	90
com.sun.star.awt.FontWeight.NORMAL	100
com.sun.star.awt.FontWeight.SEMIBOLD	110
com.sun.star.awt.FontWeight.BOLD	150
com.sun.star.awt.FontWeight.ULTRABOLD	175
com.sun.star.awt.FontWeight.BLACK	200

Al estilo cursiva o itálica le corresponden las siguiente constantes.

Constante	*Valor*
com.sun.star.awt.FontSlant.NONE	0
com.sun.star.awt.FontSlant.OBLIQUE	1
com.sun.star.awt.FontSlant.ITALIC	2
com.sun.star.awt.FontSlant.DONTKNOW	3
com.sun.star.awt.FontSlant.REVERSE_OBLIQUE	4
com.sun.star.awt.FontSlant.REVERSE_ITALIC	5

El tipo de subrayado esta determinado por las constantes.

Constante	*Valor*
com.sun.star.awt.FontUnderline.NONE	0
com.sun.star.awt.FontUnderline.SINGLE	1
com.sun.star.awt.FontUnderline.DOUBLE	2
com.sun.star.awt.FontUnderline.DOTTED	3
com.sun.star.awt.FontUnderline.DONTKNOW	4
com.sun.star.awt.FontUnderline.DASH	5
com.sun.star.awt.FontUnderline.LONGDASH	6
com.sun.star.awt.FontUnderline.DASHDOT	7
com.sun.star.awt.FontUnderline.DASHDOTDOT	8

Constante	Valor
com.sun.star.awt.FontUnderline.SMALLWAVE	9
com.sun.star.awt.FontUnderline.WAVE	10
com.sun.star.awt.FontUnderline.DOUBLEWAVE	11
com.sun.star.awt.FontUnderline.BOLD	12
com.sun.star.awt.FontUnderline.BOLDDOTTED	13
com.sun.star.awt.FontUnderline.BOLDDASH	14
com.sun.star.awt.FontUnderline.BOLDLONGDASH	15
com.sun.star.awt.FontUnderline.BOLDDASHDOT	16
com.sun.star.awt.FontUnderline.BOLDDASHDOTDOT	17
com.sun.star.awt.FontUnderline.BOLDWAVE	18

Recuerda que puedes usar tanto las constantes como el valor de estas, según tu gusto y criterio. Algunos de los valores de estas constantes, los puedes establecer pero no veras ningún efecto aparente.

Algunas otras opciones de formato de fuente son.

```
Sub FormatoCeldas2()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   With oSel
         'Espacios sin subrayar
         .CharWordMode = True
         'Tachado de fuente
         .CharCrossedOut = True
         'Contorno de fuente
         .CharContoured = True
   End With

End Sub
```

Cuando se subraya la fuente y la propiedad CharWordMode se establece en verdadera (True), te deja los espacios entre palabras sin subrayar. Las otras dos propiedades creo que son bastante autoexplicativas.

Ahora, veamos algo muy divertido, el color, establezcamos el color de la fuente y del subrayado de esta.

```
Sub FormatoCeldas3()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   With oSel
         'Color de la fuente azul
         .CharColor = RGB( 0,0,255 )
         'Estilo de subrayado
         .CharUnderline = 1
         'Color de subrayado rojo
         .CharUnderlineColor = RGB(255,0,0,)
         'Si se muestra el color
         .CharUnderlineHasColor = True
```

```
    End With

End Sub
```

Con la propiedad CharColor y la ayuda de la función RGB, establecemos el color de la fuente, para el subrayado, primero establecemos el estilo (CharUnderline), después el color de este (CharUnderlineColor) y por ultimo establecemos si queremos que se muestre el color (CharUnderlineHasColor), si esta propiedad se establece en falso (false), el color del subrayado será del mismo color que el de la fuente.

Con la siguiente macro, a cada celda del rango seleccionado se le establecerá un color de fuente y de subrayado aleatorio.

```
Sub FormatoCeldas4()
Dim oSel As Object
Dim col As Long
Dim fil As Long

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName = "ScCellRangeObj" Then
         With oSel
               For col = 0 To oSel.getColumns.getCount - 1
                     For fil = 0 To oSel.getRows.getCount - 1
                           With .getCellByPosition( col, fil )
                                 .CharColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
                                 .CharUnderline = 1
                                 .CharUnderlineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255)
                                 .CharUnderlineHasColor = True
                           End With
                     Next
               Next
         End With
   End If

End Sub
```

La alineación del contenido de la celda, se establece con las propiedades VertJustify para el sentido vertical y HoriJustify para el sentido horizontal de la siguiente forma.

```
Sub FormatoCeldas5()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   'Alineamos en medio verticalmente
   oSel.VertJustify = 2
   'Alineamos centrado horizontalmente
   oSel.HoriJustify = 2

End Sub
```

Los valores posibles para cada propiedad son:

Propiedad VertJustify	*Valor*
com.sun.star.table.CellVertJustify.STANDARD	0
com.sun.star.table.CellVertJustify.TOP	1
com.sun.star.table.CellVertJustify.CENTER	2

Propiedad VertJustify	*Valor*
com.sun.star.table.CellVertJustify.BOTTOM	3

Propiedad HoriJustify	*Valor*
com.sun.star.table.CellHoriJustify.STANDARD	0
com.sun.star.table.CellHoriJustify.LEFT	1
com.sun.star.table.CellHoriJustify.CENTER	2
com.sun.star.table.CellHoriJustify.RIGHT	3
com.sun.star.table.CellHoriJustify.BLOCK	4
com.sun.star.table.CellHoriJustify.REPEAT	5

Con la combinación de estos valores, tenemos muchas posibilidades de alineación de una celda. La constante com.sun.star.table.CellHoriJustify.REPEAT, es muy interesante por que te repite el valor de la celda hasta llenar el ancho de ella, prueba a establecer esta propiedad en una celda cuyo contenido sea un punto para que notes su comportamiento.

Si queremos ajustar el texto automáticamente en la celda, usamos.

```
Sub FormatoCeldas6()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   'Ajuste de texto
   oSel.IsTextWrapped = True

End Sub
```

Si queremos reducir el texto hasta que quepa en la celda, usamos.

```
Sub FormatoCeldas7()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   'Reducción de texto
   oSel.ShrinkToFit =True

End Sub
```

Si queremos cambiar la orientación del texto.

```
Sub FormatoCeldas8()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   'Orientacion vertical del texto
   oSel.Orientation = 3

End Sub
```

Los valores para esta propiedad son:

Propiedad Orientation	*Valor*
com.sun.star.table.CellOrientation.STANDARD	0
com.sun.star.table.CellOrientation.TOPBOTTOM	1
com.sun.star.table.CellOrientation.BOTTOMTOP	2
com.sun.star.table.CellOrientation.STACKED	3

O podemos establecer el ángulo de rotación.

```
Sub FormatoCeldas9()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   oSel.RotateAngle = 9000

End Sub
```

El valor del ángulo esta dado en centésimas de ángulo, es decir, en el ejemplo anterior establecemos el ángulo en 90º (9000) para 180º podemos 18000 y así sucesivamente.

Toma en cuenta que algunas de las propiedades vistas hasta ahora, se autoexcluyen mutuamente, por lo que tienes que asegurarte que no se contraponen, de los dos ejemplos siguiente, solo el segundo es correcto.

```
Sub FormatoCeldas10()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   oSel.HoriJustify = 2
   oSel.IsTextWrapped = True
   'Tamaño de sangría
   oSel.ParaIndent = 1000

End Sub

Sub FormatoCeldas11()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   oSel.HoriJustify = 1
   oSel.IsTextWrapped = True
   'Tamaño de sangría
   oSel.ParaIndent = 1000

End Sub
```

También podemos establecer el color de fondo de las celdas.

```
Sub FormatoCeldas12()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   'Color del fondo de la celda (gris)
   oSel.CellBackColor = RGB(200,200,200)

End Sub
```

La siguiente macro te colorea cada celda de la selección, tanto la fuente como el fondo de la celda de un color aleatorio:

```
Sub FormatoCeldas13()
Dim oSel As Object
Dim col As Long
Dim fil As Long

   oSel = ThisComponent.getCurrentSelection()

   If oSel.getImplementationName = "ScCellRangeObj" Then
         With oSel
               For col = 0 To oSel.getColumns.getCount - 1
                     For fil = 0 To oSel.getRows.getCount - 1
                           With .getCellByPosition( col, fil )
                                 .CharColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
                                 .CellBackColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
                           End With
                     Next
               Next
         End With
   End If

End Sub
```

Los bordes de las celdas tienen muchas posibilidades, veamos algunas.

```
Sub FormatoCeldas14()
Dim oSel As Object
Dim oBordeLinea As New com.sun.star.table.BorderLine

   oSel = ThisComponent.getCurrentSelection()

   'Establecemos las características de la linea
   With oBordeLinea
         'El color (rojo)
         .Color = RGB(255,0,0)
         'El ancho de la linea exterior
         .InnerLineWidth = 50
         'El ancho de la linea interior
         .OuterLineWidth = 50
         'La distancia entre las lineas
         .LineDistance = 20
   End With

   'Establecemos los bordes
   With oSel
         .TopBorder = oBordeLinea          'Superior
         .BottomBorder = oBordeLinea       'Inferior
         .LeftBorder = oBordeLinea         'Izquierdo
         .RightBorder = oBordeLinea        'Derecho
   End With

End Sub
```

El ancho de las lineas y la distancia entre ellas esta dada en centésimas de milímetros, si establecemos solo una de las dos líneas, veras una linea simple. Si deseas establecer también las líneas diagonales, usa la propiedad DiagonalTLBR para la línea que va desde la esquina superior izquierda a la inferior derecha y DiagonalBLTR para la que va de la esquina inferior izquierda a la superior derecha, tienes que establecerles la misma estructura de línea (com.sun.star.table.BorderLine) vista en los ejemplos.

El código anterior te establecerá el borde de cada celda del rango seleccionado, si quieres establecer el borde de solo los extremos del rango, tiene que usar la estructura TableBorder.

```
Sub FormatoCeldas15()
Dim oSel As Object
Dim oBordeLinea As New com.sun.star.table.BorderLine
Dim oBordeTabla As New com.sun.star.table.TableBorder

   oSel = ThisComponent.getCurrentSelection()

   'Establecemos las características de la linea
   With oBordeLinea
         'El color (azul)
         .Color = RGB(0,0,255)
         'El ancho de la linea interior
         .OuterLineWidth = 75
   End With

   'Establecemos las características de la tabla
   With oBordeTabla
         .TopLine = oBordeLinea            'Superior
         .IsTopLineValid = True
         .BottomLine = oBordeLinea         'Inferior
         .IsBottomLineValid = True
         .LeftLine = oBordeLinea           'Izquierdo
         .IsLeftLineValid = True
         .RightLine = oBordeLinea          'Derecho
         .IsRightLineValid = True
   End With

   'Establecemos los bordes
   oSel.TableBorder = oBordeTabla

End Sub
```

Las propiedades IsTopLineValid, IsBottomLineValid, IsLeftLineValid y IsRightLineValid, tienes que establecerlas en verdadero (true) si quieres que se vea la línea.

Para establecer los margenes de la celda, usamos.

```
Sub FormatoCeldas16()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   'Para establecer los margenes de la celda
   With oSel
         .ParaTopMargin = 500              'Arriba
         .ParaBottomMargin = 500           'Abajo
         .ParaLeftMargin = 800       'Izquierda
         .ParaRightMargin = 800      'Derecha
   End With

End Sub
```

Para dar un formato predeterminado a la celda, usamos la propiedad NumerFormat, la cual es necesario establecer como un entero largo (long) correspondiente a la clave del formato que nos interese, por ejemplo.

```
Sub FormatoCeldas17()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   'Establecemos la selección en el formato Estándar
   oSel.NumberFormat = 0

End Sub
```

El numero de clave no es secuencial, por lo que tenemos que saber de antemano la clave del formato que nos interesa, para ello, usa la macro que te muestro en: Listar formatos en un archivo de Calc.

Ahora veamos como combinar y separar celdas. Para ello, usamos el método ***merge***, con el argumento verdadero (true), como en el siguiente ejemplo.

```
Sub FormatoCeldas18()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   'Combinamos las celdas seleccionadas
   oSel.merge( True )

End Sub
```

Este método solo esta presente en un rango de celdas, no en una colección de rangos así que validalo cuando lo uses. Si estableces el argumento en falso (false), separara las celdas de la selección. Este método es sencillo pero debes de tomar en cuenta algunas consideraciones. Si seleccionas un rango ya combinado, junto con un área que no, este método no hará nada, claro, por código estoy hablando, pero has la prueba en la interfaz del usuario, es decir, manualmente y observa que pasa, ahora, tratemos de emularlo con código.

```
Sub FormatoCeldas19()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   'Separamos las celdas seleccionadas
   oSel.merge( False )
   'Las combinamos
   oSel.merge( True )

End Sub
```

Si se puede, ¿verdad?, primero las separamos y después las combinamos. La siguiente consideración que debes tomar en cuenta, es saber que pasa cuando las celdas se combinan, para ello, haremos unas pruebas ya no con la selección sino con rangos con nombre, la macro siguiente te alternara entre combinar y separar la el rango seleccionado, nota como consultamos la propiedad getIsMerged para saber si el rango esta combinado.

```
Sub FormatoCeldas20()
Dim oRango As Object
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   oRango = oHojaActiva.getCellRangeByName( "B2:D10" )

   'Invertimos la combinación
   oRango.merge( Not oRango.getIsMerged )
```

```
End Sub
```

Ahora, en la siguiente macro, tienes que poner mucha atención, nota el rango que estamos combinando, las celdas en las cuales verificamos si están combinadas, el valor que nos devuelve el consultar su propiedad getIsMerged y observa hasta cuando nos funciona el método para separar las celdas.

```
Sub FormatoCeldas21()
Dim oRango As Object
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName( "E15:H20" )
   oRango.merge( True )

   'Nota que la celda G19 forma parte del rango combinado
   'consultamos la propiedad y tratamos de separar
   Msgbox oHojaActiva.getCellRangeByName( "G19" ).getIsMerged()
   oHojaActiva.getCellRangeByName( "G19" ).merge( False )

   Msgbox oHojaActiva.getCellRangeByName( "F17" ).getIsMerged()
   'Tratamos de separar las celdas
   oHojaActiva.getCellRangeByName( "G19" ).merge( False )

   Msgbox oHojaActiva.getCellRangeByName( "E16" ).getIsMerged()
   oHojaActiva.getCellRangeByName( "G19" ).merge( False )

   'Ahora si, nos dice que esta combinada y
   'finalmente logramos separarlas
   Msgbox oHojaActiva.getCellRangeByName( "E15" ).getIsMerged()
   oHojaActiva.getCellRangeByName( "E15" ).merge( False )

End Sub
```

¿Lo notaste?, cuando se combinan celdas, todas, excepto la de la esquina superior izquierda del rango, se marcan como “no” combinadas, entonces, eso nos obliga a buscar y encontrar esta celda a partir de otra cuando intentemos separar celdas, para hacer esto, nos auxiliaremos de un cursor cuyo uso ya lo hemos visto.

```
Sub FormatoCeldas22()
Dim oRango As Object
Dim oHojaActiva As Object
Dim oCursor As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName( "E15:H20" )
   oRango.merge( True )

   'Verificamos que este combinada
   MsgBox oHojaActiva.getCellRangeByName( "E15" ).getIsMerged()

   'Creamos un cursor a partir de una celda que no sea la de
   'la esquina superior izquierda
   oCursor = oHojaActiva.createCursorByRange( oHojaActiva.getCellRangeByName( "G19" ) )
   'Expandimos el cursor al área combinada
   oCursor.collapseToMergedArea()
   'Separamos las celdas
   oCursor.merge( False )

End Sub
```

Esta vez, a partir de otra celda, expandimos el cursor a la región combinada y pudimos separar fácilmente. Toma en cuenta que pasa con las celdas que tienen contenido cuando se combinan, verificalo en la interfaz del usuario y trata de anticipar este caso.

Como ultimo tema de este capitulo, veamos como proteger y desproteger celdas, recuerda que esta protección solo es efectiva si la hoja de calculo esta protegida.

```
Sub ProtegerCeldas1()
Dim oRango As Object
Dim oHojaActiva As Object
Dim oPC As New "com.sun.star.util.CellProtection"

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName( "J27:L30" )

   With oPC
         'Determina si la celda esta bloqueada para modificarse
         .IsLocked = True
         'Determina si se ocultan las formulas, solo veras el resultado
         .IsFormulaHidden = True
         'Determina si se oculta el contenido de las celdas
         .IsHidden = True
         'Para ocultar solo para la impresión
         .IsPrintHidden = True
   End With
   oRango.CellProtection = oPC
   oHojaActiva.Protect("")

End Sub
```

Para desproteger las celdas, es suficiente con desproteger la hoja, de todos modos puedes deshabilitar las opciones de protección con solo establecer las opciones de protección en falso (False), por supuesto, no siempre tienen que estar todas seleccionadas, tu determinaras el nivel de acceso en cada uno de tus proyecto, por ejemplo.

```
Sub ProtegerCeldas2()
Dim oRango As Object
Dim oHojaActiva As Object
Dim oPC As New "com.sun.star.util.CellProtection"

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName( "J27:L30" )
   'Desprotegemos la hoja
   oHojaActiva.unProtect("")
   With oPC
         'Determina si la celda esta bloqueada para modificarse
         .IsLocked = True
         'Determina si se ocultan las formulas, solo veras el resultado
         .IsFormulaHidden = True
         'Determina si se oculta el contenido de las celdas
         .IsHidden = True
         'Para ocultar solo para la impresión
         .IsPrintHidden = True
   End With
   oRango.CellProtection = oPC

End Sub
```

6.5.2 Formato de filas y columnas

Para establecer la altura de las filas, usamos la propiedad Height, el valor se establece en centésimas de milímetros, por lo que para tener una fila de 0.50 cm le asignamos un valor de 500 como en el siguiente ejemplo.

```
Sub FormatoFilas1()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   'Establecemos el alto de las filas seleccionadas
   oSel.getRows.Height = 500

End Sub
```

Cuando tratas de establecer la altura de las filas en cero (0), si bien no te dará ningún error, tampoco veras ningún cambio, aun, estableciendo el valor en la unidad veras que la fila sigue siendo notable. Si lo que deseas es ocultarla, es mejor usar la propiedad correspondiente, para este caso, IsVisible, como en.

```
Sub FormatoFilas2()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   'Ocultamos las filas seleccionadas
   oSel.getRows.IsVisible = False

End Sub
```

Esta propiedad se establece en falso (false) para ocultar la fila y en verdadero (true) para mostrarla, el siguiente ejemplo te alternara en ocultar y mostrar conforme la ejecutes.

```
Sub FormatoFilas3()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   'Alternamos entre visible y oculta
   oSel.getRows.IsVisible = Not oSel.getRows.IsVisible

End Sub
```

Las filas, adquieren una propiedad particular cuando se les aplica un filtro, esta propiedad se llama IsFiltered, y te devolverá verdadero (true), cuando la fila consultada este oculta como consecuencia de un filtro, toma nota de que “solo” cuando este oculta por un filtro, si tu la ocultas con IsVisible, esta propiedad (IsFiltered) seguirá siendo falsa (false), prueba a hacer un filtro en tu hoja activa y procura que la fila consultada (getByIndex(Indice)) este filtrada.

```
Sub FormatoFilas4()
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Consultamos si la fila 3 esta filtrada
   MsgBox oHojaActiva.getRows.getByindex(2).IsFiltered

End Sub
```

También podemos establecer el alto óptimo de las filas con la propiedad OptimalHeight, debe estar en verdadero (true) para que tenga efecto.

```
Sub FormatoFilas5()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   'Establecemos el alto optimo de las filas seleccionadas
   oSel.getRows.OptimalHeight = True

End Sub
```

A las filas también es posible cambiarles el color de fondo, esto nos permite establecer el color de toda la fila.

```
Sub FormatoFilas6()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   'Establecemos el color de fondo de las celdas de la fila completa
   oSel.getRows.CellBackColor = RGB(200,200,200)

End Sub
```

La siguiente macro te cambiara el color de las filas impares de la selección.

```
Sub FormatoFilas7()
Dim oSel As Object
Dim co1 As Long

   oSel = ThisComponent.getCurrentSelection()

   For co1 = 0 To oSel.getRows.getCount() - 1 Step 2
         'Establecemos un color aleatorio para las filas impares de la selección
         oSel.getRows.getByIndex( co1 ).CellBackColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   Next

End Sub
```

A las columnas también les podemos establecer el ancho.

```
Sub FormatoColumnas1()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   'Establecemos el ancho de las columnas seleccionadas
   oSel.getColumns.Width = 1000

End Sub
```

Y ocultarlas.

```
Sub FormatoColumnas2()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   'Ocultamos las columnas seleccionadas
```

```
   oSel.getColumns.IsVisible = False

End Sub
```

Ajustar al contenido.

```
Sub FormatoColumnas3()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()

   'Ajustamos al contenido
   oSel.getColumns.OptimalWidth = True

End Sub
```

Las columnas no tienen una propiedad IsFiltered, por que como sabes, no hay filtros horizontales, pero con un poco de ingenio y usando la propiedad IsVisible, puedes crearte tu versión de filtros horizontales. Tampoco puedes usar la propiedad CellBackColor para establecer el color de fondo, pero también podemos implementar nuestra versión, aquí una primera aproximación.

```
Sub FormatoColumnas4()
Dim oSel As Object
Dim oCursor As Object

   oSel = ThisComponent.getCurrentSelection()
   'Nos apoyamos en un cursor
   oCursor = oSel.getSpreadSheet.createCursorByRange( oSel )
   oCursor.expandToEntireColumns()
   oCursor.CellBackColor = RGB(210,210,210)

End Sub
```

La siguiente macro, establecerá el ancho de las columnas en 1 cm, después el alto de las filas también en 1 y coloreará alternadamente filas y columnas para hacer un lindo mosaico.

```
Sub FormatoFilasColumnas1()
Dim oSel As Object
Dim oCursor As Object
Dim co1 As Long

   oSel = ThisComponent.getCurrentSelection()

   oSel.getRows.Height = 1000
   oSel.getColumns.Width = 1000

   For co1 = 0 To oSel.getRows.getCount() - 1 Step 2
         oSel.getRows.getByIndex( co1 ).CellBackColor = RGB( 230,230,230 )
   Next

   For co1 = 0 To oSel.getColumns.getCount() - 1 Step 2
         oCursor = oSel.getSpreadSheet.createCursorByRange( oSel.getCellByPosition(co1,0) )
         oCursor.expandToEntireColumns()
         oCursor.CellBackColor = RGB(230,230,230)
   Next

End Sub
```

Las filas y columnas cuentan con algunas propiedades involucradas en la impresión de los documentos, por lo que se verán en el capitulo correspondiente a impresión.

6.5.3 Estilos y autoformato

Como buen usuario de OpenOffice.org que somos, sabemos que una de sus características más ricas esta en sus “estilos”, es decir, en la posibilidad de agrupar bajo un nombre, una serie de características de formato y que podemos aplicar con un solo clic. Dentro de las hojas de cálculo, tenemos dos estilos perfectamente definidos, los estilos de celda que veremos en este capitulo y los estilos de página que se verán en el siguiente. Para ver los estilos de celda actuales usamos la siguiente macro.

```
Sub Estilos1()
Dim oDoc As Object
Dim oEstilos As Object
Dim oEstilosCelda As Object
Dim oSel As Object

   'Accedemos al documento actual
   oDoc = ThisComponent
   'Accedemos a todos los estilos
   oEstilos = oDoc.getStyleFamilies()
   'Accedemos a los estilos de celda
   oEstilosCelda = oEstilos.getByName("CellStyles")
   'Mostramos sus nombres
   MsgBox Join( oEstilosCelda.getElementNames(), Chr(13) )

End Sub
```

Si no tienes ningún estilo de celda personalizado, crea uno con el formato que quieras (tecla F11), vuelve a correr la macro y tienes que ver listado el nuevo estilo.

En nuestro ejemplo, hemos agregado un nuevo estilo de celda que se llama “Domingos”, los estilos de celda los podemos establecer a una celda, un rango de celdas, un grupo de rangos de una forma sumamente sencilla, veamos como.

```
Sub Estilos2()
Dim oSel As Object
```

```
    oSel = ThisComponent.getCurrentSelection()
    'Aplicamos el estilo de celda Domingos a la selección actual
    oSel.CellStyle = "Domingos"

End Sub
```

Veamos la diferencia entre aplicar formato directo a las celdas y usar un estilo, en la siguiente macro, formateamos la selección con las características indicada en cada línea.

```
Sub Estilos3()
Dim oSel As Object

    oSel = ThisComponent.getCurrentSelection()

    With oSel
          'Tipo de letra
          .CharFontName = "Nimbus Sans L"
          'Tamaño de letra
          .CharHeight = 15
          'Negritas
          .CharWeight = 150
          'Cursiva o italica
          .CharPosture = 2
          'Alineamos en medio verticalmente
          .VertJustify = 2
          'Alineamos centrado horizontalmente
          .HoriJustify = 2
          'Color de fondo de la celda
          .CellBackColor = RGB(204,204,204)
          'Color de la fuente
          .CharColor = RGB(0,0,255)
    End With

End Sub
```

Ahora, crea un estilo con estas mismas características, para nuestro ejemplo le llamaremos “Resaltado1” y procedamos a aplicar a la selección.

```
Sub Estilos4()
Dim oSel As Object

    oSel = ThisComponent.getCurrentSelection()
    oSel.CellStyle = "Resaltado1"

End Sub
```

No puede ser más fácil, pero (siempre hay un pero), si el estilo de celda no existe, el código no te dará ningún error, simplemente no hará nada, el nombre del estilo, debe estar escrito tal cual se dio de alta incluyendo mayúsculas y minúsculas, también, como sucede en la interfaz del usuario, cuando se borra un estilo de celda personalizado, que son los únicos que puedes eliminar, las celdas que tengan este estilo, pasarán a tener el estilo de celda predeterminado, por ello, es recomendable, verificar que el estilo exista y en caso de que no exista, crearlo, al crearlo, es recomendable verificar que no exista ya el nombre propuesto, pues si intentas agregar un nombre ya existente, te dará un error en tiempo de ejecución. Veamos un ejemplo.

```
Sub Estilos5()
Dim oDoc As Object
Dim oSel As Object
```

```
Dim oEstilos As Object
Dim oEstilosCelda As Object
Dim oEstiloNuevo As Object

   'Accedemos al documento actual
   oDoc = ThisComponent
   'Accedemos a la selección
   oSel = ThisComponent.getCurrentSelection()
   'Accedemos a todos los estilos
   oEstilos = oDoc.getStyleFamilies()
   'Accedemos a los estilos de celda
   oEstilosCelda = oEstilos.getByName("CellStyles")
   'Verificamos que el estilo exista
   If oEstilosCelda.hasByName( "Resaltado1" ) Then
         oSel.CellStyle = "Resaltado1"
   Else
         'Si no existe lo creamos
         oEstiloNuevo = oDoc.createInstance( "com.sun.star.style.CellStyle" )
         'Lo agregamos a la colección de estilos
         oEstilosCelda.insertByName( "Resaltado1", oEstiloNuevo )
         'Establecemos su formato
         With oEstiloNuevo
               .CharFontName = "Nimbus Sans L"
               .CharHeight = 15
               .CharWeight = 150
               .CharPosture = 2
               .VertJustify = 2
               .HoriJustify = 2
               .CellBackColor = RGB(204,204,204)
               .CharColor = RGB(0,0,255)
         End With
         'Y lo aplicamos
         oSel.CellStyle = "Resaltado1"
   End If

End Sub
```

Para borrar un estilo, usamos el método removeByName de la siguiente manera, recordando que al borrar el estilo, “todas”, las celdas que tengan dicho estilo, regresaran al estilo predeterminado, perdiéndose todo el formato.

```
Sub Estilos6()
Dim oDoc As Object
Dim oEstilos As Object
Dim oEstilosCelda As Object
Dim oEstiloNuevo As Object

   'Accedemos al documento actual
   oDoc = ThisComponent
   'Accedemos a todos los estilos
   oEstilos = oDoc.getStyleFamilies()
   'Accedemos a los estilos de celda
   oEstilosCelda = oEstilos.getByName("CellStyles")
   'Verificamos que el estilo exista
   If oEstilosCelda.hasByName( "Domingos" ) Then
         If MsgBox("¿Estas seguro de borrar el estilo?",36,"Borra estilo") = 6 then
               'Quitamos el estilo
               oEstilosCelda.removeByName( "Domingos" )
               MsgBox "Estilo de celda borrado"
         Else
               MsgBox "NO se borro ningún estilo"
         End If
   Else
```

```
        MsgBox "El estilo no existe"
    End If

End Sub
```

El manejo de autoformato, es muy similar al de estilos, la diferencia, es que el autoformato siempre se aplica a un rango de celdas, mínimo, de tres columnas por tres filas, este servicio esta disponible tanto en hojas de calculo como en tablas de Writer, para ver los autoformatos disponibles usamos.

```
Sub Autoformato1()
Dim oAutoFormatos As Object

   oAutoFormatos = createUnoService("com.sun.star.sheet.TableAutoFormats")
   'Mostramos sus nombres
   MsgBox Join( oAutoFormatos.getElementNames(), Chr(13) )

End Sub
```

Para aplicar un autoformato se usa el método autoFormat de la siguiente manera.

```
Sub Autoformato2()
Dim oSel As Object

   oSel = ThisComponent.getCurrentSelection()
   oSel.autoFormat("Verde")

End Sub
```

Toma en cuenta que el rango seleccionado, “debe” ser un objeto ScCellRangeObj, es decir, un rango de celdas, si bien, por código, podrás aplicarlo a una sola celda pero te formateará un mínimo de tres fila por tres columnas, si seleccionas varios rangos de celdas, te dará un error pues ese objeto no implementa el método autoFormat, lo mejor, como siempre, es validar que la selección es correcta, si usas esta técnica. El nombre del autoformato, como en los estilos, debe ser exactamente como se llamó al crearlo, pero a diferencia de los estilos, si escribes mal el nombre, te dará un error en tiempo de ejecución, por ello, es mejor validar que exista, antes de usarlo, como en el siguiente ejemplo.

```
Sub Autoformato3()
Dim oAutoFormatos As Object
Dim oSel As Object
Dim sNombreAF As String

   sNombreAF = "VerdeRojo"
   oAutoFormatos = createUnoService("com.sun.star.sheet.TableAutoFormats")
   'Verificamos que exista el autoformato
   If oAutoFormatos.hasByName( sNombreAF ) Then
        oSel = ThisComponent.getCurrentSelection()
        'Nos aseguramos que la seleccion sea un rango de celdas
        If oSel.getImplementationName = "ScCellRangeObj" Then
             'de mínimo tres filas por tres columnas
             If oSel.getColumns.getCount > 2 And oSel.getRows.getCount > 2 Then
                  oSel.autoFormat( sNombreAF )
             Else
                  MsgBox "El rango debe ser de minimo, 3 columnas por 3 filas"
             End If
        Else
             MsgBox "No es un rango de celdas"
        End If
```

```
   Else
         MsgBox "El autoformato no existe"
   End If

End Sub
```

Para eliminar un autoformato, usamos el método removeByName como en.

```
Sub Autoformato4()
Dim oAutoFormatos As Object

   oAutoFormatos = createUnoService("com.sun.star.sheet.TableAutoFormats")
   'Verificamos que exista
   If oAutoFormatos.hasByName( "VerdeRojo" ) Then
         If MsgBox("¿Estas seguro de borrar el autoformato?",36,"Borra autoformato") = 6 Then
               'Quitamos el autoformato
               oAutoFormatos.removeByName( "VerdeRojo" )
               MsgBox "Autoformato borrado"
         Else
               MsgBox "NO se borro nada"
         End If
   Else
         MsgBox "El autoformato no existe"
   End If

End Sub
```

A diferencia de los estilos de celda, cuando borras un autoformato, las celdas establecidas con este permanecen con dicho formato, pero el autoformato borrado ya no estará disponible para usarse, por lo que tenemos que crearlo de nuevo. Para crear un autoformato, primero hay que saber (y entender) como esta estructurado, observa la siguiente tabla.

	1ª Columna	1ª Columna de datos	2ª Columna de datos	Ultima Columna
Encabezado 1a Fila	0	1	2	3
1a Fila de datos	4	5	6	7
2a Fila de datos	8	9	10	11
Pie de Tabla Ultima Fila	12	13	14	15

Un autoformato, no es más que una colección de formatos de celdas individuales, cada cruce de columna y fila representa un estilo de celda, que, como podrás notar, tenemos 16 estilos diferentes que podemos establecer. Como ya lo mencionamos, el área mínima para usar un autoformato, es de 3 x 3 celdas, si tu rango tiene más área, el formato de las columnas de datos se ira duplicando y alternando, lo mismo pasará con las filas en el sentido vertical. Observa la siguiente imagen, en la parte superior, están nuestros datos sin formato (“pelones” dicen en mi pueblo), y en la parte inferior, como es que queremos que quede, como podrás notar, soy una nulidad para el diseño, así que no te pongas muy estricto.

	B	C	D	E	F	G	H	I
36								
37		Año	2005	2006	2007	2008	2009	Totales
38		Contabilidad	76081.82	68164.15	46077.44	49390.86	15366.01	255080.29
39		Informática	96887.99	60834.95	42601.31	81023.95	57478.21	338826.41
40		Administración	24012.66	37464.43	56335.85	106617.89	109821.11	334251.95
41		Servicio	90399.05	104747.58	101754.58	51378.32	23100.28	371379.8
42		Gerencia	98312.58	30369.52	79851.07	39156.52	43670.63	291360.32
43		Sumas	385694.11	301580.64	326620.25	327567.54	249436.25	
44								
45								
46		**Año**	**2005**	**2006**	**2007**	**2008**	**2009**	**Totales**
47		**Contabilidad**	$76,081.82	$68,164.15	$46,077.44	$49,390.86	$15,366.01	**$255,080.29**
48		**Informática**	**$96,887.99**	**$60,834.95**	**$42,601.31**	**$81,023.95**	**$57,478.21**	**$338,826.41**
49		**Administración**	$24,012.66	$37,464.43	$56,335.85	$106,617.89	$109,821.11	**$334,251.95**
50		**Servicio**	**$90,399.05**	**$104,747.58**	**$101,754.58**	**$51,378.32**	**$23,100.28**	**$371,379.80**
51		**Gerencia**	$98,312.58	$30,369.52	$79,851.07	$39,156.52	$43,670.63	**$291,360.32**
52		**Sumas**	**$385,694.11**	**$301,580.64**	**$326,620.25**	**$327,567.54**	**$249,436.25**	
53								

Para crear el autoformato anterior, usamos la siguiente macro, te recuerdo que el formato de cada celda (campo) del autoformato, tiene las mismas características de formato de las celdas vistas anteriormente (aunque no es posible aplicar todos los formatos presentes normalmente en celdas), por lo que no comentaré esa parte, vamos a formatear 16 celdas, así que será una macro un poco más larga de las que hemos venido probando, pero muy sencilla por que se repite bastante el código, veamos.

```
Sub CrearAutoFormato()
Dim oAutoFormatos As Object
Dim oNuevoAF As Object
Dim sNombreAF As String
Dim oDoc As Object
Dim oSel As Object
Dim oBordeLinea As New com.sun.star.table.BorderLine
Dim oBordeTabla As New com.sun.star.table.TableBorder

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   oNuevoAF = oDoc.createInstance("com.sun.star.sheet.TableAutoFormat")
   oAutoFormatos = createUnoService("com.sun.star.sheet.TableAutoFormats")
   sNombreAF = "MiFormatoNuevo"

   'Verificamos que no exista
   If Not oAutoFormatos.hasByName( sNombreAF ) Then
         'Lo agregamos a la colección de AutoFormatos
         oAutoFormatos.insertByName( sNombreAF, oNuevoAF )
         'Le damos formato a cada celda
         With oBordeLinea
               .Color = 255
               .OuterLineWidth = 60
         End With
         With oBordeTabla
               .TopLine = oBordeLinea
               .IsTopLineValid = True
               .BottomLine = oBordeLinea
               .IsBottomLineValid = True
               .LeftLine = oBordeLinea
               .IsLeftLineValid = True
               .RightLine = oBordeLinea
               .IsRightLineValid = True
         End With

         'Celda 0
         Call FormatearCampoAF( oNuevoAF.getByIndex(0), 16711680, 16777215, "Liberation Sans", 13,
oBordeTabla, 150, 2, 2 )
```

```
        'Celda 1
        Call FormatearCampoAF( oNuevoAF.getByIndex(1), 16711680, 16777215, "Liberation Sans", 13,
oBordeTabla, 150, 2, 2 )
        'Celda 2
        Call FormatearCampoAF( oNuevoAF.getByIndex(2), 16711680, 16777215, "Liberation Sans", 13,
oBordeTabla, 150, 2, 2 )
        'Celda 3
        Call FormatearCampoAF( oNuevoAF.getByIndex(3), 16711680, 16777215, "Liberation Sans", 13,
oBordeTabla, 150, 2, 2 )
        'Celda 4
        Call FormatearCampoAF( oNuevoAF.getByIndex(4), 13421772, 2621593, "Liberation Sans", 12,
oBordeTabla, 150, 3, 2 )
        'Celda 5
        Call FormatearCampoAF( oNuevoAF.getByIndex(5), 13421772, 0, "Liberation Sans", 10,
oBordeTabla, 100, 0, 2 )
        'Celda 6
        Call FormatearCampoAF( oNuevoAF.getByIndex(6), 13421772, 0, "Liberation Sans", 10,
oBordeTabla, 100, 0, 2 )
        'Celda 7
        Call FormatearCampoAF( oNuevoAF.getByIndex(7), 2621593, 16777215, "Liberation Sans", 12,
oBordeTabla, 150, 0, 2 )
        'Celda 8
        Call FormatearCampoAF( oNuevoAF.getByIndex(8), 13421772, 2621593, "Liberation Sans", 12,
oBordeTabla, 150, 3, 2 )
        'Celda 9
        Call FormatearCampoAF( oNuevoAF.getByIndex(9), 16764057, 0, "Liberation Sans", 10,
oBordeTabla, 150, 0, 2 )
        'Celda 10
        Call FormatearCampoAF( oNuevoAF.getByIndex(10), 16764057, 0, "Liberation Sans", 10,
oBordeTabla, 150, 0, 2 )
        'Celda 11
        Call FormatearCampoAF( oNuevoAF.getByIndex(11), 2621593, 16777215, "Liberation Sans", 12,
oBordeTabla, 150, 0, 2 )
        'Celda 12
        Call FormatearCampoAF( oNuevoAF.getByIndex(12), 2621593, 16777215, "Liberation Sans", 12,
oBordeTabla, 150, 3, 2 )
        'Celda 13
        Call FormatearCampoAF( oNuevoAF.getByIndex(13), 2621593, 16777215, "Liberation Sans", 12,
oBordeTabla, 150, 0, 2 )
        'Celda 14
        Call FormatearCampoAF( oNuevoAF.getByIndex(14), 2621593, 16777215, "Liberation Sans", 12,
oBordeTabla, 150, 0, 2 )
        'Celda 15
        Call FormatearCampoAF( oNuevoAF.getByIndex(15), 13421772, 0, "Liberation Sans", 10,
oBordeTabla, 150, 0, 2 )

        'De forma predeterminada incluimos todos las opciones
        With oNuevoAF
              .IncludeNumberFormat = True
              .IncludeBackground = True
              .IncludeBorder = True
              .IncludeFont = True
              .IncludeJustify = True
              .IncludeNumberFormat = True
              .IncludeWidthAndHeight = True
        End With

        'Solo lo aplicamos si la selección es un rango de celdas
        If oSel.getImplementationName = "ScCellRangeObj" Then
              oSel.autoFormat( sNombreAF )
        End If
   Else
        'Si existe el formato lo aplicanos
        If oSel.getImplementationName = "ScCellRangeObj" Then
              oSel.autoFormat( sNombreAF )
        End If
```

```
   End If

End Sub

'Puedes agregar todas las propiedades que consideres
Sub FormatearCampoAF(Campo As Object, ColorFondo As Long, ColorFuente As Long, Fuente As String,
TamFuente As Integer, Bordes As Object, Negritas As Integer, HJ As Byte, VJ As Byte)

   With Campo
         .CellBackColor = ColorFondo
         .CharColor = ColorFuente
         .CharFontName = Fuente
         .CharHeight = TamFuente
         .TableBorder = Bordes
         .CharWeight = Negritas
         .HoriJustify = HJ
         .VertJustify = VJ
   End With
End Sub
```

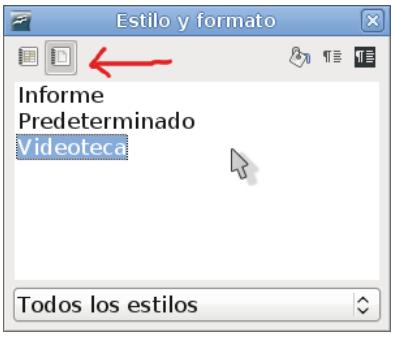
La macro anterior tiene unas pequeñas diferencias con la imagen mostrada al inicio de ella, tu tarea es notar las diferencias, así mismo, tiene una pequeña deficiencia que también tienes que encontrar, no esta difícil, solo tienes que ser un poco observador. Observa que hacemos uso de una subrutina (macro) de apoyo para darle formato a cada campo del autoformato, solo estoy considerando las propiedades más representativas, por supuesto, puedes complementarla con todas las que necesites pues estos campos tienen más de cuarenta propiedades.

6.5.4 Formato de página

El formato de página siempre tiene que establecerse por medio de un estilo de página, este, en conjunto con otra serie de propiedades de la hoja donde estés trabajando, te dará control completo sobre el formato de página. La siguiente macro te muestra en un cuadro de mensaje, la lista de formatos de página existentes en el documento desde el cual se ejecuta.

```
Sub FormatoPagina1()
Dim oDoc As Object
Dim oEstilos As Object
Dim oEstilosPagina As Object

   'Accedemos al documento actual
   oDoc = ThisComponent
   'Accedemos a todos los estilos
   oEstilos = oDoc.getStyleFamilies()
   'Accedemos a los estilos de página
   oEstilosPagina = oEstilos.getByName("PageStyles")
   'Mostramos sus nombres
   MsgBox Join( oEstilosPagina.getElementNames(), Chr(13) )

End Sub
```

Verifica que efectivamente sean los estilos que tienes, ve a tu hoja y presiona la tecla F11 para abrir el cuadro de dialogo “*Estilo y Formato*”, selecciona los estilos de página, la lista debería ser igual a la que te devuelve la macro anterior.



Para saber que estilo de página tenemos en nuestra hoja activa, usamos:

```
Sub FormatoPagina2()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oEstilos As Object
Dim oEstilosPagina As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   'Verificamos el nombre del estilo actual
   MsgBox oHojaActiva.PageStyle

End Sub
```

Para aplicar el estilo de página que quieras, usamos la siguiente macro:

```
Sub FormatoPagina3()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oEstilos As Object
Dim oEstilosPagina As Object
Dim sEstilo As String

   sEstilo = "Report"
   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oEstilos = oDoc.getStyleFamilies()
   oEstilosPagina = oEstilos.getByName("PageStyles")
   'Verificamos que el nombre del estilo exista
   If oEstilosPagina.hasByName( sEstilo ) Then
         'Si existe lo aplicamos
         oHojaActiva.PageStyle( sEstilo )
   Else
         MsgBox "No se aplicó ningún estilo de página"
   End If

End Sub
```

Con el siguiente ejemplo, mostramos algunas propiedades del estilo de página predeterminado.

```
Sub FormatoPagina4()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oEstilos As Object
Dim oEstilosPagina As Object
```

```
Dim sEstilo As String
Dim oEP As Object

   sEstilo = "Default"
   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oEstilos = oDoc.getStyleFamilies()
   oEstilosPagina = oEstilos.getByName("PageStyles")
   If oEstilosPagina.hasByName( sEstilo ) Then
         oEP = oEstilosPagina.getByName( sEstilo )
         'Ancho y alto de la página
         MsgBox oEP.Width & " - " & oEP.Height
         'Margen superior e inferior
         MsgBox oEP.TopMargin & " - " & oEP.BottomMargin
         'Margen izquierdo y derecho
         MsgBox oEP.LeftMargin & " - " & oEP.RightMargin
   End If

End Sub
```

Para borrar un estilo, usamos el método *removeByName* de la siguiente manera.

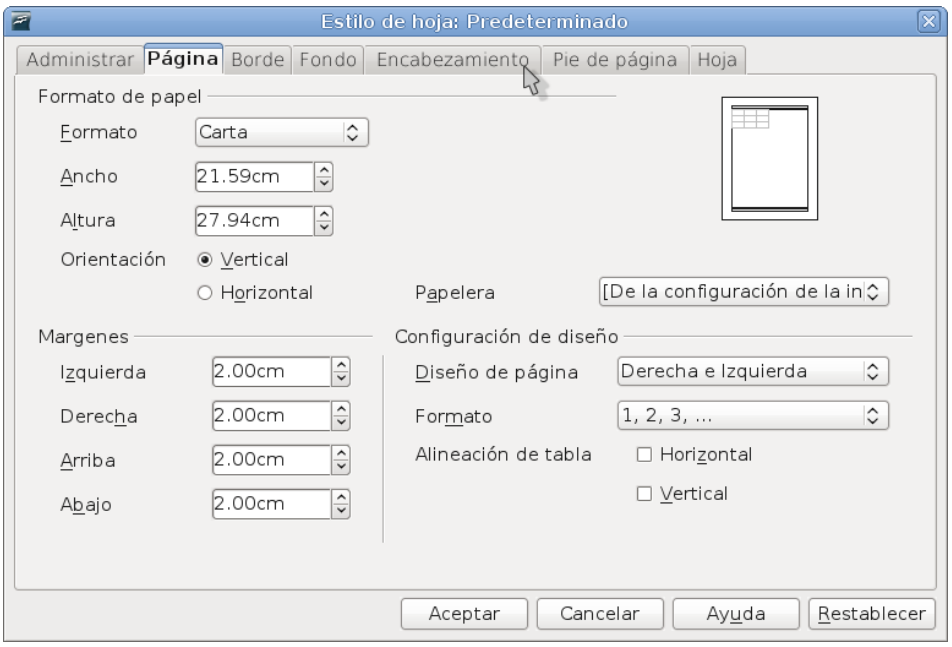
```
Sub FormatoPagina5()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oEstilos As Object
Dim oEstilosPagina As Object
Dim sEstilo As String

   sEstilo = "Videoteca"
   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oEstilos = oDoc.getStyleFamilies()
   oEstilosPagina = oEstilos.getByName("PageStyles")
   If oEstilosPagina.hasByName( sEstilo ) Then
         If MsgBox("¿Estas seguro de borrar el estilo de página?", 36) = 6 Then
               'Borramos el estilo
               oEstilosPagina.removeByName( sEstilo )
               MsgBox "Estilo borrado"
         Else
               MsgBox "El estilo no se borro"
         End If
   Else
         MsgBox "No existe el estilo de página"
   End If

End Sub
```

Ten cuidado con la macro anterior, dentro de la interfaz del usuario, los estilos Reporte (Report) y Predeterminado (Default) no se pueden borrar, pero por código si que se puede, si llegas a borrar “todos” los estilos, ya no podrás hace uso del menú *Formato* | *Página*, si no me crees y te gusta experimentar como a mi, borra todos los estilos para verificarlo, pero, aunque los borres, las hojas nuevas, al estar basadas en la plantilla predeterminada, tendrán de nuevo los estilos predeterminados, claro, si los borras de la plantilla, si que te quedarás sin estilos, no obstante, puedes volver a crearlos, como en el siguiente ejemplo.

```
Sub FormatoPagina6()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oEstilos As Object
Dim oEstilosPagina As Object
```

```
Dim sEstilo As String
Dim oEP As Object

   sEstilo = "Videoteca"
   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oEstilos = oDoc.getStyleFamilies()
   oEstilosPagina = oEstilos.getByName("PageStyles")
   If oEstilosPagina.hasByName( sEstilo ) Then
         oHojaActiva.PageStyle( sEstilo )
   Else
         'Creamos un nuevo estilo
         oEP = oDoc.createInstance( "com.sun.star.style.PageStyle" )
         'Lo agregamos a la colección de estilos
         oEstilosPagina.insertByName( sEstilo, oEP )
         'Le damos formato
         With oEp
               .Width = 27940               'Ancho
               .Height = 21590              'Alto
                     'Margen superior
               .BottomMargin = 2000         'Margen inferior
               .LeftMargin = 1000           'Margen izquierdo
               .RightMargin = 1000          'Margen derecho
               .IsLandscape = True          'Orientación de la página (horizontal)
               .CenterHorizontally = True   'Centrado horizontal del contenido
               .ScaleToPagesX = 1           'Una página de ancho
               .ScaleToPagesY = 0           'Las que salgan de alto
         End With
   End If

End Sub
```

Observa las características indicadas dentro de la misma macro, como dicen en mi pueblo -ni están todas las que son, ni son todas las que están-. Las características establecidas son las mismas del menú *Formato* | *Página...*, que, como podrás notar, son bastantes, veamos las más comunes.

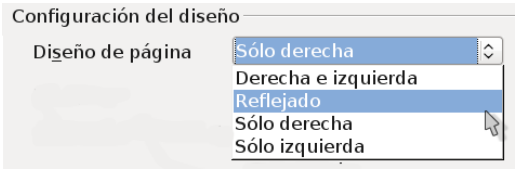


El ancho (Width) y alto (Height) de la página, se refiere al tamaño del papel, hay que establecerlos en centésimas de milímetro, es decir, cada 100 unidades es igual a 1 milímetro, estas propiedades, tienes que usarlas correctamente en combinación con la propiedad Orientación

(IsLandscape), es decir, si estableces esta propiedad en verdadero (True) es decir horizontal, entonces el ancho debe ser mayor que el alto, si la estableces en falso (False) es decir vertical, el alto debe ser mayor para que lo veas correctamente. En el siguiente ejemplo, la primer forma es correcta, la segunda no.

```
With oEP
      .Width = 27940              'Ancho
      .Height = 21590             'Alto
      .IsLandscape = True         'Orientación de la página (horizontal)
End With

With oEP
      .Width = 27940              'Ancho
      .Height = 21590             'Alto
      .IsLandscape = False        'Orientación de la página (vertical)
End With
```

Los margenes también están en centésimas de milímetros y se ven afectados por el diseño de página (PageStyleLayout), es decir, si estableces las páginas derecha e izquierda igual o si las reflejas, este valor esta determinado por la enumeración PageStyleLayout, cuyos posibles valores son los siguientes.

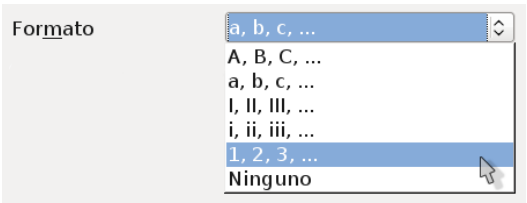


Propiedad PageStyleLayout	*Valor*	*Valor en Interfaz*
com.sun.star.style.PageStyleLayout.ALL	0	Derecha e izquierda
com.sun.star.style.PageStyleLayout.LEFT	1	Solo izquierda
com.sun.star.style.PageStyleLayout.RIGHT	2	Solo derecha
com.sun.star.style.PageStyleLayout.MIRRORED	3	Reflejado

Puedes establecer el valor o la constante como en el siguiente ejemplo.

```
oEP.PageStyleLayout = 0
oEP.PageStyleLayout = com.sun.star.style.PageStyleLayout.MIRRORED
```

El Formato, se refiere al estilo de numeración que se usará cuando se establece la numeración de páginas en el encabezado o en el pie de página, sus valores van del 0 al 5 y se corresponden al orden mostrado en la interfaz del usuario:

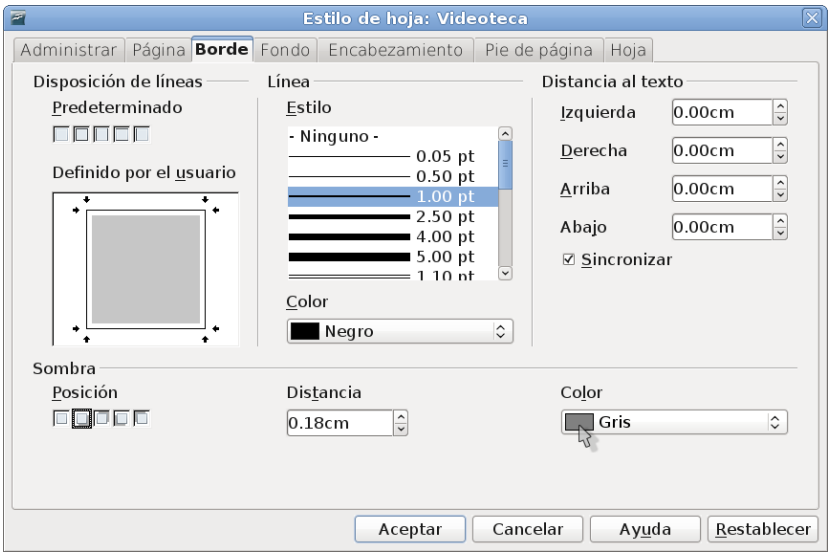



```
oEP.NumberingType = 4
```

En “Alineación de la tabla”, lo correcta seria, “centrado del contenido”, pues eso es lo que realmente hacen estas propiedades, para el sentido horizontal (CenterHorizontally) y vertical (CenterVertically), son valores booleanos.

```
        oEP.CenterHorizontally = True
        oEP.CenterVertically = False
```

En la ficha Borde, tenemos:



Para establecer el borde de la página, usamos la estructura *BorderLine*, ya vista en otros capítulos, en el siguiente ejemplo, establecemos el borde de cada lado de un color diferente.

```
Dim oBordeLinea As New com.sun.star.table.BorderLine

        With oBordeLinea
              .Color = RGB(200,0,0)
              .InnerLineWidth = 100  'Línea interior
              .OuterLineWidth = 50   'Línea exterior
              .LineDistance = 100    'Distancia entre líneas
        End With

        oEP.TopBorder = oBordeLinea
        oBordeLinea.Color = RGB(0,200,0)
        oEP.BottomBorder = oBordeLinea
        oBordeLinea.Color = RGB(0,0,200)
        oEP.LeftBorder = oBordeLinea
        oBordeLinea.Color = RGB(200,200,200)
        oEP.RightBorder = oBordeLinea
```

Si quieres ver solo una línea, establece solo una de las líneas, ya sea la interior (InnerLineWidth) o la exterior (OuterLineWidth) y no establezcas la distancia entre ellas (LineDistance), de nuevo, las unidades son centésimas de milímetros. La opción *Distancia al texto*, se refiere a la distancia entre los bordes y el contenido, si quieres la mista distancia para los cuatro bordes, como si estuviera activada la casilla de verificación *Sincronizar*, usa la propiedad *BorderDistance*, si quieres un distancia diferente para cada lado, usa las siguientes propiedades.

```
        'Para sincronizar todos los bordes a .5 mm
        oEP.BorderDistance = 500
```

```
'Para establecer cada borde diferente
oEP.TopBorderDistance = 500
oEP.BottomBorderDistance = 1000
oEP.LeftBorderDistance = 1500
oEP.RightBorderDistance = 2000
```

La sombra se establece con la propiedad ShadowFormat, que es una estructura com.sun.star.table.ShadowFormat, con las siguiente propiedades.

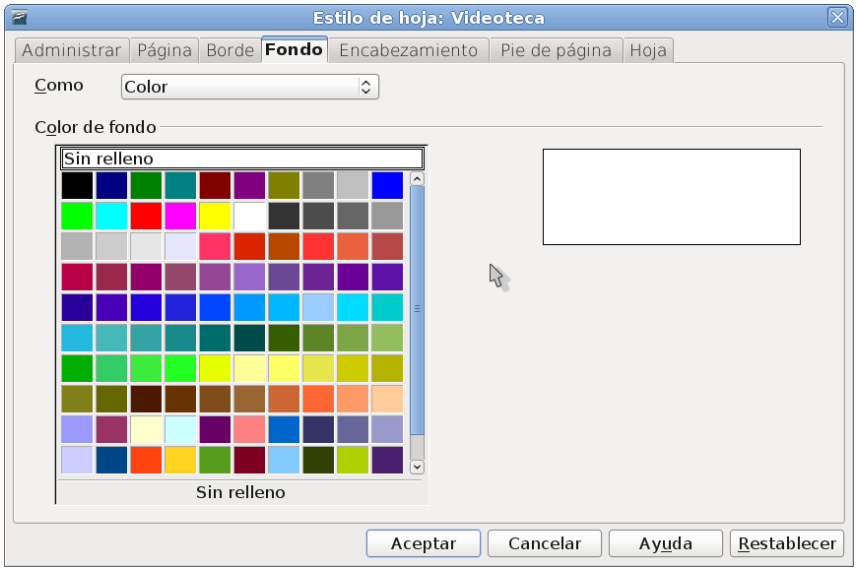
```
With oSombra
      .Location = 4                 'Donde esta la sombra
      .ShadowWidth = 1000           'Ancho de la sombra
      .IsTransparent = False        'Si es transparente
      .Color = RGB(180,180,180)     'Color
End With
oEP.ShadowFormat = oSombra
```

La propiedad Location, esta determinada por la siguiente enumeración.

Propiedad ShadowLocation	*Valor*	*Valor en Interfaz*
com.sun.star.table.ShadowLocation.NONE	0	Ninguna
com.sun.star.table.ShadowLocation.TOP_LEFT	1	Arriba a la izquierda
com.sun.star.table.ShadowLocation.TOP_RIGHT	2	Arriba a la derecha
com.sun.star.table.ShadowLocation.BOTTOM_LEFT	3	Abajo a la izquierda
com.sun.star.table.ShadowLocation.BOTTOM_RIGHT	4	Abajo a la derecha

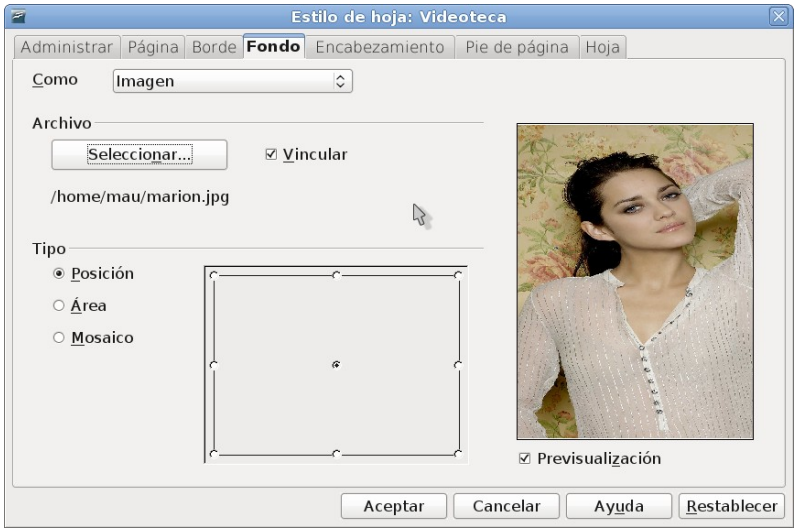
Si estableces la propiedad de transparencia (IsTransparent) en verdadero (True), la sombra no se vera.

La siguiente ficha se llama Fondo, veamos su propiedad que es muy sencilla.




```
oEP.BackColor = RGB(200,220,240)
```

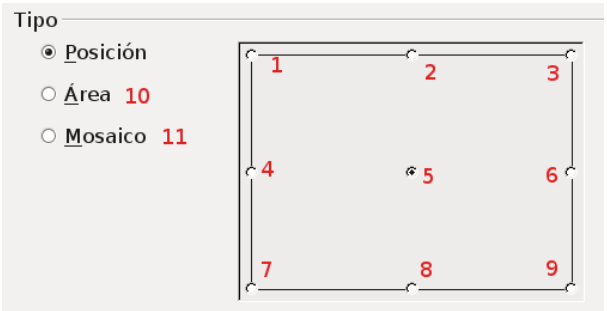
También podemos establecer una imagen de fondo, aquí puedes ver a mi novia Marion Cotillard.



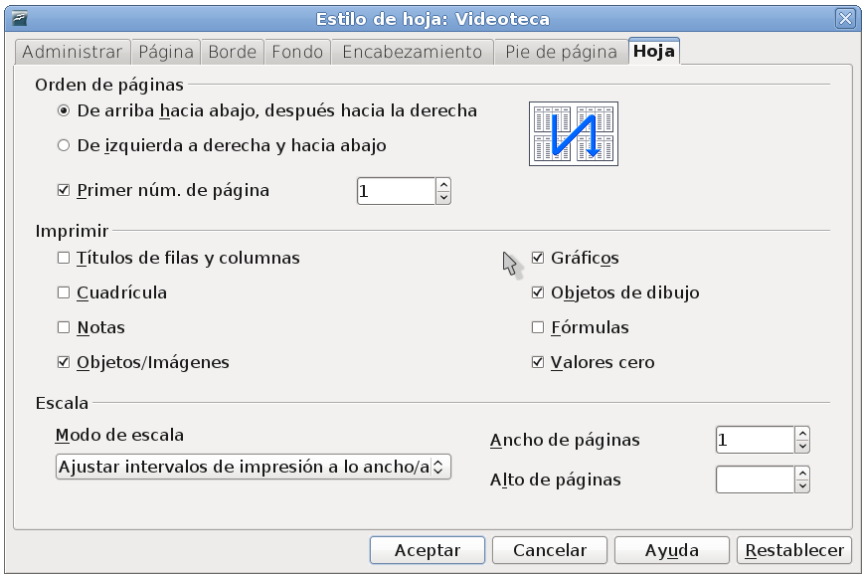
La ruta de la imagen la establecemos con *BackGraphicURL*, es importante que le pases la ruta en formato URL, y por supuesto que la imagen exista, si no, no te dará un error pero no te mostrará nada, cuida mucho el tamaño de las imágenes insertadas en documentos, si se usara solo en pantalla, que es lo usual, no requieres ni un gran tamaño ni una gran resolución.

```
'Establecemos la ruta de la imagen
oEP.BackGraphicURL = ConvertToURL( "/home/mau/marion.jpg" )
'Y la posición donde la queremos
oEP.BackGraphicLocation = 5
```

La posición (BackGraphicLocation) puede tomar valores de 1 a 11, de acuerdo a la siguiente imagen, en la opción Área, te cubrirá la hoja completa con la imagen.



Nos saltamos el encabezado y pie de página que por su complejidad veremos al final, y veamos la ultima ficha; Hoja, que tiene las siguientes propiedades.



El orden de impresión (Orden de página) lo establecemos con la propiedad; *PrintDownFirst*, valor booleano que equivale a.

- Verdadero (True) = De arriba hacia abajo, después hacia la derecha
- Falso (False) = De izquierda a derecha y hacia abajo

```
oEP.PrintDownFirst = True
```

El valor de *Primer núm, de página*, esta representado por la propiedad *FirstPageNumber*, valor tipo entero (integer) que establece un inicio de numeración de página diferente de uno. Ten cuidado si lo usas en conjunto con el total de hojas, pues te puede dar resultados algo extraños, como mostrarte Hoja 5 de 3.

```
oEP.FirstPageNumber = 8
```

Las siguientes propiedades solo determinan que opciones se imprimirán y cuales no, en cada línea te muestro su equivalencia en la interfaz del usuario.

```
    With oEP
          .PrintAnnotations = False    'Imprimir las notas de celdas
          .PrintCharts = True          'Imprimir los Gráficos
          .PrintDrawing = True         'Imprimir los Objetos de dibujo
          .PrintFormulas = False       'Imprimir las formulas, en vez del resultado
          .PrintGrid = False           'Imprimir la cuadrícula de las celdas
          .PrintHeaders = False        'Imprimir encabezados de filas y columnas
          .PrintObjects = True         'Imprimir Objetos e imágenes
          .PrintZeroValues = True      'Imprimir los valores cero
    End With
```

La escala de impresión nos puede resolver muy fácilmente problemas de impresión, la primera opción es aumentar o disminuirla según nuestras necesidades con la propiedad *PageScale*, un valor tipo entero (integer) entre 10 y 400, si lo estableces fuera de estos limites, se ajustará al valor mínimo o máximo más cercano, toma en cuenta que este porcentaje de cambio, solo es visible en la vista preliminar y a la hora de imprimir, en tu interfaz de usuario no veras ningún cambio de tamaño.

```
        oEP.PageScale = 80     'Ajustamos al 80% de escala
```

La segunda opción es ajustar a un determinado número de páginas, tienes que considerar un ajuste óptimo entre cantidad de datos y este valor, pues si tienes muchos datos y estableces un valor de páginas muy pequeño, el factor de escala se reducirá tanto que los datos no serán legibles.

```
        'Ajustamos a tres páginas de impresión
        oEP.ScaleToPages = 3
```

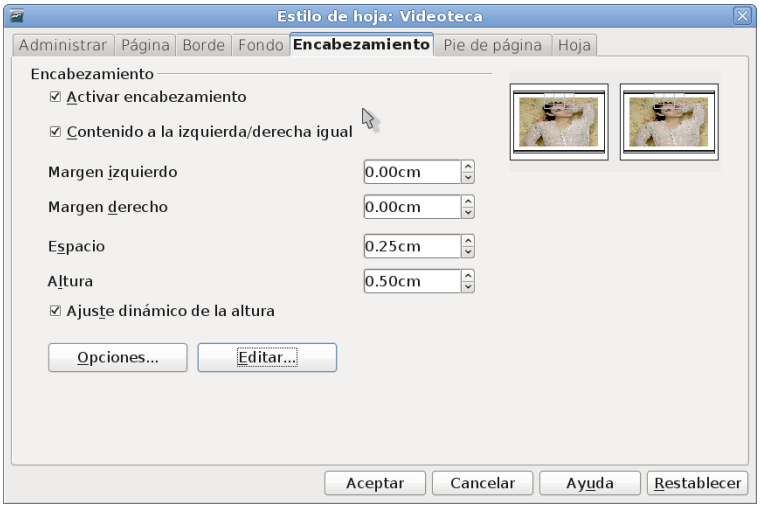
Por ultimo la que más me gusta, puedes establecer un determinado número de páginas de ancho por otro de alto, por ejemplo:

```
        'Establecemos la impresión en tres de ancho por dos de alto
        oEP.ScaleToPagesX = 3
        oEP.ScaleToPagesY = 2
```

Estas propiedades son muy útiles en eso grandes listados de datos, muy comunes donde tienes muchas filas y un determinado número de columnas que caben en el ancho de una hoja, ya sea vertical u horizontal, si estableces el ancho en 1 y el alto en 0, estamos forzando a usar exactamente una de ancho por las “necesarias” de alto.

```
        'Establecemos la impresión en una de ancho por las que salgan de alto
        oEp.ScaleToPagesX = 1
        oEp.ScaleToPagesY = 0
```

Ahora si, ya casi para terminar, veamos como modificar el encabezado y el pie de página de nuestro estilo de página. Lo primero que hay que hacer, es activarlos o desactivarlos según nuestras necesidades:



Como son muy similares las opciones de encabezado y pie de página, te iré mostrando sus propiedades, una junto a otra. Para activarlos o desactivarlos usamos:

```
        'Activamos el encabezado
        oEp.HeaderOn = True
        'y desactivamos el pie de página
        oEp.FooterOn = False
```

La opción *Contenido a la izquierda/derecha igual*, nos permite, si esta desactivada, establecer las opciones de encabezado o pie de página, de forma diferente para las páginas pares e impares de nuestra impresión, esta opción, junto con la de *Diseño de página* vista anteriormente, nos permiten configurar nuestra impresión como si de un libro se tratará. Para activarla o desactivarla usamos.

```
        'Lo desactivamos en el encabezado
        oEP.HeaderIsShared = False
        'Lo activamos en el pie de página
        oEP.FooterIsShared = True
```

Después tenemos las opciones para establecer los margenes en centésimas de milímetros, este margen es distinto e independiente del margen de la página.

```
        'Establecemos los margenes del encabezado en 3 cm
        oEP.HeaderRightMargin = 3000
        oEP.HeaderLeftMargin = 3000

        'Establecemos los margenes del pie de página en 2 cm
        oEP.FooterRightMargin = 2000
        oEP.FooterLeftMargin = 2000
```

La propiedad *Espacio*, se refiere a la distancia entre en el encabezado o pie de página y el contenido del documento.

```
        'Ajustamos la distancia del encabezado al cuerpo del documento¡
        oEP.HeaderBodyDistance = 500
        'Ajustamos la distancia del pie de página al cuerpo del documento¡
        oEP.FooterBodyDistance = 1000
```

La propiedad *Altura*, es la altura del encabezado o pie de página, es decir, el espacio disponible para introducir información, pero toma en cuenta que dentro de este valor, esta considerada la distancia entre el encabezado o pie de página y el cuerpo del documento, por lo que si quieres una distancia absoluta, debes de sumarle dicho valor, como en.

```
        'Ajustamos la altura del encabezado sumando el espacio al cuerpo
        oEP.HeaderHeight = 2000 + oEP.HeaderBodyDistance
        'Y del pie de página
        oEP.FooterHeight = 1000 + oEP.FooterBodyDistance
```

La propiedad *Ajuste dinámico de la altura*, nos permite adaptar la altura al contenido del encabezado o pie de página, si esta activada, la medida introducida en la propiedad anterior no será tomada en cuenta, esta propiedad es muy útil si establecemos por código el contenido de los encabezados o pies de página. Para activarla o desactivarla, usamos.

```
        'Establecemos la altura dinámica para el encabezado
        oEP.HeaderDynamic = True
        'La desactivamos para el pie de página
        oEP.FooterDynamic = False
```

Igual que en el estilo de la página, en el encabezado como en el pie de página, tenemos las opciones de establecer, un borde, una sombra, un color de fondo o una imagen, la forma es la misma, lo que cambia, es las propiedades a las que se establecen estas.

```
        'Configuramos el tipo de borde
```

```
        With oBordeLinea
              .Color = RGB(0,150,0)
              .InnerLineWidth = 0    'Línea interior
              .OuterLineWidth = 80   'Línea exterior
        End With

        'Establecemos el mismo para todos
        oEP.HeaderTopBorder = oBordeLinea
        oEP.HeaderBottomBorder = oBordeLinea
        oEP.HeaderLeftBorder = oBordeLinea
        oEP.HeaderRightBorder = oBordeLinea

        'Cambiamos el color para el borde del pie de página
        With oBordeLinea
              .Color = RGB(150,150,0)
        End With

        'Establecemos el mismo para todos
        oEP.FooterTopBorder = oBordeLinea
        oEP.FooterBottomBorder = oBordeLinea
        oEP.FooterLeftBorder = oBordeLinea
        oEP.FooterRightBorder = oBordeLinea
```

Para la sombra aplica igual que en estilo de página.

```
        'Configuramos la sombra
        With oSombra
              .Location = 4                'Donde esta la sombra
              .ShadowWidth = 500           'Ancho de la sombra
              .Color = RGB(150,160,170)    'Color
        End With
        'Para el encabezado
        oEP.HeaderShadowFormat = oSombra
        'Para el pie de página
        oEP.FooterShadowFormat = oSombra
```

Y el color de fondo.

```
        'Establecemos el color de fondo del encabezado de forma aleatoria
        oEP.HeaderBackColor = RGB ( Rnd()*255,Rnd()*255,Rnd()*255 )
        'Igual para el pie de página
        oEP.FooterBackColor = RGB ( Rnd()*255,Rnd()*255,Rnd()*255 )
```

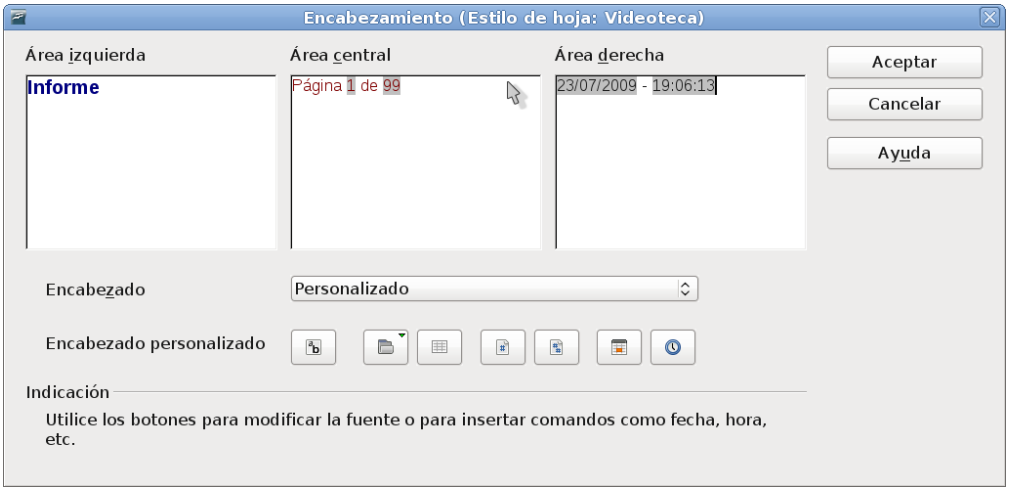
Para agregar una imagen, usamos.

```
        'Establecemos la ruta de la imagen
        oEP.HeaderBackGraphicURL = ConvertToURL( "/home/mau/arriba.jpg" )
        'Y la posición donde la queremos
        oEP.HeaderBackGraphicLocation = 10

        'Ahora en el pie de página
        oEP.FooterBackGraphicURL = ConvertToURL( "/home/mau/abajo.jpg" )
        'Y la posición donde la queremos
        oEP.FooterBackGraphicLocation = 10
```

Para quitar una imagen agregada, tienes que establecer esta propiedad en vacía.

```
        oEP.HeaderBackGraphicURL = ""
```

El contenido que puede llevar el encabezado o el pie de página, es muy rico, puedes establecer de forma independiente, el tipo de letra y su estilo de las tres áreas disponibles, como se ilustra en la imagen siguiente.



Veamos como controlar el contenido de estas secciones. Lo primero que tenemos que tener presente, es la opción vista anteriormente; *Contenido a la izquierda/derecha igual,* si esta propiedad esta seleccionada, entonces solo tenemos que modificar el contenido de un lado, pero si esta desactivada, entonces tenemos que establecer, tanto el encabezado o pie de página, tanto de las páginas pares como de las impares, estas propiedades son.

- RightPageHeaderContent Para el encabezado de las páginas derechas
- LeftPageHeaderContent Para el encabezado de las páginas izquierdas
- RightPageFooterContent Para el pie de página de las páginas derechas
- LeftPageFooterContent Para el pie de página de las páginas izquierdas

Para nuestro ejemplo, dado que son las mismas opciones para uno y otro, daremos por hecho que las propiedades (HeaderIsShared y FooterIsShared) están seleccionadas, de este modo, solo modificamos las propiedades de las páginas derechas, que serán iguales a las de la izquierda, pero no te confundas, aun manipulando solo las de las páginas derechas, tenemos acceso a las tres secciones mostradas en la imagen anterior, veamos como.

```
        'Mostramos el contenido de la izquierda del encabezado
        MsgBox oEP.RightPageHeaderContent.LeftText.String
        'Del centro
        MsgBox oEP.RightPageHeaderContent.CenterText.String
        'Y de la derecha
        MsgBox oEP.RightPageHeaderContent.RightText.String
```

Vamos a cambiar el texto de mostrado en el ejemplo anterior de las seis secciones accedibles, tres para el encabezado y tres para el pie de página, de la siguiente manera.

```
Sub FormatoPagina8()
Dim oDoc As Object
Dim oEstilos As Object
Dim oEstilosPagina As Object
Dim sEstilo As String
Dim oEP As Object
Dim oContenido As Object
Dim oTexto As Object

   sEstilo = "Videoteca"
   oDoc = ThisComponent
```

```
    oEstilos = oDoc.getStyleFamilies()
    oEstilosPagina = oEstilos.getByName("PageStyles")
    If oEstilosPagina.hasByName( sEstilo ) Then
          oEP = oEstilosPagina.getByName( sEstilo )
          oEP.HeaderOn = True
          oEP.FooterOn = True

          'Accedemos al contenido del encabezado
          oContenido = oEP.RightPageHeaderContent()
          'Accedemos a la parte izquierda
          oTexto = oContenido.LeftText()
          'Cambiamos el texto
          oTexto.String = "Esta es la izquierda" & Chr(13) & "en el encabezado"
          'Y el color
          oTexto.Text.CharColor = RGB(255,0,0)
          'Reasignamos el contenido para ver los cambios
          oEP.RightPageHeaderContent = oContenido

          'Lo mismo para el centro del encabezado
          oTexto = oContenido.CenterText()
          oTexto.String = "Este es el centro" & Chr(13) & "en el encabezado"
          oTexto.Text.CharColor = RGB(0,255,0)
          oEP.RightPageHeaderContent = oContenido

          'Y la derecha
          oTexto = oContenido.RightText()
          oTexto.String = "Esta es la derecha" & Chr(13) & "en el encabezado"
          oTexto.Text.CharColor = RGB(0,0,255)
          oEP.RightPageHeaderContent = oContenido

          'Ahora modificamos el pie de página
          oContenido = oEP.RightPageFooterContent()
          oTexto = oContenido.LeftText()
          oTexto.String = "Esta es la izquierda" & Chr(13) & "en el pie de página"
          oTexto.Text.CharColor = RGB(0,0,255)
          oEP.RightPageFooterContent = oContenido

          oTexto = oContenido.CenterText()
          oTexto.String = "Este es el centro" & Chr(13) & "en el pie de página"
          oTexto.Text.CharColor = RGB(0,255,0)
          oEP.RightPageFooterContent = oContenido

          oTexto = oContenido.RightText()
          oTexto.String = "Esta es la derecha" & Chr(13) & "en el pie de página"
          oTexto.Text.CharColor = RGB(255,0,0)
          oEP.RightPageFooterContent = oContenido
    End If

End Sub
```

Como sabes, hay una serie de campos predeterminados que podemos incluir en cualquier área de un encabezado o de un pie de página, campos como, la fecha, la hora, el número de página y el total de páginas, veamos como insertarlos.

```
Sub FormatoPagina9()
Dim oDoc As Object
Dim oEstilos As Object
Dim oEstilosPagina As Object
Dim sEstilo As String
Dim oEP As Object
Dim oContenido As Object
Dim oTexto As Object
Dim oCursor As Object
```

```
Dim oCampo As Object

   sEstilo = "Videoteca"
   oDoc = ThisComponent
   oEstilos = oDoc.getStyleFamilies()
   oEstilosPagina = oEstilos.getByName("PageStyles")
   If oEstilosPagina.hasByName( sEstilo ) Then
         oEP = oEstilosPagina.getByName( sEstilo )
         oEP.HeaderOn = True
         oEP.FooterOn = True

         'Accedemos al contenido del pie de página
         oContenido = oEP.RightPageFooterContent()
         'Creamos una instancia del campo Nombre del archivo
         oCampo = oDoc.createInstance ("com.sun.star.text.TextField.FileName")
         'Limpiamos las tres secciones
         oContenido.LeftText.String = ""
         oContenido.CenterText.String = ""
         oContenido.RightText.String = ""
         'Creamos un cursor en esa sección
         oCursor = oContenido.LeftText.createTextCursor()
         'Nos movemos al final
         oCursor.gotoEnd( False )
         'Insertamos el campo
         oContenido.LeftText.insertTextContent (oCursor, oCampo, True)

         'Creamos el campo Número de página
         oCampo = oDoc.createInstance ("com.sun.star.text.TextField.PageNumber")
         'Creamos un cursor en la sección central
         oCursor = oContenido.CenterText.createTextCursor()
         oCursor.gotoEnd( False )     'Nos movemos
         oCursor.String = "Página "   'Insertamos un texto
         oCursor.gotoEnd( False )     'Nos movemos
         'Insertamos el campo
         oContenido.CenterText.insertTextContent (oCursor, oCampo, True)
         oCursor.gotoEnd( False )     'Nos movemos
         oCursor.String = " de "           'Insertamos más texto
         oCursor.gotoEnd( False )     'Nos movemos
         'Creamos el campo, Total de páginas
         oCampo = oDoc.createInstance ("com.sun.star.text.TextField.PageCount")
         'Lo insertamos
         oContenido.CenterText.insertTextContent (oCursor, oCampo, True)

         'Creamos el campo de Fecha
         oCampo = oDoc.createInstance ("com.sun.star.text.TextField.DateTime")
         oCampo.IsDate = True         'Es una fecha
         'Creamos un cursor en la sección derecha
         oCursor = oContenido.RightText.createTextCursor()
         oCursor.gotoEnd( False )     'Nos movemos
         'Insertamos el campo
         oContenido.RightText.insertTextContent (oCursor, oCampo, True)

         'Actualizamos el contenido del pie de página
         oEP.RightPageFooterContent = oContenido

   End If

End Sub
```

6.5.5 Formato condicional

El formato condicional nos permite, de acuerdo al resultado de una expresión o formula, formatear una celda con diferentes estilos, esto es importante, solo puedes aplicar estilos de celdas existentes, por lo que asegurate de tener algunos de prueba. Puedes establecer de una a tres condiciones, que se evaluarán una a una, veamos una primer condición sencilla.

```
Sub FormatoCondicional1()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As Object
Dim oFC As Object
Dim mCondiciones(3) As New com.sun.star.beans.PropertyValue

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName( "A1" )

   'Accedemos al formato condicional
   oFC = oRango.getPropertyValue("ConditionalFormat")
   'Limpiamos cualquier formato existente, si no lo haces, las condiciones
   'se agregarán como segunda o tercer condición según corresponda
   oFC.clear()
   'Establecemos las condiciones del formato
   mCondiciones(0).Name = "Operator"
   'El operador =
   mCondiciones(0).Value = com.sun.star.sheet.ConditionOperator.EQUAL
   mCondiciones(1).Name = "Formula1"
   'El valor de la formula 1, es decir, si la celda = 100
   mCondiciones(1).Value = "100"
   mCondiciones(2).Name = "StyleName"
   'En caso de que se cumpla la condiciones, aplicamos el estilo Resaltado1,
   'recuerda que debes de crearlo previamente
   mCondiciones(2).Value = "Resaltado1"
   'Agregamos las condiciones al formato
   oFC.addNew ( mCondiciones() )
   'Reestablecemos la propiedad para que surtan efecto los cambios
   oRango.setPropertyValue( "ConditionalFormat", oFC )

End Sub
```

Acabamos de agregar, a la celda A1, la condición de que, si el valor de la celda es igual a 100, le establezca el estilo de celda Resaltado1, si el estilo no existe, no te dará ningún error, simplemente no aplicará ningún formato, verifiquemos que la agrego correctamente.

Ahora intentemos, agregar dos condiciones en vez de una, por ejemplo, a la celda C1, si es igual a 50 un estilo y si es mayor a 50 otro estilo, que quedaría así.

```
Sub FormatoCondicional2()
```

```
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As Object
Dim oFC As Object
Dim mCondiciones(3) As New com.sun.star.beans.PropertyValue

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName( "C1" )

   'Accedemos al formato condicional
   oFC = oRango.getPropertyValue("ConditionalFormat")
   oFC.clear()

   mCondiciones(0).Name = "Operator"
   mCondiciones(0).Value = com.sun.star.sheet.ConditionOperator.EQUAL
   mCondiciones(1).Name = "Formula1"
   mCondiciones(1).Value = "50"
   mCondiciones(2).Name = "StyleName"
   mCondiciones(2).Value = "Roja"
   oFC.addNew ( mCondiciones() )

   mCondiciones(0).Name = "Operator"
   mCondiciones(0).Value = com.sun.star.sheet.ConditionOperator.GREATER
   mCondiciones(1).Name = "Formula1"
   mCondiciones(1).Value = "50"
   mCondiciones(2).Name = "StyleName"
   mCondiciones(2).Value = "Azul"
   oFC.addNew ( mCondiciones() )

   'Reestablecemos la propiedad para que surtan efecto los cambios
   oRango.setPropertyValue( "ConditionalFormat", oFC )

End Sub
```

Observa la matriz de propiedades mCondiciones, esta forma de establecer pares de propiedades con un nombre (Name) y un valor (Value), ya la hemos usado anteriormente por lo que no te debe ser desconocida, la propiedad operador (Operator), es una enumeración que puede tener los siguiente valores.

com.sun.star.sheet.ConditionOperator	*Valor*	*Valor en Interfaz*
com.sun.star.sheet.ConditionOperator.NONE	0	Ninguna
com.sun.star.sheet.ConditionOperator.EQUAL	1	Igual
com.sun.star.sheet.ConditionOperator.NOT_EQUAL	2	Distinta de
com.sun.star.sheet.ConditionOperator.GREATER	3	Mayor que
com.sun.star.sheet.ConditionOperator.GREATER_EQUAL	4	Mayor o igual
com.sun.star.sheet.ConditionOperator.LESS	5	Menor que
com.sun.star.sheet.ConditionOperator.LESS_EQUAL	6	Menor o igual
com.sun.star.sheet.ConditionOperator.BETWEEN	7	Entre
com.sun.star.sheet.ConditionOperator.NOT_BETWEEN	8	No entre
com.sun.star.sheet.ConditionOperator.FORMULA	9	Formula

Veamos como establecer las condiciones para evaluar entre un par de valores.

```
Sub FormatoCondicional3()
Dim oDoc As Object
Dim oHojaActiva As Object
```

```
Dim oRango As Object
Dim oFC As Object
Dim mCondiciones(3) As New com.sun.star.beans.PropertyValue

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName( "B1:B10" )

   'Accedemos al formato condicional
   oFC = oRango.getPropertyValue("ConditionalFormat")
   oFC.clear()

   mCondiciones(0).Name = "Operator"
   mCondiciones(0).Value = com.sun.star.sheet.ConditionOperator.BETWEEN
   mCondiciones(1).Name = "Formula1"
   mCondiciones(1).Value = "0"
   mCondiciones(2).Name = "Formula2"
   mCondiciones(2).Value = "5"
   mCondiciones(3).Name = "StyleName"
   mCondiciones(3).Value = "Reprobado"
   oFC.addNew ( mCondiciones() )

   mCondiciones(0).Name = "Operator"
   mCondiciones(0).Value = com.sun.star.sheet.ConditionOperator.BETWEEN
   mCondiciones(1).Name = "Formula1"
   mCondiciones(1).Value = "6"
   mCondiciones(2).Name = "Formula2"
   mCondiciones(2).Value = "8"
   mCondiciones(3).Name = "StyleName"
   mCondiciones(3).Value = "Suficiente"
   oFC.addNew ( mCondiciones() )

   mCondiciones(0).Name = "Operator"
   mCondiciones(0).Value = com.sun.star.sheet.ConditionOperator.BETWEEN
   mCondiciones(1).Name = "Formula1"
   mCondiciones(1).Value = "9"
   mCondiciones(2).Name = "Formula2"
   mCondiciones(2).Value = "10"
   mCondiciones(3).Name = "StyleName"
   mCondiciones(3).Value = "Excelente"
   oFC.addNew ( mCondiciones() )

   'Restablecemos la propiedad para que surtan efecto
   oRango.setPropertyValue( "ConditionalFormat", oFC )

End Sub
```

Observa que ahora hemos usado la propiedad Formula1 y Formula2 para dar el intervalo de valores que nos interesa evaluar, entre 0 y 5, entre 6 y 8 y por ultimo entre 9 y 10, también observa que ahora, en vez de aplicar el formato condicional a una sola celda, lo hemos hecho a un rango de celdas, en este caso, “B1:B10”.

Donde realmente me parece, se ve la nobleza del formato condicional, es con el uso de formulas, pues nos da un margen amplísimo para establecer las condiciones, por ejemplo, supongamos que tenemos un rango de celdas con fechas y queremos que los sábados y domingos se distingan de los demás días, crea dos estilos de celda nuevos, uno para los sábados y otro para los domingos, creamos su formato condicional con el siguiente ejemplo.

```
Sub FormatoCondicional4()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As Object
Dim oFC As Object
```

```
Dim mCondiciones(3) As New com.sun.star.beans.PropertyValue

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName( "I2:J32" )

   'Accedemos al formato condicional
   oFC = oRango.getPropertyValue("ConditionalFormat")
   oFC.clear()
   mCondiciones(0).Name = "Operator"
   mCondiciones(0).Value = com.sun.star.sheet.ConditionOperator.FORMULA
   mCondiciones(1).Name = "Formula1"
   mCondiciones(1).Value = "WEEKDAY(A1)=7"
   mCondiciones(2).Name = "StyleName"
   mCondiciones(2).Value = "Sabados"
   oFC.addNew ( mCondiciones() )

   mCondiciones(0).Name = "Operator"
   mCondiciones(0).Value = com.sun.star.sheet.ConditionOperator.FORMULA
   mCondiciones(1).Name = "Formula1"
   mCondiciones(1).Value = "WEEKDAY(A1)=1"
   mCondiciones(2).Name = "StyleName"
   mCondiciones(2).Value = "Domingos"
   oFC.addNew ( mCondiciones() )

   'Reestablecemos la propiedad para que surtan efecto
   oRango.setPropertyValue( "ConditionalFormat", oFC )

End Sub
```

Observa como hemos establecido el valor (Value) de la propiedad Formula1, estamos haciendo uso de una función incorporada DIASEM (WEEKDAY) de Calc, que nos devuelve el día de la semana que corresponda a la fecha pasada como argumento; Domingo=1, Lunes=2, etc, nota como, al ser una función de Calc, si esta lleva argumentos, forzosamente hay que proporcionárselos, observa que nuestro rango empieza en I2 y a la función le estamos pasando la celda A1, no te confundas, le decimos que es la celda A1 por que la referencia es “relativa”, el valor que queremos evaluar, es el de la misma celda I2, I3, etc, el formato condicional, ajustará correctamente las referencias a cada celda, por supuesto, compruebalo.

Vamos a crear un formato condicional un poco más elaborado, pero más divertido. Observa el par de listas en la imagen siguiente.

	A	B	C	D
1				
2		enero		enero
3		febrero		marzo
4		abril		abril
5		mayo		junio
6		julio		julio
7		agosto		septiembre
8		octubre		octubre
9		noviembre		diciembre
10				

La tarea, es hacer un formato condicional que nos resalte los meses que faltan en una y otra lista. Por supuesto, antes de ver la solución, trata de resolverlo por ti mismo, primero te muestro la imagen con el resultado, para que veas que es posible, después viene la macro con la solución que no tienes que ver hasta que intentes resolverlo, confío en ti. Observa como hemos satisfecho la condición, están resaltados con fondo gris, fuente azul y negritas, los meses que “no” están en cada lista, por supuesto, esto funciona con cualquier par de listas.

	A	B	C	D
1				
2		enero		enero
3		**febrero**		**marzo**
4		abril		abril
5		**mayo**		**junio**
6		julio		julio
7		**agosto**		**septiembre**
8		octubre		octubre
9		**noviembre**		**diciembre**
10				

Ahora la solución.

```
Sub FormatoCondicional5()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As Object
Dim oFC As Object
Dim mCondiciones(3) As New com.sun.star.beans.PropertyValue

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oRango = oHojaActiva.getCellRangeByName( "B2:B9" )
   oFC = oRango.getPropertyValue("ConditionalFormat")
   oFC.clear()

   'Condiciones para la primer lista
   mCondiciones(0).Name = "Operator"
   mCondiciones(0).Value = com.sun.star.sheet.ConditionOperator.FORMULA
   mCondiciones(1).Name = "Formula1"
   'Observa como ahora usamos referencia absolutas
   mCondiciones(1).Value = "COUNTIF($D$2:$D$9;A1)=0"
   mCondiciones(2).Name = "StyleName"
   mCondiciones(2).Value = "Faltante"
   oFC.addNew ( mCondiciones() )
   'Reestablecemos la propiedad para que surtan efecto
   oRango.setPropertyValue( "ConditionalFormat", oFC )

   'Para la segunda lista
   oRango = oHojaActiva.getCellRangeByName( "D2:D9" )
   oFC = oRango.getPropertyValue("ConditionalFormat")
   oFC.clear()
   mCondiciones(0).Name = "Operator"
   mCondiciones(0).Value = com.sun.star.sheet.ConditionOperator.FORMULA
   mCondiciones(1).Name = "Formula1"
   'Observa como ahora usamos referencia absolutas
   mCondiciones(1).Value = "COUNTIF($B$2:$B$9;A1)=0"
   mCondiciones(2).Name = "StyleName"
   mCondiciones(2).Value = "Faltante"
   oFC.addNew ( mCondiciones() )
   'Reestablecemos la propiedad para que surtan efecto
   oRango.setPropertyValue( "ConditionalFormat", oFC )

End Sub
```

Y con esto terminamos el tema de los formatos, que, te habrás dado cuenta, muchas veces consume mucho más código que otras necesidades.

¡¡Feliz programación!!

6.6 Imprimiendo

La impresión en OpenOffice.org es sumamente sencilla, pero debes de tomar en cuenta los elementos que están implicados en ella, por ejemplo: el área de impresión, la impresora disponible, el estilo de página establecido, entre otros elementos que veremos en este capítulo.

El método usado para imprimir es ***print*** y se usa de la siguiente manera.

```
Sub Imprimiendo1()
Dim mOpc()

   ThisComponent.Print( mOpc() )

End Sub
```

Observa que simple, pero !cuidado¡, la línea anterior podría enviarte a imprimir una gran cantidad de hojas, dependiendo de la configuración actual que tengas, usala con cuidado pero mejor aun, controla todas las opciones de impresión que aprenderemos aquí.

Para mostrar el nombre de la impresora activa, usamos.

```
Sub Imprimiendo2()
Dim mDI()

   'Mostramos el nombre de la impresora activa
   mDI = ThisComponent.getPrinter()
   MsgBox mDI(0).Value

End Sub
```

El nombre mostrado dependerá del nombre registrado de tu impresora en tu S.O., para cambiar la impresora a usar, usamos.

```
Sub Imprimiendo3()
'Matriz para el descriptor de impresión
Dim mDI(2) As New com.sun.star.beans.PropertyValue
Dim mOpc()

   'Cambiamos la impresora para imprimir
   mDI(0).Name = "Name"
   mDI(0).Value = "HP_PSC_2200"
   'Cambiamos la orientación 0 = Vertical, 1 = Horizontal
   mDI(1).Name = "PaperOrientation"
   mDI(1).Value = 1
   'Establecemos las opciones de la impresora
   ThisComponent.setPrinter( mDI )
   'Enviamos a imprimir
   ThisComponent.Print( mOpc() )

End Sub
```

Si la impresora no esta registrada con dicho nombre, el lenguaje no te dará ni mostrará un error, así que asegurate de usar el nombre correcto. Si bien podemos cambiar la

orientación del papel para imprimir desde este descriptor, mi recomendación es que solo uses el descriptor de impresión para cambiar de impresora, y todas las demás opciones las establezcas en un estilo de página donde tenemos más riqueza de opciones y parámetros.

En las opciones de impresión tienes algunas características para controlar la cantidad de copias, el orden y las páginas que deseamos imprimir.

```
Sub Imprimiendo4()
'Matriz para las opciones de impresión
Dim mOpc(2) As New com.sun.star.beans.PropertyValue
Dim mDI(2) As New com.sun.star.beans.PropertyValue

   mDI(0).Name = "Name"
   mDI(0).Value = "HP_LaserJet"
   mDI(1).Name = "PaperOrientation"
   mDI(1).Value = com.sun.star.view.PaperOrientation.PORTRAIT
   mDI(2).Name = "PaperFormat"
   mDI(2).Value = com.sun.star.view.PaperFormat.LETTER
   ThisComponent.setPrinter( mDI )

   'El número de copias
   mOpc(0).Name = "CopyCount"
   mOpc(0).Value = 1
   'Si se imprimen en juegos
   mOpc(1).Name = "Collate"
   mOpc(1).Value = True
   'Las páginas a imprimir
   mOpc(2).Name = "Pages"
   mOpc(2).Value = "1"

   'Enviamos a imprimir
   ThisComponent.Print( mOpc() )

End Sub
```

De estas opciones, la propiedad ***Pages***, tiene las siguientes variantes para establecer las hojas que deseamos imprimir del documento:

- Una sola página 1
- Varias páginas 3,8,12
- Un rango de páginas 5-10
- Combinación de las anteriores 4,7,10-15

Asegurate de pasarle el valor a esta propiedad como una cadena de texto, es decir, entre comillas y el valor de la propiedad CopyCount, como un valor entero, si no, te dará un error al querer establecer las opciones de impresión.

Observa que en descriptor de impresión, hemos establecido el formato del papel, para nuestro ejemplo en tamaño carta (LETTER), de nuevo, si no tienes problemas con tu impresión, el tamaño del papel es mejor establecerlo en el estilo de página, en mi experiencia, algunas impresoras “rebeldes”, se niegan a tomar el formato de papel establecido en el formato de página, por ello, se tiene que establecer en el descriptor de impresión, te sugiero hacer tus pruebas respectivas y si no es necesario, solo establece el tamaño en el estilo de página. Los valores que puede tomar esta propiedad son.

com.sun.star.view.PaperFormat	*Valor*	*Valor en Interfaz*
com.sun.star.view.PaperFormat.A3	0	A3
com.sun.star.view.PaperFormat.A4	1	A4
com.sun.star.view.PaperFormat.A5	2	A5

com.sun.star.view.PaperFormat	*Valor*	*Valor en Interfaz*
com.sun.star.view.PaperFormat.B5	3	B5
com.sun.star.view.PaperFormat.LETTER	4	Carta
com.sun.star.view.PaperFormat.LEGAL	5	Oficio
com.sun.star.view.PaperFormat.TABLOID	6	Doble carta
com.sun.star.view.PaperFormat.USER	7	Usuario

Nota que hasta ahora, hemos usado el objeto ThisComponent, es decir, el objeto desde donde se llama a la macro, el método ***Print*** esta presente desde cualquier archivo de Calc, por lo que puedes invocarlo desde cualquiera. En las hojas de calculo, tenemos la posibilidad de definir áreas de impresión, que no son más que rangos de celdas con un nombre, pero con algunas características que los hacen especiales, por ejemplo, la posibilidad de repetir una o varias filas o columnas en cada hoja de la impresión. Las opciones de las áreas de impresión que aprenderemos a usar y manipular, son las mismas presentes en el menú *Formato | Imprimir Rangos ->*. Para obtener las áreas de impresión actuales usamos.

```
Sub Imprimiendo5()
Dim mAI()
Dim sMensaje As String

   mAI = ThisComponent.getCurrentController.getActiveSheet.getPrintAreas()
   sMensaje = "La hoja activa tiene " & Format( UBound( mAI() ) + 1 ) & " áreas de impresión"
   MsgBox sMensaje

End Sub
```

Observa que estamos obteniendo (getPrintAreas) las áreas de impresión desde la hoja activa pues este método esta implementado en las hojas del archivo, este método te devolverá una matriz con todos las áreas de impresión establecidas en la hoja desde donde se invoque, cada elemento de esta matriz, tendrá la estructura de la dirección de un rango de celdas, visto múltiples veces a lo largo de este libro, definido por com.sun.star.table.CellRangeAddress.

En el siguiente ejemplo mostramos la dirección de cada área de impresión si las hay.

```
Sub Imprimiendo6()
Dim mAI()
Dim sMensaje As String
Dim co1 As Integer

   mAI = ThisComponent.getCurrentController.getActiveSheet.getPrintAreas()
   If UBound( mAI() ) > -1 Then
         For co1 = LBound(mAI) To UBound(mAI)
               sMensaje = DireccionRango( mAI(co1) )
               MsgBox sMensaje
         Next
   Else
         MsgBox "No hay área de impresión definidas en la hoja activa"
   End If

End Sub


Function DireccionRango( DirRango As Object) As String
Dim sTmp As String
Dim oRango As Object

   oRango = ThisComponent.getCurrentController.getActiveSheet.getCellRangeByPosition( _
                  DirRango.StartColumn, DirRango.StartRow, DirRango.EndColumn, DirRango.EndRow)
```

```
    sTmp = oRango.getSpreadsheet.getName() & "." & _
                oRango.getColumns().getByIndex(0).getName() & _
                oRango.getRangeAddress.StartRow + 1 & ":" & _
                oRango.getColumns().getByIndex(oRango.getColumns().getCount()-1).getName() & _
                oRango.getRangeAddress.EndRow + 1
    DireccionRango = sTmp

End Function
```

Si no tienes ningún área de impresión definida, Calc tomará todas las hojas con datos como disponibles para impresión, mi recomendación es que siempre establezcas tus áreas de impresión, esto te permite tener siempre el control de lo que se imprimirá, además de poder combinar áreas de diferentes hojas que al ser impresas quedan como si de un solo estilo se tratara. Para borrar las áreas de impresión de una hoja, usamos.

```
Sub Imprimiendo7()
Dim mAI()

   'Borramos todas las áreas de impresión de la hoja activa
   ThisComponent.getCurrentController.getActiveSheet.setPrintAreas( mAI() )

End Sub
```

Observa como simplemente pasándole una matriz vacía, todas las áreas de impresión serán borrados, nota que ahora estamos usando el método ***setPrintAreas***, si la hoja no tiene ningún área actual, este método no te dará error. Como supongo lo habrás deducido, para agregar áreas de impresión, solo tenemos que llenar dicha matriz con direcciones de rango validos como en el siguiente ejemplo.

```
Sub Imprimiendo8()
Dim mAI(1) As New com.sun.star.table.CellRangeAddress

   'Rango A1:E5
   mAI(0).Sheet = 0
   mAI(0).StartColumn = 0
   mAI(0).StartRow = 0
   mAI(0).EndColumn = 4
   mAI(0).EndRow = 4
   'Rango K11:P16
   mAI(1).Sheet = 1
   mAI(1).StartColumn = 10
   mAI(1).StartRow = 10
   mAI(1).EndColumn = 15
   mAI(1).EndRow = 15

   'Agregamos las áreas de impresión
   ThisComponent.getCurrentController.getActiveSheet.setPrintAreas( mAI() )

End Sub
```

Si, ya lo notaste, hice un poco de “trampita”, nota que en el primer rango, en la propiedad *Sheet*, puse 0 y en el segundo rango, en esta misma propiedad puse 1, esto no afecta el “destino” de las áreas de impresión, pues estas se agregarán a la hoja desde donde es invocado el método *setPrintAreas*, que, para nuestro ejemplo, es la hoja activa, no esta de más recordarte que los valores de inicio y fin, tanto de columna como de fila, deben estar dentro de rangos válidos y en el orden correcto. Esta forma de agregar áreas de impresión, te sustituye las existentes, pero como ya sabes manejar matrices, tu tarea es modificar la macro para agregar nuevas sin borrar las existentes.

Ahora, vamos a establecer las filas que queremos repetir en cada hoja de impresión.

```
Sub Imprimiendo9()
Dim oHojaActiva As Object
Dim mAI(0) As New com.sun.star.table.CellRangeAddress
Dim oFilR As New com.sun.star.table.CellRangeAddress

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   mAI(0).Sheet = 0
   mAI(0).StartColumn = 0
   mAI(0).StartRow = 0
   mAI(0).EndColumn = 3
   mAI(0).EndRow = 18
   oHojaActiva.setPrintAreas( mAI() )

   'Le decimos que queremos títulos de fila a repetir
   oHojaActiva.setPrintTitleRows(True)
   'Establecemos el rango de filas
   oFilR.StartRow = 2
   OFilR.EndRow = 3
   'Las establecemos
   oHojaActiva.setTitleRows( oFilR )

End Sub
```

Nota que solo hacemos uso de la fila de inicio y fin, las demás propiedades de la estructura *CellRangeAddress*, no serán tomadas en cuenta, asegurate de establecer un rango de filas valido. Cuando haces uso del método *setTitleRows*, en teoría, automáticamente, la propiedad *setPrintTitleRows*, se establece en verdadero (True), pero no esta de más que lo establezcas primero, también, si estableces *setPrintTitleRows*, en verdadero (True), pero no estableces un rango con *setTitleRows*, de forma predeterminada, se establecerá la fila 1 como titulo a repetir, del mismo modo, si estableces en *setTitleRows*, una estructura vacía, la fila 1 será establecida.

Para repetir las columnas es similar.

```
Sub Imprimiendo10()
Dim oHojaActiva As Object
Dim oColR As New com.sun.star.table.CellRangeAddress

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Le decimos que queremos titulos de columna a repetir
   oHojaActiva.setPrintTitleColumns(True)
   'Establecemos el rango de Columnas A:B
   oColR.StartColumn = 0
   OColR.EndColumn = 1
   'Las establecemos
   oHojaActiva.setTitleColumns( oColR )

End Sub
```

Estos métodos tienen las mismas consideraciones vistas para las filas, si vas a repetir filas y columnas, puedes hacer uso de una sola estructura *CellRangeAddress*, los métodos respectivos omitirán las propiedades no necesarias para ellos, como en el siguiente ejemplo.

```
Sub Imprimiendo11()
Dim oHojaActiva As Object
Dim oTitulosR As New com.sun.star.table.CellRangeAddress

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oTitulosR.StartColumn = 1
   oTitulosR.EndColumn = 1
   oTitulosR.StartRow = 2
```

```
    oTitulosR.EndRow = 2
    'Los establecemos
    oHojaActiva.setTitleColumns( oTitulosR )
    oHojaActiva.setTitleRows( oTitulosR )

End Sub
```

Si vas al menú *Insertar* | *Salto Manual ->*, veras que tienes dos opciones, Salto de fila y Salto de columna, tú puedes establecer la fila o columna que desees como un salto de página manual, es decir, estas forzando un salto de página, veamos como hacerlo.

```
Sub Imprimiendo12()
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Establecemos la fila 5 como salto de página
   oHojaActiva.getRows.getByIndex(4).IsStartOfNewPage = True
   'Verificamos que se haya establecido
   If oHojaActiva.getRows.getByIndex(4).IsManualPageBreak Then
        MsgBox "La fila es un salto de página"
   End If

End Sub
```

Nota como estamos accediendo a una fila (la 5) y la establecemos como inicio de una nueva página con la propiedad; *IsStartOfNewPage*, al establecer esta propiedad en verdadero (True), estamos insertando un salto de página manualmente, lo cual verificamos con la propiedad; *IsManualPageBreak*, por supuesto, para quitarlo, solo tienes que establecerla en falso (False).

```
Sub Imprimiendo13()
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Quitamos el salto de página
   oHojaActiva.getRows.getByIndex(4).IsStartOfNewPage = False
   'Lo verificamos
   If Not oHojaActiva.getRows.getByIndex(4).IsManualPageBreak Then
        MsgBox "La fila NO es un salto de página"
   End If

End Sub
```

Lo mismo para las columnas.

```
Sub Imprimiendo14()
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Establecemos un salto de página de columna
   oHojaActiva.getColumns.getByIndex(2).IsStartOfNewPage = True
   'Lo verificamos
   If oHojaActiva.getColumns.getByIndex(2).IsManualPageBreak Then
        MsgBox "La columna es un salto de página"
   End If

End Sub


Sub Imprimiendo15()
```

```
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Quitamos un salto de página de columna
   oHojaActiva.getColumns.getByIndex(2).IsStartOfNewPage = False
   'Lo verificamos
   If Not oHojaActiva.getColumns.getByIndex(2).IsManualPageBreak Then
         MsgBox "La columna NO es un salto de página"
   End If

End Sub
```

En las hojas de Calc, tenemos dos tipos de saltos de página, los automáticos y los manuales, los primeros se insertan automáticamente, de acuerdo; al formato de la página y su contenido, los segundos los establecemos nosotros, con el siguiente ejemplo, mostramos todos los saltos de página, tanto de fila como de columna que tenga la hoja activa e informamos si es manual o no.

```
Sub Imprimiendo16()
Dim oHojaActiva As Object
Dim mSP()
Dim co1 As Integer

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Obtenemos todos los saltos de página, por filas, de la hoja activa
   mSP = oHojaActiva.getRowPageBreaks()
   'Mostramos los salto de fila
   For co1 = LBound( mSP ) To UBound( mSP )
         MsgBox "El salto de página esta en la línea: " & mSP(co1).Position + 1 & Chr(13) & _
                "Es salto manual: " & mSP(co1).ManualBreak
   Next co1
   'Ahora los de columna
   mSP = oHojaActiva.getColumnPageBreaks()
   For co1 = LBound( mSP ) To UBound( mSP )
         MsgBox "El salto de página esta en la columna: " & mSP(co1).Position + 1 & Chr(13) & _
                "Es salto manual: " & mSP(co1).ManualBreak
   Next co1

End Sub
```

Cuidado, la macro anterior puede llegar a mostrarte decenas de saltos de página, esto puede pasar cuando tienes datos y borras todas las áreas de impresión, de forma predeterminada, se agregan saltos de página en toda la hoja, de nuevo, la recomendación es siempre establecer tus áreas de impresión de forma explicita.

Puedes quitar “todos”, los saltos de páginas, tanto de fila como de columna, con el siguiente método.

```
Sub Imprimiendo17()
Dim oHojaActiva As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Quitamos TODOS los saltos de páginas manuales
   oHojaActiva.removeAllManualPageBreaks()

End Sub
```

El siguiente ejemplo, te inserta un salto de página cada 2 filas en la selección actual, procura no tener una selección muy grande y sobre todo no mandar a imprimir que te podrían salir muchas hojas, ve el resultado en tu vista preliminar.

```
Sub Imprimiendo18()
Dim oSel As Object
Dim co1 As Integer

   oSel = ThisComponent.getCurrentSelection()
   'Insertamos un salto de página cada dos filas
   For co1 = 2 To oSel.getRows.getCount() - 1 Step 2
        oSel.getRows.getByIndex(co1).IsStartOfNewPage = True
   Next

End Sub
```

Por ultimo, para ver tu vista previa, usa el siguiente método, que no me gusta pero por ahora es la única forma que conozco.

```
Sub Imprimiendo19()
Dim oDocF As Object
Dim oDH As Object

   oDocF = ThisComponent.getCurrentController.Frame
   oDH = createUnoService("com.sun.star.frame.DispatchHelper")
   'Mostramos la vista previa
   oDH.executeDispatch(oDocF, ".uno:PrintPreview", "", 0, Array())

End Sub
```

En versiones anteriores de OOo, cuando enviabas a imprimir, de forma predeterminada, se enviaba todas las hojas del documento, ahora, solo se envían las hojas seleccionadas, claro, esto siempre lo puedes cambiar desde el cuadro de dialogo imprimir en el menú *Archivo* | *Imprimir*..., Para seleccionar varias hojas a imprimir, solo tienes que seleccionar un rango de cada una de ellas, puede ser solo una celda, y agregarlos a un contenedor de rangos que ya aprendimos a usar, y seleccionarlos, al seleccionar rangos de diferentes hojas, como consecuencia, se seleccionan las hojas que los contienen, después, podemos enviar a imprimir como en el siguiente ejemplo.

```
Sub Imprimiendo20()
Dim oDoc As Object
Dim oHojas As Object
Dim mOpc()
Dim oRangos As Object

   oDoc = ThisComponent
   oHojas = oDoc.getSheets()
   'Creamos el contender para los rangos
   oRangos = oDoc.createInstance("com.sun.star.sheet.SheetCellRanges")

   'Agregamos la primer celda de cada hoja, puede ser cualquier celda
   oRangos.addRangeAddress( oHojas.getByIndex(6).getCellRangeByName("A1").getRangeAddress() ,False )
   oRangos.addRangeAddress( oHojas.getByIndex(7).getCellRangeByName("A1").getRangeAddress() ,False )
   oRangos.addRangeAddress( oHojas.getByIndex(8).getCellRangeByName("A1").getRangeAddress() ,False )

   'Al seleccionar las celdas de diferentes hojas, estamos seleccionando dichas hojas
   oDoc.getCurrentController.select( oRangos )

   'Enviamos a imprimir
   oDoc.print( mOpc() )

End Sub
```

La recomendación general, es que siempre establezcas tus áreas de impresión correctamente, así como el estilo de página con la configuración deseada, con lo cual, la impresión se facilitará enormemente.

6.7 Rangos de datos

Los rangos de datos, son áreas rectangulares de rangos de celdas delimitados por al menos una fila y una columna en blanco a las cuales se les establece un nombre y tienen características especiales que facilitan su uso como si de una base de datos se tratara. No confundas estos nombres, con los que puedes definir en el cuadro de nombres de la barra de formulas, ya que son distintos. Las mayor parte de las opciones que estudiaremos en este capitulo, son las presentes en el menú *Datos*, de la interfaz del usuario. El uso de hojas de calculo con datos tratados como bases de datos, es, según mi experiencia, el uso más cotidiano dado a esta herramienta, y no es gratuito, las herramientas disponibles para trabajar con datos estructurados de esta manera, presentes en Calc, sin hacer uso de macros, es bastante amplio, poderoso y versátil, ahora, imagínate lo que se puede hacer, automatizando estos procesos. Si bien este libro no es de bases de datos explícitamente, te ayudará mucho recordar que cada columna de tu rango de datos, lo podemos llamar; campo, y cada fila de estos; registros. Así mismo, en la primer fila de estos, se establecen los “títulos de campo”, normalmente en un formato diferente del resto de los datos, aunque esta primer fila no es indispensable, es mucho mejor tenerla. También, es recomendable, no dejar ninguna fila en blanco, entre esta y los datos, las filas, es decir, los registros, es mejor si están completos, es decir, que todos sus campos contienen datos, de nuevo, esto no es indispensable pero una base de datos se hace para llenarse de datos. Tampoco es recomendable dejar filas completas en blanco. Trata de que tus datos sean “consistentes”, esto quiere decir que si una columna (campo) lo llamas Edad, efectivamente se capturen números, o fechas si la calculas. En la actualidad hay una amplia fuente de información para una buena construcción de tus bases de datos, de tarea, busca algo acerca del siguiente tema: “normalización de bases de datos”, veras que tema tan interesante es.

6.7.1 Definiendo rangos

En el siguiente ejemplo, definimos un rango de bases de datos, en la primer hoja del documento, en el rango A1:D24.

```
Sub RangoDeDatos1()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDir As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   'Accedemos al conjunto de los rangos de bases de datos
   oRangosBD = oDoc.DataBaseRanges()
   'El nombre del nuevo rango
```

```
    sNombre = "Direcciones"
    'La dirección del nuevo rango
    With oDir
          .Sheet = 0            'La hoja
          .StartColumn = 0      'La columna de inicio
          .EndColumn = 3        'La columna final
          .StartRow = 0         'La fila de inicio
          .EndRow = 23          'La fila final
    End With
    'Lo agregamos a la colección
    oRangosBD.addNewByName( sNombre, oDir )
    'Regresamos el rango recién agregado
    oRBD = oRangosBD.getByName( sNombre )
    With oRBD
          .MoveCells = True           'Para que se actualice al insertar o eliminar celdas
          .KeepFormats = True   'Para que mantenga los formatos
    End With

End Sub
```

El nombre del nuevo rango no debe de existir o te dará un error, en el siguiente ejemplo, solicitamos al usuario el nombre del rango de datos y tomamos la selección actual como dirección para el nuevo rango de datos, también, verificamos que no exista el nuevo nombre.

```
Sub RangoDeDatos2()
Dim oDoc As Object
Dim oSel As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDir As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   'Nos aseguramos de que sea un rango de celdas
   If oSel.getImplementationNAme = "ScCellRangeObj" Then
         'Solicitamos el nuevo nombre
         sNombre = Trim( InputBox("Escribe el nombre del nuevo rango") )
         'Si no esta vacío
         If sNombre <> "" Then
               oRangosBD = oDoc.DataBaseRanges()
               'Verificamos que no exista el nombre
               If Not oRangosBD.hasByName( sNombre ) Then
                     'Y lo agregamos. Observa como tomamos la dirección de la selección
                     oRangosBD.addNewByName( sNombre, oSel.getRangeAddress() )
                     oRBD = oRangosBD.getByName( sNombre )
                     With oRBD
                           .MoveCells = True
                           .KeepFormats = True
                     End With
               Else
                     MsgBox "Ya existe el nombre del rango"
               End If
         Else
               MsgBox "El nombre no puede estar vacío"
         End If
   Else
         MsgBox "No es un rango de celdas"
   End If

End Sub
```

Para borrar un rango de datos usamos el método *removeByName*, toma en cuenta que lo único que se borra es el nombre del rango de datos y sus propiedades, las celdas y sus valores se mantienen.

```
Sub RangoDeDatos3()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   'El nombre del rango a borrar
   sNombre = "Pruebas"
   'Si el nombre no existe obtendrás un error
   If oRangosBD.hasByName( sNombre ) Then
         'Lo removemos de la colección
         oRangosBD.removeByName( sNombre )
         MsgBox "Rango de datos borrado"
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

Podemos cambiar las propiedades de un rango de datos existente, por ejemplo, cambiarle el nombre.

```
Sub RangoDeDatos4()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   'El nombre del rango a modificar
   sNombre = "Pruebas"
   If oRangosBD.hasByName( sNombre ) Then
         oRBD = oRangosBD.getByName( sNombre )
         'Le cambiamos el nombre
         oRBD.setName ("Nuevo nombre")
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

O cambiar su dirección.

```
Sub RangoDeDatos5()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDir As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   'El nombre del rango a modificar
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
```

```
        oRBD = oRangosBD.getByName( sNombre )
        'Cambiamos la dirección
        With oDir
              .Sheet = 0              'Cambiamos la hoja
              .StartColumn = 2        'La columna de inicio
              .EndColumn = 5          'La columna final
              .StartRow = 5           'La fila de inicio
              .EndRow = 19            'La fila final
        End With
        oRBD.setDataArea( oDir )
    Else
        MsgBox "El rango de datos no existe"
    End If

End Sub
```

Cambia la dirección de un rango con cuidado, puede suceder que acabe apuntando a un rango vacío de celdas sin querer.

6.7.2 Ordenar datos

Para ordenar datos, no necesitas forzosamente un rango de datos con nombre, veamos como ordenar un rango de celdas con y sin nombre de rango de datos. Tomaremos como datos de ejemplo, los siguientes.

Nº	Nombre	Año	Edad
1	nikole	1976	33
2	gloria	1976	33
3	antonio	1977	32
4	lidia	1967	42
5	paola	1979	30
6	vanessa	1974	35
7	paola	1972	37
8	paola	1968	41
9	paola	1968	41
10	lizette	1978	31
11	lizette	1978	31
12	lizette	1978	31
13	nikole	1977	32
14	gloria	1975	34
15	antonio	1979	30
16	lidia	1977	32
17	paola	1968	41
18	vanessa	1978	31
19	lizette	1969	40
20	nikole	1970	39
21	gloria	1971	38
22	antonio	1973	36

23	lidia	1968	41

Vamos a ordenar los datos por nombre.

```
Sub OrdenarDatos1()
Dim oHoja As Object
Dim oRango As Object
Dim mCamposOrden(0) As New com.sun.star.table.TableSortField
Dim mDescriptorOrden()

   'La hoja donde esta el rango a ordenar
   oHoja = ThisComponent.getSheets.getByName("agosto")
   'El rango a ordenar
   oRango = oHoja.getCellRangeByName("A1:D24")
   'Descriptor de ordenamiento, o sea, el "como"
   mDescriptorOrden = oRango.createSortDescriptor()

   'Los campos a orden, o sea, el "que"
   'Los campos empiezan en 0
   mCamposOrden(0).Field = 1
   'Orden ascendente
   mCamposOrden(0).IsAscending = True
   'Sensible a MAYUSCULAS/minusculas
   mCamposOrden(0).IsCaseSensitive = False
   'Tipo de campo AUTOMATICO
   mCamposOrden(0).FieldType = com.sun.star.table.TableSortFieldType.AUTOMATIC

   'Indicamos si el rango contiene títulos de campos
   mDescriptorOrden(1).Name = "ContainsHeader"
   mDescriptorOrden(1).Value = True
   'La matriz de campos a ordenar
   mDescriptorOrden(3).Name = "SortFields"
   mDescriptorOrden(3).Value = mCamposOrden

   'Ordenamos con los parámetros establecidos
   oRango.sort( mDescriptorOrden )

End Sub
```

Los puntos a los que debes poner atención son; el tipo de campo *FieldType*, puede tomar los siguientes valores.

com.sun.star.table.TableSortFieldType	*Valor*	*Valor en Interfaz*
com.sun.star.table.TableSortFieldType.AUTOMATIC	0	Automático
com.sun.star.table.TableSortFieldType.NUMERIC	1	Numérico
com.sun.star.table.TableSortFieldType.ALPHANUMERIC	2	Alfanumérico

En mis pruebas, establecer en uno y en otro, no me ha dado muchas variantes en velocidad, esto es, supongo, por que he hecho pruebas con pocos datos, habría que hacer pruebas de rendimiento con grandes datos para ver su desempeño pues no creo que esta propiedad este de adorno.

En el descriptor de orden, si estableces la propiedad; *ContainsHeader*, en falso (False), y tu rango de datos efectivamente tiene títulos de campo, estos no serán tomados en cuenta y se ordenaran en relación con el resto de tus datos, mi recomendación es que siempre establezcas esta propiedad en verdadero (True) y efectivamente te asegures de tenerlos, claro, a menos de que no te sea indispensable o de plano no los necesites.

Si quieres ordenar por más de un campo (por ahora el limite es tres), no hay más que agregar el segundo criterio a la matriz de campos, como en el ejemplo siguiente que ordenamos por nombre ascendente y después por edad descendente.

```
Sub OrdenarDatos2()
Dim oHoja As Object
Dim oRango As Object
Dim mCamposOrden(1) As New com.sun.star.table.TableSortField
Dim mDescriptorOrden()

   oHoja = ThisComponent.getSheets.getByName("agosto")
   oRango = oHoja.getCellRangeByName("A1:D24")
   mDescriptorOrden = oRango.createSortDescriptor()

   mCamposOrden(0).Field = 1
   mCamposOrden(0).IsAscending = True
   mCamposOrden(0).IsCaseSensitive = False
   mCamposOrden(0).FieldType = com.sun.star.table.TableSortFieldType.AUTOMATIC
   'Agregamos un segundo campo
   mCamposOrden(1).Field = 3
   'Este es descendente
   mCamposOrden(1).IsAscending = False
   mCamposOrden(1).IsCaseSensitive = False
   mCamposOrden(1).FieldType = com.sun.star.table.TableSortFieldType.AUTOMATIC

   mDescriptorOrden(1).Name = "ContainsHeader"
   mDescriptorOrden(1).Value = True
   mDescriptorOrden(3).Name = "SortFields"
   mDescriptorOrden(3).Value = mCamposOrden

   'Ordenamos con los parámetros establecidos
   oRango.sort( mDescriptorOrden )

End Sub
```

Una opción muy interesante, es la posibilidad de enviar el resultado a un destino diferente como en el siguiente ejemplo.

```
Sub OrdenarDatos3()
Dim oHoja As Object
Dim oRango As Object
Dim mCamposOrden(0) As New com.sun.star.table.TableSortField
Dim oDestino As Object
Dim mDescriptorOrden()

   oHoja = ThisComponent.getSheets.getByName("agosto")
   oRango = oHoja.getCellRangeByName("A1:D24")
   oDestino = oHoja.getCellRangeByName("G1")
   mDescriptorOrden = oRango.createSortDescriptor()

   mCamposOrden(0).Field = 1
   mCamposOrden(0).IsAscending = True
   mCamposOrden(0).IsCaseSensitive = False
   mCamposOrden(0).FieldType = com.sun.star.table.TableSortFieldType.AUTOMATIC

   mDescriptorOrden(1).Name = "ContainsHeader"
   mDescriptorOrden(1).Value = True
   mDescriptorOrden(3).Name = "SortFields"
   mDescriptorOrden(3).Value = mCamposOrden
   'Establecemos que queremos copiar el resultado a otro lado
   mDescriptorOrden(5).Name = "CopyOutputData"
   mDescriptorOrden(5).Value = True
   'Establecemos el destino de la copia
```

```
    mDescriptorOrden(6).Name = "OutputPosition"
    mDescriptorOrden(6).Value = oDestino.getCellAddress()

    'Ordenamos con los parámetros establecidos
    oRango.sort( mDescriptorOrden )

End Sub
```

Observa como establecemos el destino con una estructura *getCellAddress*, esta solo incluye la hoja destino, la columna y la fila de inicio. Si en el rango destino existen datos, estos serán totalmente reemplazados sin preguntarte nada.

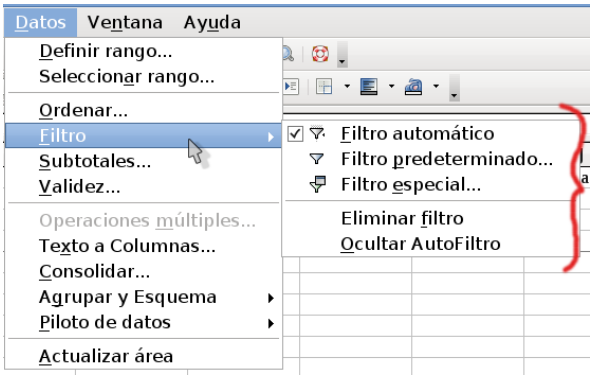
Ahora, ordenamos, pero accediendo desde un rango de datos.

```
Sub OrdenarDatos4()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oRango As Object
Dim mCamposOrden(0) As New com.sun.star.table.TableSortField
Dim mDescriptorOrden()

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   'Verificamos que exista el nombre del rango de datos
   If oRangosBD.hasByName( sNombre ) Then
         'Referencia al rango
         oRBD = oRangosBD.getByName( sNombre )
         'Referencia al origen del rango
         oRango = oRBD.getReferredCells()
         mDescriptorOrden = oRango.createSortDescriptor()

         mCamposOrden(0).Field = 1
         mCamposOrden(0).IsAscending = True
         mCamposOrden(0).IsCaseSensitive = False
         mCamposOrden(0).FieldType = com.sun.star.table.TableSortFieldType.AUTOMATIC

         mDescriptorOrden(1).Name = "ContainsHeader"
         mDescriptorOrden(1).Value = True
         mDescriptorOrden(3).Name = "SortFields"
         mDescriptorOrden(3).Value = mCamposOrden

         oRango.sort( mDescriptorOrden )
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

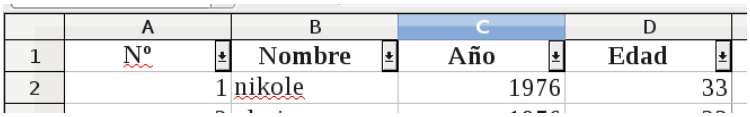
La propiedad importante es; *getReferredCells*, que te da acceso al rango origen de los datos, esta propiedad te devuelve un objeto *ScCellRangeObj*, por lo que puedes tener acceso a todas las propiedades de manipulación y formato vistas de este objeto, por ejemplo, puedes aplicarle un autoformato de tabla a los datos.

6.7.3 Filtrar datos

Las opciones que estudiaremos en este capitulo, son las presentes en el menú *Datos | Filtro ->* y seguiremos usando los datos del tema anterior.



Doy por hecho que no tienes problemas, como usuario, con cada una de estas opciones, veamos como establecerlas por código. Primero, la más sencilla, el filtro automático, te agrega un control de lista desplegable en el encabezado de campo de tus datos, como en.



	A	B	C	D
1	Nº	Nombre	Año	Edad
2	1	nikole	1976	33

Para hacer lo mismo por código, usamos.

```
Sub FiltrarDatos1()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
         'Referencia al rango
         oRBD = oRangosBD.getByName( sNombre )
         'Mostramos el autofiltro
         oRBD.AutoFilter = True
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

¿Y para quitarlos?, muy bien deducido, solo la establecemos en falso (False)

```
Sub FiltrarDatos2()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
```

```
    sNombre = "Direcciones"
    If oRangosBD.hasByName( sNombre ) Then
          'Referencia al rango
          oRBD = oRangosBD.getByName( sNombre )
          'Mostramos el autofiltro
          oRBD.AutoFilter = False
    Else
          MsgBox "El rango de datos no existe"
    End If

End Sub
```

Pero cuidado, la macro anterior, solo te quitará las flechas para desplegar el filtro del campo, si tienes establecido un filtro automático, este permanecerá, para eliminar completamente un filtro, tienes que hacer dos cosas; primero, eliminar el filtro y después mostrar las filas ocultas como te muestro en el siguiente ejemplo.

```
Sub FiltrarDatos3()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDesFiltro As Object
Dim mCamposFiltro() As New com.sun.star.sheet.TableFilterField

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
         oRBD = oRangosBD.getByName( sNombre )
         'Obtenemos la descripción del filtro
         oDesFiltro = oRBD.getFilterDescriptor()
         'Le pasamos una matriz vacía con una estructura de campo de filtro
         oDesFiltro.FilterFields = mCamposFiltro
         'Quitamos las flechas
         oRBD.AutoFilter = False
         'Mostramos las filas ocultas
         oRBD.getReferredCells.getRows.IsVisible = True
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

Entonces, cuando filtramos por código, no es indispensable mostrar las flechas de los campos del filtro, podemos filtrar directamente como en el siguiente ejemplo.

```
Sub FiltrarDatos4()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDesFiltro As Object
Dim mCamposFiltro(0) As New com.sun.star.sheet.TableFilterField

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
         oRBD = oRangosBD.getByName( sNombre )
         oDesFiltro = oRBD.getFilterDescriptor()
         'El campo por el que queremos filtrar
```

```
        mCamposFiltro(0).Field = 1
        'El tipo de comparación
        mCamposFiltro(0).Operator = com.sun.star.sheet.FilterOperator.EQUAL
        'Si es un número
        mCamposFiltro(0).IsNumeric = False
        'El valor de comparación
        mCamposFiltro(0).StringValue = "lizette"
        'Le pasamos los campos
        oDesFiltro.FilterFields = mCamposFiltro
        'Refrescamos el rango para ver el resultado del filtro
        oRBD.refresh()
   Else
        MsgBox "El rango de datos no existe"
   End If

End Sub
```

Que comprobamos que lo hace correctamente:

Nº	Nombre	Año	Edad
10	lizette	1978	31
11	lizette	1978	31
12	lizette	1978	31
19	lizette	1969	40

Ahora, veamos como filtrar un campo con dos condiciones.

```
Sub FiltrarDatos5()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDesFiltro As Object
Dim mCamposFiltro(1) As New com.sun.star.sheet.TableFilterField

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
        oRBD = oRangosBD.getByName( sNombre )
        oDesFiltro = oRBD.getFilterDescriptor()
        mCamposFiltro(0).Field = 1
        mCamposFiltro(0).Operator = com.sun.star.sheet.FilterOperator.EQUAL
        mCamposFiltro(0).IsNumeric = False
        mCamposFiltro(0).StringValue = "lizette"
        'Agregamos la segunda condición al mismo campo
        mCamposFiltro(1).Field = 1
        'Establecemos la relación con la condición anterior
        mCamposFiltro(1).Connection = com.sun.star.sheet.FilterConnection.OR
        mCamposFiltro(1).Operator = com.sun.star.sheet.FilterOperator.EQUAL
        mCamposFiltro(1).IsNumeric = False
        mCamposFiltro(1).StringValue = "paola"

        oDesFiltro.FilterFields = mCamposFiltro
        oRBD.refresh()
   Else
        MsgBox "El rango de datos no existe"
   End If

End Sub
```

Observa muy bien la propiedad *Connection*, es muy importante establecer correctamente esta, para obtener el resultado deseado, solo tiene dos posibilidades, una “O” (OR) de disyunción o una “Y” (AND) de conjunción, muchos errores de filtros incorrectos, son por la incorrecta aplicación de este sencillo parámetro, comprobamos que nuestro filtro esta correcto, y veamos más ejemplos.

Nº	Nombre	Año	Edad
5	paola	1979	30
7	lizette	1978	31
8	lizette	1978	31
9	lizette	1978	31
14	paola	1968	41
16	lizette	1969	40
21	paola	1972	37
22	paola	1968	41
23	paola	1968	41

Ahora filtraremos con condiciones en dos campos diferentes.

```
Sub FiltrarDatos6()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDesFiltro As Object
Dim mCamposFiltro(1) As New com.sun.star.sheet.TableFilterField

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
         oRBD = oRangosBD.getByName( sNombre )
         oDesFiltro = oRBD.getFilterDescriptor()
         mCamposFiltro(0).Field = 1
         mCamposFiltro(0).Operator = com.sun.star.sheet.FilterOperator.EQUAL
         mCamposFiltro(0).IsNumeric = False
         mCamposFiltro(0).StringValue = "lidia"
         'Agregamos la segunda condición a otro campo
         mCamposFiltro(1).Field = 3
         'Establecemos la relación con la condición anterior (Y)
         mCamposFiltro(1).Connection = com.sun.star.sheet.FilterConnection.AND
         mCamposFiltro(1).Operator = com.sun.star.sheet.FilterOperator.EQUAL
         'Ahora si buscamos por número
         mCamposFiltro(1).IsNumeric = True
         mCamposFiltro(1).NumericValue = 32

         oDesFiltro.FilterFields = mCamposFiltro
         oRBD.refresh()
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

Y una vez más lo comprobamos.

Nº	Nombre	Año	Edad
13	lidia	1977	32

En los siguientes ejemplos, ya no te mostraré el resultado, dando por entendido, que estas comprobando, como yo, que el resultado esperado es el correcto. En el siguiente ejemplo, filtramos a todos los que tengan entre 30 y 35 años.

```
Sub FiltrarDatos7()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDesFiltro As Object
Dim mCamposFiltro(1) As New com.sun.star.sheet.TableFilterField

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
         oRBD = oRangosBD.getByName( sNombre )
         oDesFiltro = oRBD.getFilterDescriptor()
         mCamposFiltro(0).Field = 3
         mCamposFiltro(0).Operator = com.sun.star.sheet.FilterOperator.GREATER_EQUAL
         mCamposFiltro(0).IsNumeric = True
         mCamposFiltro(0).NumericValue = 30
         'Agregamos la segunda condición a otro campo
         mCamposFiltro(1).Field = 3
         mCamposFiltro(1).Connection = com.sun.star.sheet.FilterConnection.AND
         mCamposFiltro(1).Operator = com.sun.star.sheet.FilterOperator.LESS_EQUAL
         mCamposFiltro(1).IsNumeric = True
         mCamposFiltro(1).NumericValue = 35

         oDesFiltro.FilterFields = mCamposFiltro
         oRBD.refresh()
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

Observa como hemos cambiado la propiedad “operador” (*Operator*) para satisfacer la condición, los posibles valores para esta propiedad, viene condicionados por la enumeración *com.sun.star.sheet.FilterOperator*, cuyos valores son.

com.sun.star.sheet.FilterOperator	*Valor*	*Valor en Interfaz*
com.sun.star.sheet.FilterOperator.EMPTY	0	Vacío
com.sun.star.sheet.FilterOperator.NOT_EMPTY	1	No vacío
com.sun.star.sheet.FilterOperator.EQUAL	2	Igual
com.sun.star.sheet.FilterOperator.NOT_EQUAL	3	No igual
com.sun.star.sheet.FilterOperator.GREATER	4	Mayor que
com.sun.star.sheet.FilterOperator.GREATER_EQUAL	5	Mayor o igual que
com.sun.star.sheet.FilterOperator.LESS	6	Menor que
com.sun.star.sheet.FilterOperator.LESS_EQUAL	7	Menor o igual que
com.sun.star.sheet.FilterOperator.TOP_VALUES	8	El mayor valor
com.sun.star.sheet.FilterOperator.TOP_PERCENT	9	El mayor porcentaje

com.sun.star.sheet.FilterOperator	*Valor*	*Valor en Interfaz*
com.sun.star.sheet.FilterOperator.BOTTOM_VALUES	10	El menor valor
com.sun.star.sheet.FilterOperator.BOTTOM_PERCENT	11	El menor porcentaje

En el siguiente ejemplo, seleccionamos los cinco registros con más edad.

```
Sub FiltrarDatos8()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDesFiltro As Object
Dim mCamposFiltro(0) As New com.sun.star.sheet.TableFilterField

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
         oRBD = oRangosBD.getByName( sNombre )
         oDesFiltro = oRBD.getFilterDescriptor()
         mCamposFiltro(0).Field = 3
         mCamposFiltro(0).Operator = com.sun.star.sheet.FilterOperator.TOP_VALUES
         mCamposFiltro(0).IsNumeric = True
         'Los cinco de más edad
         mCamposFiltro(0).NumericValue = 5

         oDesFiltro.FilterFields = mCamposFiltro
         oRBD.refresh()
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

Ahora, filtremos todos los nombres que comiencen por la letra “L”.

```
Sub FiltrarDatos9()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDesFiltro As Object
Dim mCamposFiltro(0) As New com.sun.star.sheet.TableFilterField

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
         oRBD = oRangosBD.getByName( sNombre )
         oDesFiltro = oRBD.getFilterDescriptor()
         mCamposFiltro(0).Field = 1
         mCamposFiltro(0).Operator = com.sun.star.sheet.FilterOperator.EQUAL
         mCamposFiltro(0).IsNumeric = False
         'Observa el parametro de la condición
         mCamposFiltro(0).StringValue = "l.*"
         'Establecemos que use expresiones regulares
         oDesFiltro.UseRegularExpressions = True
         oDesFiltro.FilterFields = mCamposFiltro
         oRBD.refresh()
   Else
         MsgBox "El rango de datos no existe"
   End If
```

```
End Sub
```

Observa como hemos establecido la propiedad para usar expresiones regulares (*UseRegularExpressions*) del descriptor del filtro para que tenga efecto nuestra condición. Las expresiones regulares son un concepto muy poderoso como podrás averiguarlo en tu buscador favorito.

Otra posibilidad bastante interesante de los filtros, es poder copiar el resultado en otra posición y dejar el origen intacto, veamos como.

```
Sub FiltrarDatos10()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
Dim oRBD As Object
Dim oDesFiltro As Object
Dim mCamposFiltro(0) As New com.sun.star.sheet.TableFilterField
Dim oDestino As Object

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
         oRBD = oRangosBD.getByName( sNombre )
         oDesFiltro = oRBD.getFilterDescriptor()
         mCamposFiltro(0).Field = 1
         mCamposFiltro(0).Operator = com.sun.star.sheet.FilterOperator.EQUAL
         mCamposFiltro(0).IsNumeric = False
         mCamposFiltro(0).StringValue = "gloria"
         'Le indicamos que queremos el resultado en otro lugar
         oDesFiltro.CopyOutputData = True
         'Y le indicamos donde, observa como obtenemos la dirección de la primer celda del rango
         oDestino = oRBD.ReferredCells().getCellByPosition(0,0).getCellAddress()
         'Después sumamos el ancho y alto del rango para dejar una columna y fila en blanco
         oDestino.Column = oDestino.Column + oRBD.ReferredCells.getColumns.getCount + 1
         oDestino.Row = oDestino.Row + oRBD.ReferredCells.getRows.getCount + 1
         'Establecemos el destino
         oDesFiltro.OutputPosition = oDestino
         oDesFiltro.FilterFields = mCamposFiltro
         oRBD.refresh()
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

En versiones anteriores a la 3.2, la propiedad para indicar que queremos el resultado en otra posición se llama: SaveOutputPosition, esta propiedad solo aplicaba para rangos de datos, a partir de la versión 3.2 se homologó para usar la misma propiedad tanto en rangos de datos como en rangos de celdas. Si las celdas destino no están vacías, estas, serán reemplazadas sin preguntarte nada, modifica la macro, para evaluar esto y se lo notifiques al usuario.

Otra característica muy poderosa de los filtros, es la posibilidad de filtrar los registros, omitiendo los duplicados, copia varios registros iguales para que notes la diferencia, se hace de la siguiente manera.

```
Sub FiltrarDatos11()
Dim oDoc As Object
Dim oRangosBD As Object
Dim sNombre As String
```

```
Dim oRBD As Object
Dim oDesFiltro As Object
Dim mCamposFiltro(0) As New com.sun.star.sheet.TableFilterField
Dim oDestino As Object

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   sNombre = "Direcciones"
   If oRangosBD.hasByName( sNombre ) Then
         oRBD = oRangosBD.getByName( sNombre )
         oDesFiltro = oRBD.getFilterDescriptor()
         mCamposFiltro(0).Field = 0
         'Seleccionamos los registros NO vacíos
         mCamposFiltro(0).Operator = com.sun.star.sheet.FilterOperator.NOT_EMPTY
         'Le indicamos que solo queremos registros únicos
         oDesFiltro.SkipDuplicates = True
         oDesFiltro.CopyOutputData = True
         oDestino = oRBD.ReferredCells.getCellByPosition(0,0).getCellAddress()
         oDestino.Column = oDestino.Column + oRBD.ReferredCells.getColumns.getCount + 1
         oDesFiltro.OutputPosition = oDestino
         oDesFiltro.FilterFields = mCamposFiltro
         oRBD.refresh()
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

Hasta ahora, hemos aplicados filtros, desde un rango de datos, pero los filtros no están limitados a estas áreas, al ser un método de rango de celdas (*ScCellRangeObj*), puedes aplicar un filtro a cualquier rango de celdas, la siguiente macro, toma el rango de celdas seleccionado y filtra los datos únicos dos columnas a la derecha, esta macro es muy útil para dejar listados únicos de lo que sea, verificalo.

```
Sub FiltrarDatos12()
Dim oDoc As Object
Dim oSel As Object
Dim oDesFiltro As Object
Dim mCamposFiltro(0) As New com.sun.star.sheet.TableFilterField
Dim oDestino As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   'Nos aseguramos de que sea un rango de celdas
   If oSel.getImplementationName = "ScCellRangeObj" Then
         'Creamos un nuevo descriptor de filtro vacio (True)
         oDesFiltro = oSel.createFilterDescriptor(True)

         'Establecemos los campos
         mCamposFiltro(0).Field = 0
         mCamposFiltro(0).Operator = com.sun.star.sheet.FilterOperator.NOT_EMPTY

         'Establecemos el destino
         oDestino = oSel.getCellByPosition(0,0).getCellAddress()
         oDestino.Column = oSel.getRangeAddress().EndColumn + 2

         'Establecemos las propiedades del filtro
         oDesFiltro.ContainsHeader = False
         oDesFiltro.SkipDuplicates = True
         oDesFiltro.CopyOutputData = True
         oDesFiltro.OutputPosition = oDestino
         oDesFiltro.FilterFields = mCamposFiltro
```

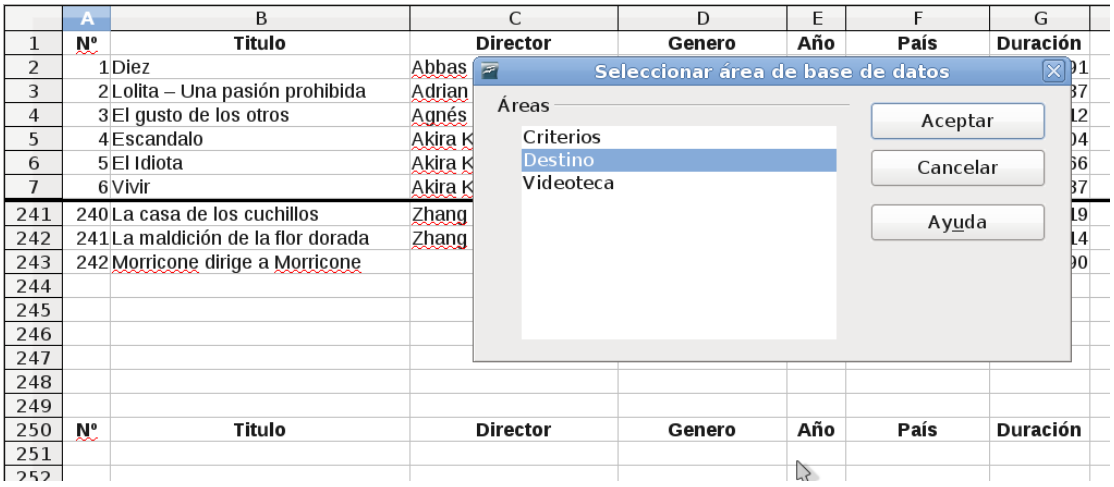
```
            'Y lo aplicamos
            oSel.filter( oDesFiltro )
    Else
            MsgBox "No es un rango de celdas"
    End If

End Sub
```

Hay diferencias importantes en comparación con los filtros en rango de datos; la primera es la forma de crear el descriptor de filtro, para un rango de celdas se usa el método *createFilterDescriptor*, al cual se le pasa el parámetro verdadero (True) si queremos que el nuevo descriptor este vacío y falso (False) si toma el existente en dicho rango de celdas, observa que también hemos establecido que este rango no tiene encabezados de campos (*ContainsHeader*), también cambia la propiedad para decirle que queremos copiar el resultado (*CopyOutputData*) en otra posición (homologada a partir de la versión 3.2, se usa la misma tanto en rangos de datos como en rangos de celda), y por ultimo, no refrescamos el rango, si no que aplicamos el filtro (*filter*), pasándole como parámetro el descriptor de filtro creado. Al igual que con los rangos de datos, si el destino contiene datos, serán reemplazados sin consultarte.

Veamos como usar las opciones del filtro especial, este tipo de filtro, nos permite establecer un rango como origen de los criterios para el filtro, lo que nos da la posibilidad de poder llegar a usar como criterios “todos” los campos de nuestra base de datos, dándonos absoluto control sobre los registros filtrados. Para demostrar el poder y versatilidad de esta opción, preparate una buena tabla de datos, con al menos cinco campos y varias decenas de registros, si lo prefieres, puedes usar la hoja llamada “Peliculas”, presente en los ejemplos que acompañan a estos apuntes y que será la que yo use.

El rango de nuestros datos es: “A1:G243”, establecemos un nombre a este rango; “Videoteca” si te parece, después, copia los títulos de los campos a la fila 250, selecciona el rango “A250:G255” y nombrala como “Criterios”, por ultimo, selecciona la celda “A260” y la bautizamos como “Destino”, si es de tu agrado, todo lo anterior puedes hacerlo por código que ya sabes hacerlo, ¿verdad?. Recuerda que esta definición de nombres es en el menú *Datos | Definir rango...*

Nuestra tabla debe verse más o menos así.



	A	B	C	D	E	F	G
1	Nº	Titulo	Director	Genero	Año	País	Duración
2	1	Diez	Abbas				91
3	2	Lolita – Una pasión prohibida	Adrian				37
4	3	El gusto de los otros	Agnés				12
5	4	Escandalo	Akira K				04
6	5	El Idiota	Akira K				66
7	6	Vivir	Akira K				37
241	240	La casa de los cuchillos	Zhang				19
242	241	La maldición de la flor dorada	Zhang				14
243	242	Morricone dirige a Morricone					90
244							
245							
246							
247							
248							
249							
250	Nº	Titulo	Director	Genero	Año	País	Duración
251							
252							

La definición de criterios para el filtro especial, se puede hacer como lo hemos venido aprendiendo, por lo que no lo repetiremos aquí, concentrándonos en estudiar la forma de establecer estos criterios desde un rango de datos con nombre. Te sugiero, con esta macro en especifico, asignarle una combinación de teclas o un botón en alguna barra de herramientas con la finalidad de que podamos ejecutarla varias veces desde la interfaz del usuario e ir viendo en vivo y en directo los resultados que nos arrogue. Aquí la macro.

```
Sub FiltrarEspecial1()
Dim oDoc As Object
Dim oRangosBD As Object
Dim oVideoteca As Object
Dim oDestino As Object
Dim oCriterios As Object
Dim oDesFiltro As Object


   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   'Nos aseguramos de que existen nuestros tres rangos
   If oRangosBD.hasByName( "Videoteca" ) And oRangosBD.hasByName( "Criterios" ) And
oRangosBD.hasByName( "Destino" ) Then
         'Obtenemos una referencia a los rangos origen
         oVideoteca = oRangosBD.getByName( "Videoteca" ).ReferredCells()
         oCriterios = oRangosBD.getByName( "Criterios" ).ReferredCells()
   oDestino = oRangosBD.getByName( "Destino" ).ReferredCells.getCellByPosition( 0,0
).getCellAddress()

         'Obtenemos el descriptor del filtro del rango de criterios a partir del rango de datos
         oDesFiltro = oCriterios.createFilterDescriptorByObject( oVideoteca )
         'Sin duplicados
         oDesFiltro.SkipDuplicates = True
         'Que pueda usar expresiones regulares
         oDesFiltro.UseRegularExpressions = True
         'Queremos el resultado en otra posición
         oDesFiltro.CopyOutputData= True
         'Le indicamos donde
         oDesFiltro.OutputPosition = oDestino
         'El rango contiene encabezados de campos
         oDesFiltro.ContainsHeader = True

         'Filtramos con las opciones seleccionadas
         oVideoteca.filter( oDesFiltro )
   Else
         MsgBox "El rango de datos no existe"
   End If

End Sub
```

Es importante notar que para el filtro especial usamos en nuevo método llamado *createFilterDescriptorByObject*, que se invoca desde el rango de criterios (*oCriterios*), pasándole como argumento, el rango de datos (*oVideoteca*), las restantes propiedades usadas en el ejemplo ya las hemos tratado.

Ahora, los criterios tienes que establecerlos en las celdas correspondientes, por ejemplo, para saber que películas tenemos de mi director favorito, usamos.

Nº	Titulo	Director	Genero	Año	País	Duración
		Akira Kurosawa				

Ejecuta la macro y vemos el resultado.

Nº	Titulo	Director	Genero	Año	País	Duración
4	Escandalo	Akira Kurosawa	Drama	1950	Japón	104
5	El Idiota	Akira Kurosawa	Drama	1951	Japón	166
6	Vivir	Akira Kurosawa	Drama	1952	Japón	137
7	Los siete samuráis	Akira Kurosawa	Drama / Acción	1954	Japón	200
8	La fortaleza escondida	Akira Kurosawa	Acción / Aventura	1958	Japón	123
9	Yojimbo el Mercenario	Akira Kurosawa	Drama	1961	Japón	110
10	Barbaroja	Akira Kurosawa	Drama	1965	Japón	180
11	Kagemusha – La Sombra del Gue	Akira Kurosawa	Drama	1980	Japón	159
12	Los sueños de Akira Kurosawa	Akira Kurosawa	Fantasía	1989	Japón	120
13	Rapsodia en Agosto	Akira Kurosawa	Drama	1991	Japón	97

Recuerda que la finalidad de establecer criterios y realizar filtros, es la de responder preguntas como por ejemplo, ¿cuantas películas tenemos del año 1974?, establece la condición y ejecuta la macro para responderla.

Nº	Titulo	Director	Genero	Año	País	Duración
				1974		

Si quieres establecer más de un criterio y lo haces en una misma fila, estas usando el operador de conjunción “Y” (*And*), como la respuesta a la pregunta; ¿cuantas películas tenemos, realizadas en Japón realizadas en 1980?

Nº	Titulo	Director	Genero	Año	País	Duración
				1980	Japón	

Si usas diferentes filas, estas usando el operador de disyunción “O” (Or), por ejemplo, para responder la pregunta; ¿cuantas películas tenemos de la India o del director Peter Greenaway?

Nº	Titulo	Director	Genero	Año	País	Duración
					India	
		Peter Greenaway				

Como declaramos en el descriptor de filtro, que se usarán expresiones regulares, puedes responder preguntas como; ¿cuantas películas empiezan con la letra “D”?

Nº	Titulo	Director	Genero	Año	País	Duración
	d.*					

Como habrás notado, las posibilidades son enormes. Te queda de tarea, establecer las condiciones para responder la siguiente pregunta; ¿cuales películas duran entre 60 y 90 minutos?, la primer pista es; nota que estamos solicitando un rango especifico, por lo tanto, tienes que usar el operador “Y” (AND) para resolverlo, y ya casi te dije todo, recuerda que el rango de criterios es solo eso un rango y no estamos limitados a cambiar lo que queramos en este rango, con lo que ya te resolví la tarea, a trabajar.

Nº	Titulo	Director	Genero	Año	País	Duración
						??

6.7.4 Subtotales

Los subtotales nos permiten obtener información de las áreas de datos, agrupar y realizar una operación determinada en un grupo de datos, en el siguiente ejemplo y continuando trabajando con nuestra base de datos de películas, obtenemos la cantidad de títulos por director.

```
Sub Subtotales1()
Dim oDoc As Object
Dim oRangosBD As Object
Dim oPeliculas As Object
Dim oRango As Object
Dim oDesSubTotal As Object
Dim mCamposSubTotal(0) As New com.sun.star.sheet.SubTotalColumn

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   'Nos aseguramos de que existe el rango
   If oRangosBD.hasByName( "Peliculas" ) Then
         'Obtenemos una referencia a los rangos origen
         oPeliculas = oRangosBD.getByName( "Peliculas" )
         'Referencia al rango de celdas origen
         oRango = oPeliculas.ReferredCells()
         'Creamos un nuevo subtotal
         oDesSubTotal = oRango.createSubTotalDescriptor( True )
         'Que ordene de forma ascendente el grupo seleccionado
         oDesSubTotal.EnableSort = True
         oDesSubTotal.SortAscending = True
         'La columna en la que se usara la función especificada
         mCamposSubTotal(0).Column = 1
         mCamposSubTotal(0).Function = com.sun.star.sheet.GeneralFunction.COUNT
         'Agregamos la operación, al grupo deseado
         oDesSubTotal.addNew( mCamposSubTotal,2 )
         'Aplicamos el subtotal
         oRango.applySubTotals( oDesSubTotal, True )
   End If

End Sub
```

El área de datos (*DataBaseRanges*), solo la utilizamos para acceder (*ReferredCells*) al rango de celdas origen, si al método del rango de celdas para crear el descriptor del subtotal (*createSubTotalDescriptor*), se le pasa como argumento un valor verdadero (*True*), te creara un nuevo descriptor vacío, si es falso (*False*), tomará el existente, si lo hay, del rango seleccionado. Es importante que establezcas que ordene el grupo, si es ascendente o descendente, queda a tu criterio y necesidades, pero si no ordenas el grupo, y dependiendo de como estén tus datos origen, te puede dar resultados incorrectos. En la matriz de columnas del subtotal (*com.sun.star.sheet.SubTotalColumn*), establecemos el campo (*Column*) en el cual queremos hacer una operación, determinada por la propiedad función (*Function*), a su vez, basada en la enumeración com.sun.star.sheet.GeneralFunction, que puede tomar los siguientes valores.

com.sun.star.sheet.GeneralFunction	*Valor*	*Valor en Interfaz*
com.sun.star.sheet.GeneralFunction.NONE	0	Ninguna
com.sun.star.sheet.GeneralFunction.AUTO	1	Automático
com.sun.star.sheet.GeneralFunction.SUM	2	Suma
com.sun.star.sheet.GeneralFunction.COUNT	3	Cuenta
com.sun.star.sheet.GeneralFunction.AVERAGE	4	Promedio
com.sun.star.sheet.GeneralFunction.MAX	5	Máximo

com.sun.star.sheet.GeneralFunction	Valor	Valor en Interfaz
com.sun.star.sheet.GeneralFunction.MIN	6	Mínimo
com.sun.star.sheet.GeneralFunction.PRODUCT	7	Producto
com.sun.star.sheet.GeneralFunction.COUNTNUMS	8	Cuenta solo números
com.sun.star.sheet.GeneralFunction.STDEV	9	Desviación estándar (Muestra)
com.sun.star.sheet.GeneralFunction.STDEVP	10	Desviación estándar (Población)
com.sun.star.sheet.GeneralFunction.VAR	11	Varianza (Muestra)
com.sun.star.sheet.GeneralFunction.VARP	12	Varianza (Población)

Al añadir los campos (*addNew*) al descriptor de subtotal, le pasamos como parámetros, la matriz de campos de columna del subtotal donde hemos establecido que campos y que operación haremos con ellos, ahí mismo, con el segundo parámetro, le indicamos por que campo queremos agrupar, por último, aplicamos (*applySubTotals*) los subtotales, cuyos parámetros son; el descriptor de subtotal y un valor booleano indicándole si deseamos reemplazar el subtotal actual (*True*), si lo hay, o agregamos al existente (*False*), usa con cuidado este parámetro, pues si agregas criterios "inconsistentes", el resultado puede ser impreciso y desastroso visualmente hablando.

La siguiente macro, borra cualquier subtotal existente en el rango del área de datos.

```
Sub Subtotales2()
Dim oDoc As Object
Dim oRangosBD As Object
Dim oPeliculas As Object
Dim oRango As Object

   oDoc = ThisComponent
   oRangosBD = oDoc.DataBaseRanges()
   If oRangosBD.hasByName( "Peliculas" ) Then
         oPeliculas = oRangosBD.getByName( "Peliculas" )
         oRango = oPeliculas.ReferredCells()
         'Quitamos el subtotal
         oRango.removeSubTotals()
   End If

End Sub
```

Como se muestra en el siguiente ejemplo, no es indispensable usar un área de datos para hacer uso de los subtotales, al ser métodos implementados en rangos de celda, podemos invocarlos desde cualquiera de estos. La siguiente macro, nos da el total de títulos por genero y suma la cantidad de minutos del mismo.

```
Sub Subtotales3()
Dim oDoc As Object
Dim oSel As Object
Dim oDesSubTotal As Object
Dim mCamposSubTotal(1) As New com.sun.star.sheet.SubTotalColumn

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()

   'Si es una sola celda o un rango de celdas
   If oSel.getImplementationName = "ScCellRangeObj" Or oSel.getImplementationName = "ScCellObj" Then
         oSel = oSel.getSpreadSheet.createCursorByRange(oSel)
         'Expandimos a la región actual
         oSel.collapseToCurrentRegion()
         'Creamos el descriptor a partir de la selección
```

```
        oDesSubTotal = oSel.createSubTotalDescriptor( True )
        oDesSubTotal.EnableSort = True
        oDesSubTotal.SortAscending = True

        'Columna de titulos
        mCamposSubTotal(0).Column = 1
        mCamposSubTotal(0).Function = com.sun.star.sheet.GeneralFunction.COUNT
        'Columna de minutos
        mCamposSubTotal(1).Column = 6
        mCamposSubTotal(1).Function = com.sun.star.sheet.GeneralFunction.SUM
        'Aplicamos al genero
        oDesSubTotal.addNew( mCamposSubTotal,3 )
        oSel.applySubTotals( oDesSubTotal, True )
    Else
        MsgBox "No es un rango de celdas"
    End If

End Sub
```

El siguiente ejemplo, nos muestra cuantos títulos tenemos por país, y después otro subtotal nos indica cuantos títulos por genero, “dentro” de cada país tenemos.

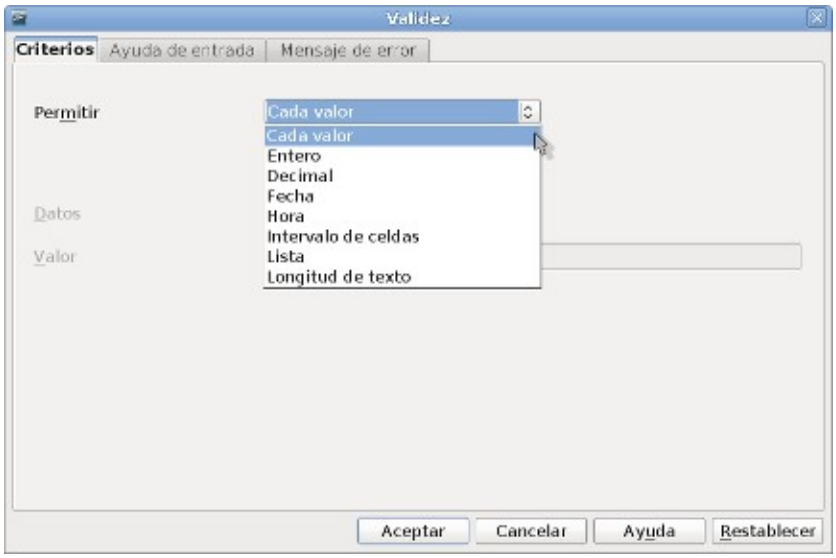
```
Sub Subtotales4()
Dim oDoc As Object
Dim oSel As Object
Dim oDesSubTotal As Object
Dim mCamposSubTotal(0) As New com.sun.star.sheet.SubTotalColumn

    oDoc = ThisComponent
    oSel = oDoc.getCurrentSelection()

    'Si es una sola celda o un rango de celdas
    If oSel.getImplementationName = "ScCellRangeObj" Or oSel.getImplementationName = "ScCellObj" Then
        oSel = oSel.getSpreadSheet.createCursorByRange(oSel)
        oSel.collapseToCurrentRegion()
        oDesSubTotal = oSel.createSubTotalDescriptor( True )
        oDesSubTotal.EnableSort = True
        oDesSubTotal.SortAscending = True

        'Columna de títulos
        mCamposSubTotal(0).Column = 1
        mCamposSubTotal(0).Function = com.sun.star.sheet.GeneralFunction.COUNT
        'Aplicamos al país
        oDesSubTotal.addNew( mCamposSubTotal,5 )

        'Columna de títulos
        mCamposSubTotal(0).Column = 1
        mCamposSubTotal(0).Function = com.sun.star.sheet.GeneralFunction.COUNT
        'Aplicamos al genero
        oDesSubTotal.addNew( mCamposSubTotal,3 )

        oSel.applySubTotals( oDesSubTotal, True )
    Else
        MsgBox "No es un rango de celdas"
    End If

End Sub
```

Observa, que hemos establecido la matriz de campos dos veces y usado el método *addNew* también dos veces para lograr el resultado deseado. Solo puedes sacar subtotales por hasta un máximo de tres grupos. El orden en que establezcas y agregues los campos, será el orden de precedencia de los subtotales. La obtención de subtotales, tiene un valor agregado bastante útil, al realizar la operación establecida en algún campo, la herramienta hace una

comparación “exacta” de cada valor, por lo que te podrás dar cuenta si tienes campos con, por ejemplo, espacios vacíos de más, al principio o al final de el, o en el caso de tener valores de campo muy similares, si un registro tiene solo un error de “dedo”, con los subtotales te podrás dar cuenta de forma visual muy fácilmente, sobre todo cuando haces uso de la función cuenta, revisa los registros que solo tengan un solo registro, por supuesto, tu debes determinar si es correcto o no. Para ejemplificar esto, si estas usando la misma base de datos que yo, saca un subtotal por genero, busca y observa los géneros “Comedia” y “Drama”, y cuéntame cual es el error. Esa es tu tarea que no es mucha.

6.7.5 Validando datos

A estas alturas del libro, te habrás dado cuenta de la importancia de validar la información que el usuario le proporciona a un programa, una gran cantidad de programas, fallan en este tema de, creo yo, alta prioridad. Calc cuenta con una herramienta llamada Validez, presente en el menú Datos, que nos permite establecer criterios en celdas para la captura de información por parte del usuario y limitarlo en la medida de lo posible, a capturar dados válidos en el programa, informando, si así lo deseas, con cuadros de mensaje para ayudarle a introducir la información correctamente.



Veamos como establecer estas opciones por código. En el siguiente ejemplo, establecemos que las celdas seleccionadas, solo puedan aceptar números enteros del 1 al 12, es decir, un entero correspondiente a un mes del año.

```
Sub ValidarDatos1()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   'Validamos que sea una celda o un rango de celdas
   If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
         'Obtenemos la estructura validación
         oValidacion = oSel.getPropertyValue("Validation")
         'Establecemos sus propiedades
         With oValidacion
               'El tipo de validación
               .Type = com.sun.star.sheet.ValidationType.WHOLE
```

```
                'El operador de la validación
                .setOperator ( com.sun.star.sheet.ConditionOperator.BETWEEN )
                'Establecemos la primer condición
                .setFormula1 ( "1" )
                'Establecemos la segunda
                .setFormula2 ( "12" )
                'Que ignore las celdas en blanco
                .IgnoreBlankCells = True
                'Que muestre un mensaje al seleccionar la celda
                .ShowInputMessage = True
                'El titulo del mensaje
                .InputTitle = "Introduce el mes del año"
                'El mensaje
                .InputMessage = "Captura un número entre 1 y 12"
                'Que muestre un mensaje si la condición no se cumple
                .ShowErrorMessage = True
                'El estilo del mensaje de alerta
                .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.STOP
                'El titulo del mensaje de error
                .ErrorTitle = "Dato erroneo"
                'El mensaje de error
                .ErrorMessage = "El valor introducido no es un número entre 1 y 12"
            End With
            'Reinsertamos la propiedad para que surtan efecto los cambios
            oSel.setPropertyValue("Validation", oValidacion)
    End If

End Sub
```

Observa como estamos obteniendo la estructura de validación (*getPropertyValue*), pasándole como argumento, el nombre de la propiedad que nos interesa. Las restantes propiedades son: el tipo (*Type*), se refiere al tipo de validación que haremos y esta condicionado por la enumeración *com.sun.star.sheet.ValidationType*, cuyos posibles valores son.

com.sun.star.sheet.ValidationType	*Valor*	*Valor en Interfaz*
com.sun.star.sheet.ValidationType.ANY	0	Cualquier valor
com.sun.star.sheet.ValidationType.WHOLE	1	Entero
com.sun.star.sheet.ValidationType.DECIMAL	2	Decimal
com.sun.star.sheet.ValidationType.DATE	3	Fecha
com.sun.star.sheet.ValidationType.TIME	4	Hora
com.sun.star.sheet.ValidationType.TEXT_LEN	5	Longitud de texto
com.sun.star.sheet.ValidationType.LIST	6	Lista
com.sun.star.sheet.ValidationType.CUSTOM	7	Formula

Después, establecemos el operador (*setOperator*), para evaluar la condición, esta propiedad puede tomar los siguientes valores.

com.sun.star.sheet.ConditionOperator	*Valor*	*Valor en Interfaz*
com.sun.star.sheet.ConditionOperator.NONE	0	Ninguna
com.sun.star.sheet.ConditionOperator.EQUAL	1	Igual
com.sun.star.sheet.ConditionOperator.NOT_EQUAL	2	Distinta de
com.sun.star.sheet.ConditionOperator.GREATER	3	Mayor que
com.sun.star.sheet.ConditionOperator.GREATER_EQUAL	4	Mayor o igual
com.sun.star.sheet.ConditionOperator.LESS	5	Menor que

com.sun.star.sheet.ConditionOperator	Valor	Valor en Interfaz
com.sun.star.sheet.ConditionOperator.LESS_EQUAL	6	Menor o igual
com.sun.star.sheet.ConditionOperator.BETWEEN	7	Entre
com.sun.star.sheet.ConditionOperator.NOT_BETWEEN	8	No entre
com.sun.star.sheet.ConditionOperator.FORMULA	9	Formula

Dependiendo de que operador selecciones, puede que necesites establecer el primer valor de comparación (*setFormula1*), y también el segundo (*setFormula2*), comúnmente los operadores “entre” (*BETWEEN*), y “no entre” (*NOT_BETWEEN*), son los que requieren los dos valores. Podemos determinar si la validación ignora o no, las celdas en blanco (*IgnoreBlankCells*), pero debes de saber que esta propiedad no determina el validar celdas vacía o no, si no que trabaja en conjunto la utilidad Detective del menú Herramientas, para determinar si una celda vacía se considera error o no. Podemos mostrar un mensaje (*ShowInputMessage*), cuando el usuario seleccione una celda, establecer su titulo (*InputTitle*), y el mensaje que verá (*InputMessage*), este texto orienta al usuario sobre los valores a introducir. En caso de que el valor capturado por el usuario, no satisfaga las condiciones de la validación, podemos mostrar un mensaje de error (*ShowErrorMessage*), de determinado estilo (*ErrorAlertStyle*), este estilo, determinará la acción a tomar con el valor capturado, este estilo esta determinado por los siguientes valores.

com.sun.star.sheet.ValidationAlertStyle	Valor	Valor en Interfaz
com.sun.star.sheet.ValidationAlertStyle.STOP	0	Stop
com.sun.star.sheet.ValidationAlertStyle.WARNING	1	Advertencia
com.sun.star.sheet.ValidationAlertStyle.INFO	2	Información
com.sun.star.sheet.ValidationAlertStyle.MACRO	3	Macro

Si estableces el valor en “stop” (*STOP*), se mostrara al usuario un mensaje (*ErrorMessage*) con un titulo (*ErrorTitle*) para informarle del error, al aceptar, la celda regresará al valor inmediato anterior, en los casos de “advertencia” (*WARNING*) e “información” (*INFO*), quedará a criterio del usuario si acepta o no el nuevo valor aun y cuando no cumpla la condición y en el caso de la opción “macro” (*MACRO*), puedes escoger una macro a ejecutar, para, por ejemplo, mostrar un mensaje más elaborado o realizar una operación más compleja. Por ultimo, es importante, “reinsertar” (*setPropertyValue*) la propiedad al objeto para que los cambios surtan efecto.

Para quitar una validación, solo establece el tipo (*Type*) en cualquier valor (*ANY*) y deshabilita el mensaje de entrada (*ShowInputMessage*).

```
Sub ValidarDatos2()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   'Validamos que sea una celda o un rango de celdas
   If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
         'Obtenemos la estructura validación
         oValidacion = oSel.getPropertyValue("Validation")
         'El tipo de validación
         oValidacion.Type = com.sun.star.sheet.ValidationType.ANY
         'Que muestre un mensaje al seleccionar la celda
         oValidacion.ShowInputMessage = False
         'Reinsertamos la propiedad para que surtan efecto los cambios
         oSel.setPropertyValue("Validation", oValidacion)
   End If
```

```
End Sub
```

El tipo decimal, te permite capturar números con decimales.

```
Sub ValidarDatos3()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
         oValidacion = oSel.getPropertyValue("Validation")
         With oValidacion
               .Type = com.sun.star.sheet.ValidationType.DECIMAL
               .setOperator ( com.sun.star.sheet.ConditionOperator.BETWEEN )
               .setFormula1 ( "0" )
               .setFormula2 ( "10" )
               .IgnoreBlankCells = True
               .ShowInputMessage = True
               .InputTitle = "Introduce la calificacion"
               .InputMessage = "Puedes usar decimales"
               .ShowErrorMessage = True
               .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.INFO
               .ErrorTitle = "Dato erroneo"
               .ErrorMessage = "El valor introducido no es válido"
         End With
         'Reinsertamos la propiedad para que surtan efecto los cambios
         oSel.setPropertyValue("Validation", oValidacion)
   End If

End Sub
```

En el siguiente ejemplo validamos que no sea una fecha futura, nota el uso de la función incorporada de Calc.

```
Sub ValidarDatos4()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
         oValidacion = oSel.getPropertyValue("Validation")
         With oValidacion
               .Type = com.sun.star.sheet.ValidationType.DATE
               .setOperator ( com.sun.star.sheet.ConditionOperator.LESS_EQUAL )
               'Usamos una formula como condición
               .setFormula1 ( "TODAY()" )
               .IgnoreBlankCells = True
               .ShowInputMessage = True
               .InputTitle = "Fecha de Nacimiento"
               .InputMessage = "La fecha no puede ser futura"
               .ShowErrorMessage = True
               .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.WARNING
               .ErrorTitle = "Dato erroneo"
               .ErrorMessage = "El valor introducido no es válido"
         End With
         oSel.setPropertyValue("Validation", oValidacion)
   End If
```

```
End Sub
```

Ahora, solo puede capturar una hora que no sea entre 1 p.m. 3 p.m.

```
Sub ValidarDatos5()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
         oValidacion = oSel.getPropertyValue("Validation")
         With oValidacion
               .Type = com.sun.star.sheet.ValidationType.TIME
               .setOperator ( com.sun.star.sheet.ConditionOperator.NOT_BETWEEN )
               'Introducir una hora que no sea entre 1 y 3 p.m.
               .setFormula1 ( "TIME(13;0;0" )
               .setFormula2 ( "TIME(15;0;0" )
               .IgnoreBlankCells = True
               .ShowInputMessage = True
               .InputTitle = "Hora de salida"
               .InputMessage = "La hora de salir"
               .ShowErrorMessage = True
               .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.STOP
               .ErrorTitle = "Dato erróneo"
               .ErrorMessage = "El valor introducido no es válido"
         End With
         oSel.setPropertyValue("Validation", oValidacion)
   End If

End Sub
```

Observa que estamos haciendo uso de la función de Calc, TIEMPO (*TIME*), que te devuelve el número de serie de la hora pasada, esto es necesario para establecer los limites correctamente. En el siguiente ejemplo, establecemos que los valores los tome desde un rango de celdas, observa que las referencias al rango son absolutas.

```
Sub ValidarDatos6()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
         oValidacion = oSel.getPropertyValue("Validation")
         With oValidacion
               .Type = com.sun.star.sheet.ValidationType.LIST
               'Introducir valores desde un rango de celdas
               .setFormula1 ( "$I$2:$I$8" )
               .ShowErrorMessage = True
               .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.STOP
               .ErrorTitle = "Dato erróneo"
               .ErrorMessage = "El valor introducido no es válido"
         End With
         oSel.setPropertyValue("Validation", oValidacion)
   End If

End Sub
```

También podemos establecer el rango desde un rango de celdas con nombre.

```
Sub ValidarDatos7()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
         oValidacion = oSel.getPropertyValue("Validation")
         With oValidacion
               .Type = com.sun.star.sheet.ValidationType.LIST
               'Introducir valores desde un rango de celdas
               .setFormula1 ( "valores" )
               .ShowList = 2
               .ShowErrorMessage = True
               .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.STOP
               .ErrorTitle = "Dato erróneo"
               .ErrorMessage = "El valor introducido no es válido"
         End With
         oSel.setPropertyValue("Validation", oValidacion)
   End If

End Sub
```

Recuerda que los rangos con nombre los establecemos, en la interfaz del usuario, desde el cuadro de nombres de la barra de formulas, o por código, como hemos aprendido en este libro, aunque también puedes usar un nombre de área de datos definida en el menú *Datos | Definir...,* de hecho, puedes usar cualquier texto o formula que te devuelva un rango de celdas válido, si el rango de celdas tiene más de una columna, solo se usara la primer columna.

Observa que hemos usado una nueva propiedad (*ShowList*), cuando el tipo de la validación esta establecida en “lista” (*LIST*), podemos determinar si mostramos la flecha de lista de selección, el valor 2 determina que se muestre con los valores ordenados de forma ascendente, el valor 1 que se muestre pero que no ordene los valores, que los muestre tal y como están en el origen, y el valor 0 que no se muestre la lista de selección, algo no muy recomendable cuando se usa una lista. Estos valores corresponden a.

com.sun.star.sheet.TableValidationVisibility	*Valor*	*Valor en Interfaz*
com.sun.star.sheet.TableValidationVisibility.INVISIBLE	0	No mostrar lista
com.sun.star.sheet.TableValidationVisibility.UNSORTED	1	Mostrar desordenada
com.sun.star.sheet.TableValidationVisibility.SORTEDASCENDING	2	Mostrar ordenada ascendente

En el siguiente ejemplo, establecemos el origen de celdas con formulas de Calc, esta formula debe estar bien construida, si no, no te dará el rango correcto.

```
Sub ValidarDatos8()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
         oValidacion = oSel.getPropertyValue("Validation")
         With oValidacion
               .Type = com.sun.star.sheet.ValidationType.LIST
               'Introducir valores desde un rango de celdas
```

```
                .setFormula1 ( "INDIRECT(ADDRESS(1;1)&"":""&ADDRESS(10;1))" )
                .ShowList = 2
                .ShowErrorMessage = True
                .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.STOP
                .ErrorTitle = "Dato erroneo"
                .ErrorMessage = "El valor introducido no es válido"
          End With
          oSel.setPropertyValue("Validation", oValidacion)
    End If

End Sub
```

Recuerda que la función DIRECCION (ADDRESS), nos devuelve una referencia en forma de texto, de acuerdo a los parámetros pasados, después, la función INDIRECTO (INDIRECT), nos devuelve, de una cadena de texto que tenga un rango de celda válido, la referencia a dicha celda. Como a la función DIRECCION se le pueda establecer entre sus parámetros el nombre de una hoja, te queda de tarea modificar la macro anterior para que el rango de celdas de la validación, este en una hoja diferente. El poder de hacerlo con funciones o desde código, es que la lista de validación la puedes actualizar dinámicamente, por ejemplo, una lista de clientes que va creciendo confirme vamos agregando registros o cualquier otro listado que vaya creciendo en sentido vertical.

Otra posibilidad, es introducir una lista fija de valores, o semifija, pues desde código la podemos actualizar siempre que queramos, veamos como.

```
Sub ValidarDatos9()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
         oValidacion = oSel.getPropertyValue("Validation")
         With oValidacion
               .Type = com.sun.star.sheet.ValidationType.LIST
               'Introducir valores fijos
               .setFormula1 ( "VALOR1;VALOR2;VALOR3" )
               .ShowList = 2
               .ShowErrorMessage = True
               .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.STOP
               .ErrorTitle = "Dato erróneo"
               .ErrorMessage = "El valor introducido no es válido"
         End With
         oSel.setPropertyValue("Validation", oValidacion)
   End If

End Sub
```

Ve a la interfaz del usuario y observa como “aparentemente”, si agrego los valores, pero observa que están en minúsculas, cuando nosotros los agregamos en mayúsculas, verifica desde la interfaz de usuario, desde *Datos* | *Validez...*, que esta forma de introducir los datos fijos, la detecta como si fuera un intervalo de celdas, lo cual es incorrecto, para que realmente la detecte como una lista “fija” de valores, tienes que pasarle cada valor como una cadena, como en el siguiente ejemplo.

```
Sub ValidarDatos10()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object
```

```
    oDoc = ThisComponent
    oSel = oDoc.getCurrentSelection()
    If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
          oValidacion = oSel.getPropertyValue("Validation")
          With oValidacion
                .Type = com.sun.star.sheet.ValidationType.LIST
                'Introducir valores fijos
                .setFormula1 ( """VALOR1"""&";"&"""VALOR2"""&";"&"""VALOR3""" )
                .ShowList = 2
                .ShowErrorMessage = True
                .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.STOP
                .ErrorTitle = "Dato erroneo"
                .ErrorMessage = "El valor introducido no es válido"
          End With
          oSel.setPropertyValue("Validation", oValidacion)
    End If

End Sub
```

Pero ve que cantidad de comillas, ¿y si son muchos valores?, podemos hacer una función que agregue las comillas por nosotros, como en el siguiente ejemplo.

```
Sub ValidarDatos11
Dim sTmp As String
Dim m
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

    oDoc = ThisComponent
    oSel = oDoc.getCurrentSelection()
    If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
          oValidacion = oSel.getPropertyValue("Validation")
          With oValidacion
                'Valores a agregar
                m = Array("Uno","Dos","Tres","Cuatro","Cinco")
                'Los juntamos
                sTmp = JuntarEnLista(m)
                .Type = com.sun.star.sheet.ValidationType.LIST
                .ShowList = 2
                .setFormula1( sTmp )
          End With
          oSel.setPropertyValue("Validation", oValidacion)
    End If
End Sub


'Toma una matriz y le agrega comillas a cada elemento
'Devuelve una cadena con los elementos, separados por ";"
Function JuntarEnLista( ByVal m ) As String
Dim co1 As Long

    For co1 = LBound(m) To UBound(m)
          m(co1) = """" & m(co1) & """"
    Next
    JuntarEnLista = Join( m, ";" )

End Function
```

Ahora si, es un poco más fácil agregar valores a una lista. El siguiente ejemplo te limita la entrada a una palabra de entre 5 y 10 letras, no te acepta solo números, solo letras o alfanuméricas.

```
Sub ValidarDatos12()
Dim oDoc As Object
Dim oSel As Object
Dim oValidacion As Object

   oDoc = ThisComponent
   oSel = oDoc.getCurrentSelection()
   If oSel.getImplementationName = "ScCellObj" Or oSel.getImplementationName = "ScCellRangeObj" Then
         oValidacion = oSel.getPropertyValue("Validation")
         With oValidacion
               .Type = com.sun.star.sheet.ValidationType.TEXT_LEN
               .setOperator ( com.sun.star.sheet.ConditionOperator.BETWEEN )
               .setFormula1 ( "5" )
               .setFormula2 ( "10" )
               .IgnoreBlankCells = True
               .ShowErrorMessage = True
               .ErrorAlertStyle = com.sun.star.sheet.ValidationAlertStyle.STOP
               .ErrorTitle = "Longitud errónea"
               .ErrorMessage = "La palabra debe ser de entre 5 y 10 caracteres"
         End With
         'Reinsertamos la propiedad para que surtan efecto los cambios
         oSel.setPropertyValue("Validation", oValidacion)
   End If

End Sub
```

Tanto en la interfaz del usuario y aun más desde código, la validación de datos es un tema central en la programación, no lo tomes como un tema menor, es preferible agregar unas cuantas líneas más de código a correr el riesgo de que un dato inconsistente nos devuelva información inconsistente. Tienes instrucciones limitadas pero suficientes para evitarlo y tienes algo ilimitado; imaginación, inteligencia, pero sobre todo, sentido común, explotalos.

6.7.6 Agrupando datos

Cuando se maneja una gran cantidad de información, encontrar una manera simple y rápida de agrupar datos, puede ser la diferencia entre eficiencia e ineficiencia, los filtros, el ordenar y los subtotales, son herramientas que pueden apoyarnos. Cuando simplemente necesitemos agrupar por un rango de columnas o filas (que no es más que ocultarlas y mostrarlas de acuerdo a nuestras necesidades) podemos probar lo siguiente.

```
Sub Agrupar1()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   'Establecemos las propiedades del grupo
   With oRango
         'La hoja donde se creará
         .Sheet = oHojaActiva.getRangeAddress.Sheet
         'La columna de inicio
         .StartColumn = 0
         'La columan de fin
         .EndColumn = 4
   End With
   'Creamos el grupo,  0 = por columnas
```

```
    oHojaActiva.group( oRango, 0 )

End Sub
```

La macro anterior nos agrupará las columnas 1 a 5 en la interfaz del usuario, es muy importante que validez que la hoja y el rango de columnas establecidas, estén dentro de rangos válidos, por ejemplo, que el índice de hoja exista, si no, puedes provocar una caída de la aplicación, estamos verificando el ¿por que? de esto.

Observa como al agrupar (*group*), le pasamos una estructura de dirección de un rango (*CellRangeAddress*), vista muchas veces en este libro y como segundo argumento, si queremos que se agrupe por columnas (*0*) o por filas (*1*).

Ahora, agrupamos las primeras diez filas de la hoja activa.

```
Sub Agrupar2()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()

   With oRango
         .Sheet = oHojaActiva.getRangeAddress.Sheet
         'Ahora establecemos el rango de filas
         .StartRow = 0
         .EndRow = 9
   End With
   'Y agrupamos,  1 = por filas
   oHojaActiva.group( oRango, 1 )

End Sub
```

Por supuesto puedes establecer con la misma estructura, tanto las columnas como las filas como en el siguiente ejemplo.

```
Sub Agrupar3()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()

   With oRango
         .Sheet = oHojaActiva.getRangeAddress.Sheet
         .StartColumn = 9
         .EndColumn = 19
         .StartRow = 9
         .EndRow = 19
   End With
   'Agrupamos por columnas 10 a la 20
   oHojaActiva.group( oRango, 0 )
   'Agrupamos por filas 10 a 20
   oHojaActiva.group( oRango, 1 )

End Sub
```

Desagrupar (*ungroup*) es trivial, usa los mismos argumentos que para agrupar.

```
Sub Agrupar4()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()

   With oRango
         .Sheet = oHojaActiva.getRangeAddress.Sheet
         .StartColumn = 0
         .EndColumn = 4
         .StartRow = 0
         .EndRow = 9
   End With

   'Deagrupamos filas y columnas
   oHojaActiva.ungroup( oRango, 0 )
   oHojaActiva.ungroup( oRango, 1 )

End Sub
```

El rango usado para desagrupar (*ungroup*), no tiene que coincidir exactamente, en tamaño al rango original de agrupamiento, es suficiente con que este “dentro” del rango con que se agrupó, antes de comprobarlo, desde la hoja donde estés haciendo estas pruebas, ve al menú *Datos* | *Agrupar y Esquema* | *Eliminar*, lo que borrara cualquier grupo creado, después, ejecuta la primer macro siguiente e inmediatamente después, la segunda.

```
Sub Agrupar5()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()

   With oRango
         .Sheet = oHojaActiva.getRangeAddress.Sheet
         .StartColumn = 0
         .EndColumn = 9
         .StartRow = 0
         .EndRow = 19
   End With

   'Agrupamos filas (1-20) y columnas (1-10)
   oHojaActiva.group( oRango, 0 )
   oHojaActiva.group( oRango, 1 )

End Sub


Sub Agrupar6()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()

   With oRango
         .Sheet = oHojaActiva.getRangeAddress.Sheet
```

```
        'Solo indicamos la primer columna y fila
        .StartColumn = 0
        .EndColumn = 0
        .StartRow = 0
        .EndRow = 0
    End With

    'Desagrupamos filas y columnas
    oHojaActiva.ungroup( oRango, 0 )
    oHojaActiva.ungroup( oRango, 1 )

End Sub
```

Observa como agrupamos por un rango y desagrupamos por otro, donde solo nos tenemos que asegurar que, el rango a desagrupar, este “dentro” del rango agrupado.

Cuando eliminas los grupos desde la interfaz del usuario, si el grupo esta contraído, las columnas o filas ocultas, se mostrarán inmediatamente, no sucede lo mismo cuando lo haces por código, vuelve a realizar la prueba anterior, pero asegurate, de que, antes de desagrupar, contrae el grupo para ocultar su contenido, ahora sí, ejecuta la macro para desagrupar, notaras que el rango usado seguirá oculto, para evitar esto, antes de desagrupar, asegurate de mostrar el detalle del rango como en el siguiente ejemplo.

```
Sub Agrupar7()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()

   With oRango
         .Sheet = oHojaActiva.getRangeAddress.Sheet
         'Solo indicamos la primer columna y fila
         .StartColumn = 0
         .EndColumn = 9
         .StartRow = 0
         .EndRow = 19
   End With

   'Expandimos el grupo contenido en el rango
   oHojaActiva.showDetail( oRango )

   'Desagrupamos filas y columnas
   oHojaActiva.ungroup( oRango, 0 )
   oHojaActiva.ungroup( oRango, 1 )

End Sub
```

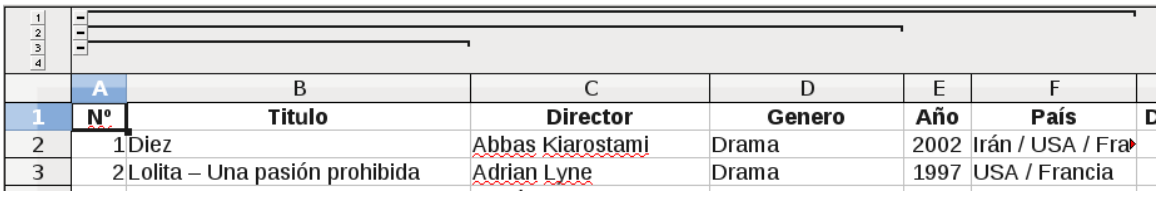
El ejemplo anterior funcionará, siempre y cuando, el rango agrupado (*group*), corresponda “exactamente” con el rango mostrado (*showDetail*) y con el rango desagrupado (*ungroup*), lo cual, en ocasiones, podría no ser tan fácil de conocer. Si lo que quieres es eliminar cualquier grupo existente en una hoja, usamos.

```
Sub Agrupar8()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
```

```
    'Eliminamos cualquier grupo de la hoja
    oHojaActiva.clearOutline()

End Sub
```

Con lo cual, es mucho más sencillo, eliminar cualquier grupo y volver a crear el necesario. Cuando agregas grupos, dependiendo de si exista o no previamente uno en el rango pasado, sucederá cualquiera de las siguientes acciones; si el rango ya contiene un grupo, se creará un nuevo nivel de agrupamiento, puedes crear hasta ocho niveles de ellos, si el rango no contiene un grupo, este, se agregará al mismo nivel, veamoslo con ejemplos, en el primero comprobamos que se agregan los grupos en niveles.

```
Sub Agrupar9()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oHojaActiva.clearOutline()
   With oRango
         .Sheet = oHojaActiva.getRangeAddress.Sheet
         .StartColumn = 0
         .EndColumn = 1
   End With
   'Agrupamos las columnas 1-2
   oHojaActiva.group( oRango, 0 )

   With oRango
         .StartColumn = 0
         .EndColumn = 3
   End With
   'Agrupamos las columnas 1-4
   oHojaActiva.group( oRango, 0 )

   With oRango
         .StartColumn = 0
         .EndColumn = 5
   End With
   'Agrupamos las columnas 1-6
   oHojaActiva.group( oRango, 0 )

End Sub
```



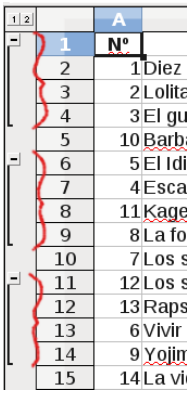
	A	B	C	D	E	F
1	Nº	Titulo	Director	Genero	Año	País
2	1	Diez	Abbas Kiarostami	Drama	2002	Irán / USA / Fra
3	2	Lolita – Una pasión prohibida	Adrian Lyne	Drama	1997	USA / Francia

Ahora veamos como agregar al mismo nivel.

```
Sub Agrupar10()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oHojaActiva.clearOutline()
```

```
    With oRango
          .Sheet = oHojaActiva.getRangeAddress.Sheet
          .StartRow = 0
          .EndRow = 3
    End With
    'Agrupamos la fila 1 a 4
    oHojaActiva.group( oRango, 1 )

    With oRango
          .StartRow = 5
          .EndRow = 8
    End With
    'Agrupamos la fila 4 a 7
    oHojaActiva.group( oRango, 1 )

    With oRango
          .StartRow = 10
          .EndRow = 13
    End With
    'Agrupamos la fila 9 a 12
    oHojaActiva.group( oRango, 1 )

End Sub
```

Y lo comprobamos:



Puedes crear un grupo y ocultarlo inmediatamente, como en.

```
Sub Agrupar11()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

    oDoc = ThisComponent
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()

    With oRango
          .Sheet = oHojaActiva.getRangeAddress.Sheet
          .StartColumn = 0
          .EndColumn = 9
    End With
    oHojaActiva.group( oRango, 0 )

    'Ocultamos el grupo contenido en el rango
    oHojaActiva.hideDetail( oRango )

End Sub
```

A diferencia de mostrar un grupo (*showDetail*), cuando lo ocultas (*hideDetail*), el rango no necesariamente debe ser del mismo tamaño, con que apunte a cualquier celda dentro del rango a ocultar, funcionará. También tienes la posibilidad de mostrar cualquier nivel de agrupamiento, los demás niveles hacia arriba, si los hay, se cerraran.

```
Sub Agrupar12()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oRango As New com.sun.star.table.CellRangeAddress

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()

   With oRango
         .Sheet = oHojaActiva.getRangeAddress.Sheet
         .StartRow = 0
         .EndRow = 4
   End With
   oHojaActiva.group( oRango, 1 )

   With oRango
         .StartRow = 0
         .EndRow = 9
   End With
   oHojaActiva.group( oRango, 1 )

   With oRango
         .StartRow = 0
         .EndRow = 14
   End With
   oHojaActiva.group( oRango, 1 )

   With oRango
         .StartRow = 0
         .EndRow = 19
   End With
   oHojaActiva.group( oRango, 1 )
   'Mostramos el nivel 2
   oHojaActiva.showLevel( 2, 1 )

End Sub
```

El método para mostrar un nivel especifico (*showLevel*), solo usa dos parámetros, el nivel a mostrar como primero y como segundo parámetro, si el nivel a mostrar esta en columnas (*0*) o en filas (*1*). Para terminar este tema, veamos un ejemplo practico bastante útil, para que veas su uso, procura llamarla, teniendo el cursor dentro de un rango de celdas que tenga al menos dos pantallas de datos en sentido vertical, es decir, de filas, si tienes más filas, es mucho mejor pues se paginara más.

```
Sub Agrupar13()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oSel As Object
Dim oCursor As Object
Dim iNumFilPag As Integer
Dim oRango As New com.sun.star.table.CellRangeAddress
Dim co1 As Long

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oSel = oDoc.getCurrentSelection()
   'Obligamos a seleccionar solo una celda
   If oSel.getImplementationName = "ScCellObj" Then
```

```
        'Creamos un cursor a partir de la celda seleccionada
        oCursor = oHojaActiva.createCursorByRange( oSel )
        'Expandimos a la región actual
        oCursor.collapseToCurrentRegion()
        'Obtenemos el número de filas visibles por página
        iNumFilPag = oDoc.getCurrentController.getVisibleRange.EndRow -
oDoc.getCurrentController.getVisibleRange.StartRow - 1
        'Establecemos el primer nivel de agrupamiento en el total de filas
        With oRango
              .Sheet = oHojaActiva.getRangeAddress.Sheet
              .StartRow = 0
              .EndRow = oCursor.getRangeAddress.EndRow
        End With
        oHojaActiva.group( oRango, 1 )

        'Agregamos un segundo nivel de agrupamiento, página por página
        For co1 = 0 To oCursor.getRangeAddress.EndRow Step iNumFilPag
              oRango.StartRow = co1 + 1
              oRango.EndRow = co1 + iNumFilPag - 1
              oHojaActiva.group( oRango, 1 )
        Next
        'Para el ultimo rango si no es exacto
        If co1 > oCursor.getRangeAddress.EndRow Then
              oRango.StartRow = co1 - iNumFilPag + 1
              oRango.EndRow = oCursor.getRangeAddress.EndRow
              oHojaActiva.group( oRango, 1 )
        End If
        oHojaActiva.ShowLevel( 1, 1 )
   Else
        MsgBox "Selecciona solo una celda"
   End If

End Sub
```

Y mira que bien queda.

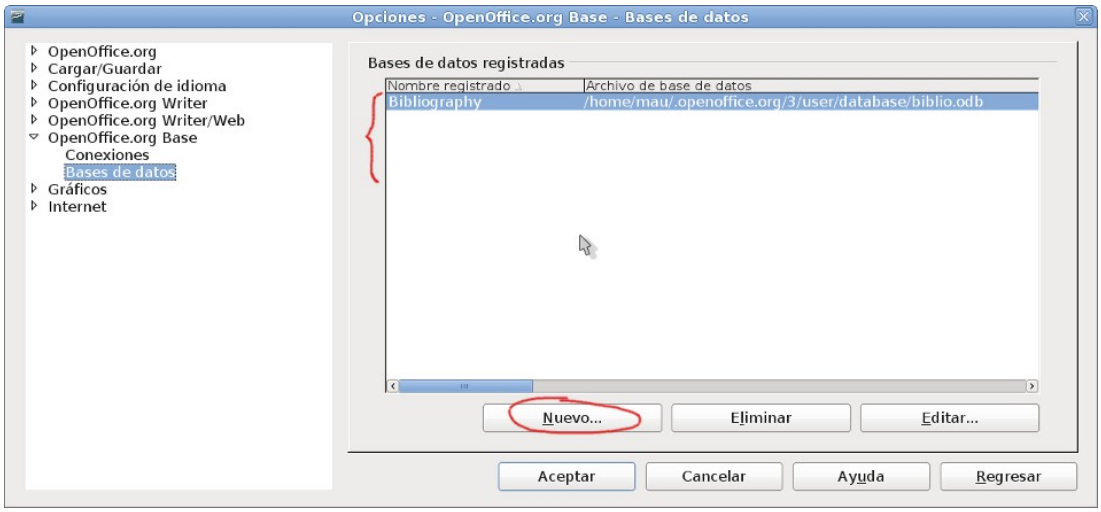
	A	B	C	D	E
1	Nº	Titulo	Director	Genero	Año
29	29	Niños del hombre	Alfonso Cuarón	Ciencia Ficción / D	2006
57	55	Tiempos modernos	Charles Chaplin	Comedia	1936
85	84	La familia	Ettore Scola	Drama	1987
113	112	Río Bravo	Howard Hawks	Western	1959
141	140	Manderley	Lars von Trier	Drama	2005
169	168	Caramelo	Nadine Labaki	Drama	2007
197	196	El mapa de la vida	Scot Elliott	Drama	1999
225	226	Ladrón de bicicletas	Vittorio de Sicca	Drama	1948
244					

Claro que el área no esta limitada a que tenga datos, puedes establecer tus criterios pero sobre todo tus necesidades particulares para agrupar.

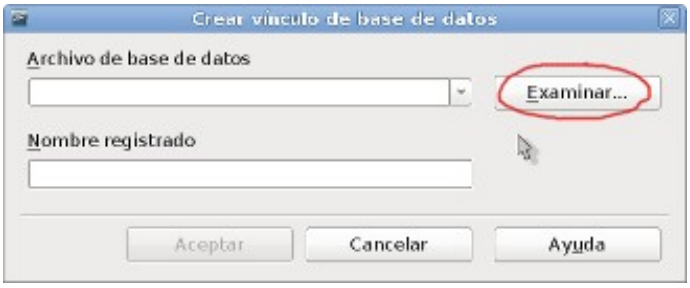
6.8 Bases de datos

La manipulación y administración de datos, es una de las necesidades más recurrentes en usuarios de hoja de cálculo, esto no es gratuito, pues las herramientas presentes en ellas, facilitan enormemente esta tarea. Para el mejor aprovechamiento de este capitulo, sería deseable que conocieras los siguientes conceptos: base de datos, tabla, campo, registro, consulta, SQL, clave primaria, índice, relaciones e integridad referencial. Dada la amplia documentación existente en castellano de estos temas, no duplicaremos esfuerzos y te queda de tarea investigar un poco acerca de ello si no los dominas.

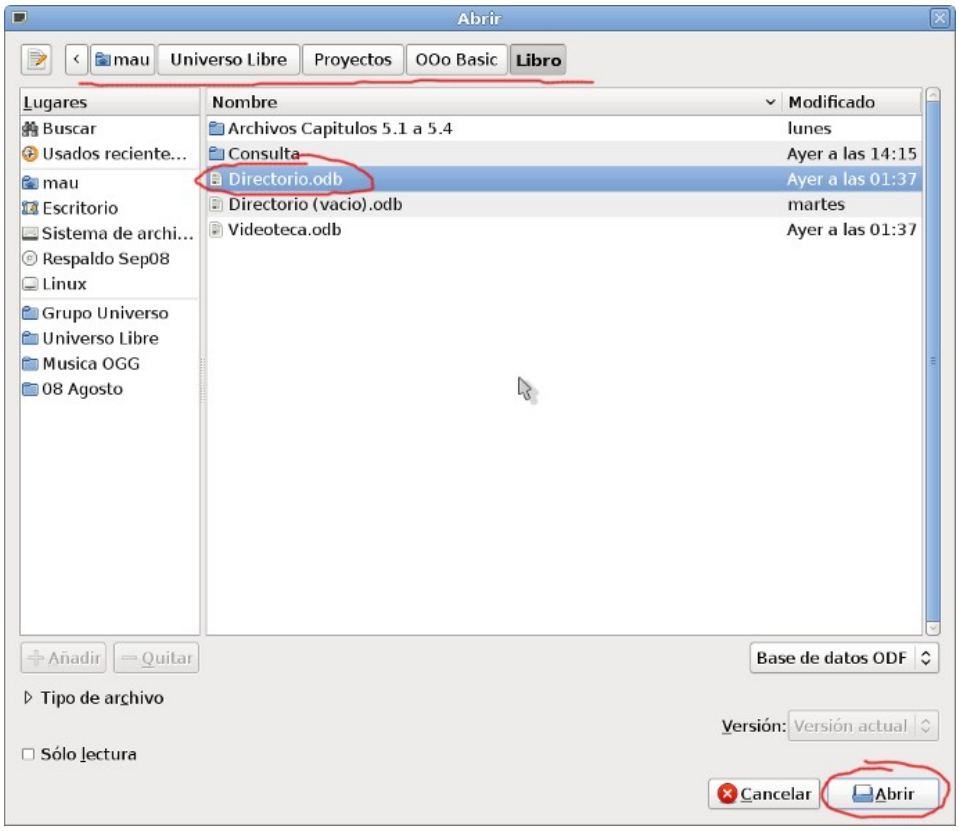
También, es indispensable que las bases de datos que usemos, al menos en este capitulo, estén registradas en OOo, para registrar una base de datos, seguimos los siguientes pasos, ve al menú *Herramientas* | *Opciones...*, selecciona la rama OpenOffice.org Base y después la subrama Bases de datos, donde tienes que ver las siguientes opciones.



Da clic en el botón de comando *Nuevo...*, para ve el siguiente cuadro de dialogo.

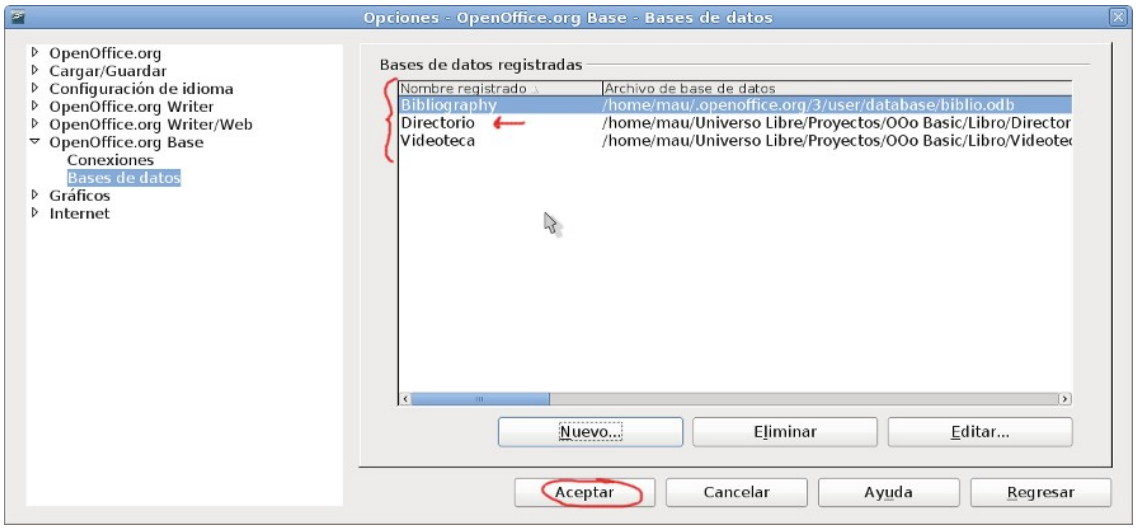


En el cual, le darás clic al botón de comando *Examinar...*, que te mostrará el conocido cuadro de dialogo para seleccionar archivos, navega hasta la ruta donde hayas guardado la base de datos o este localizada la base de datos que quieras registrar y selecciónala, por ultimo da un clic en el botón de comando *Abrir*.



Al dar clic en el botón de comando *Abrir*, tienes que regresar al cuadro de dialogo anterior, pero, ahora, tienes que ver la ruta completa del archivo que acabas de seleccionar, así como el nombre propuesto con el que se registrará la base de datos, este nombre puedes personalizarlo a tu gusto pues acepta espacios incluso, pero mi recomendación es que no uses ni espacios ni caracteres “extraños”, para nuestro ejemplo, dejaremos el propuesto.

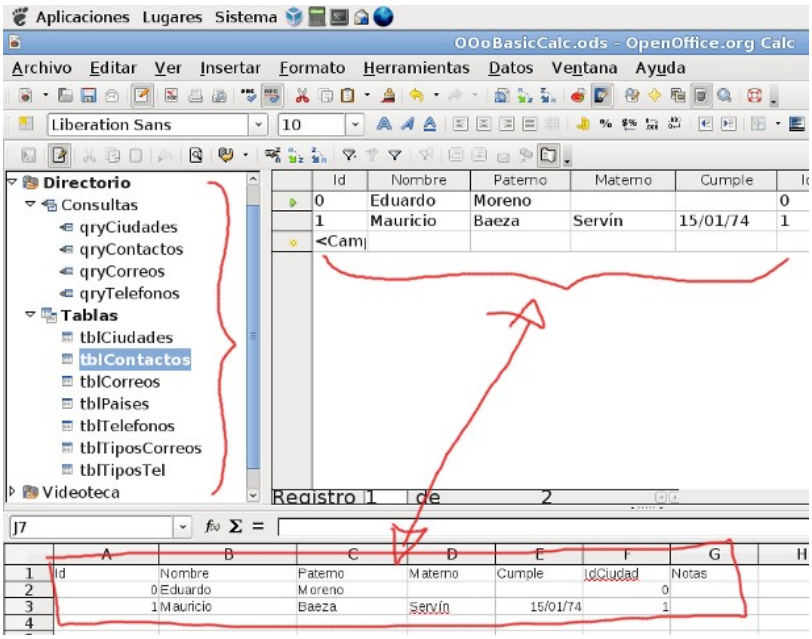
Da un clic en el botón de comando Aceptar para regresar al primero cuadro de dialogo, donde, ahora, tendrás que ver la nueva base de datos registrada.



Agrega de una vez la otra base de datos que acompaña a estos apuntes, se llama "Videoteca" y también la usaremos para nuestros ejemplos. Solo te resta cerrar este cuadro de dialogo con el botón de comando *Aceptar*.

6.8.1 Importando datos

Importa datos desde bases de datos registradas en OpenOffice.org, es trivial, solo tienes que presionar la tecla F4, que te mostrará estas bases en la ventana "*Fuente de datos*", junto con todos sus objetos; tablas y consultas, seleccionar la base de datos de nuestro interés, expandir para mostrar su contenido, seleccionar la tabla o consulta que quieras, el contenido de dicho objeto, te lo mostrará en el panel derecho del examinador de fuentes de datos, solo te resta seleccionar todo con el botón de seleccionar todo, arrastrar a tu documento y listo, tienes que tener los datos seleccionados, como en la siguiente imagen.



En este capitulo, aprenderemos a hacer esto mismo, pero claro, por código, con el cual, podrás comprobarlo más adelante, tenemos posibilidades “casi” ilimitadas para importar nuestros datos.

Para importar datos, de bases de datos previamente registradas en OpenOffice.org, usamos el método *doImport*, presente en rangos de datos, a este método, se le pasa una matriz de propiedades con las indicaciones de que y como queremos importar, por ejemplo.

```
Sub BasesDeDatos1()
Dim oHojaActiva As Object
Dim sBaseDatos As String
Dim sTabla As String
Dim mOpcBD(2) As New "com.sun.star.beans.PropertyValue"

   'El nombre de nuestra base de datos
   sBaseDatos = "Directorio"
   'La tabla que nos interesa traer
   sTabla = "tblContactos"
   'La hoja activa
   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Las propiedades de la importación, primero el nombre de la base de datos
   mOpcBD(0).Name = "DatabaseName"
   mOpcBD(0).Value = sBaseDatos
   'El tipo de objeto fuente
   mOpcBD(1).Name = "SourceType"
   mOpcBD(1).Value = com.sun.star.sheet.DataImportMode.TABLE
   'El nombre del objeto fuente
   mOpcBD(2).Name = "SourceObject"
   mOpcBD(2).Value = sTabla
   'Importamos la tabla, a partir de la celda A1
   oHojaActiva.getCellRangeByName("A1").doImport( mOpcBD() )

End Sub
```

Nota como estamos usando el nombre con que registramos la base de datos (*DatabaseName*), después, le estamos indicando el tipo de objeto (*SourceType*) que queremos importar, luego, el nombre del objeto (*SourceObject*) que queremos importar, por ultimo aplicamos la importación (*doImport*). El tipo de objeto esta determinado por la enumeración.

com.sun.star.sheet.DataImportMode	*Valor*	*Valor en Interfaz*
com.sun.star.sheet.DataImportMode.NONE	0	Ninguno
com.sun.star.sheet.DataImportMode.SQL	1	Instrucción SQL
com.sun.star.sheet.DataImportMode.TABLE	2	Tabla
com.sun.star.sheet.DataImportMode.QUERY	3	Consulta

Si estableces el tipo de objeto fuente en 0 (*NONE*), entonces tienes que pasarle una instrucción SQL, de lo contrario te dará error al importar. Todos los nombres de objetos (bases de datos, tablas, consultas, campos) distingue mayúsculas de minúsculas, por lo que tienes que ser muy cuidadoso de como nombrarlos y usar exactamente este nombre al importar, de lo contrario te puede alguno de los siguientes errores:

Cuando el nombre de la base de datos no corresponde.

Cuando la tabla este incorrecta.

En el siguiente ejemplo, importamos una tabla diferente en una celda diferente.

```
Sub BasesDeDatos2()
Dim oHojaActiva As Object
Dim sBaseDatos As String
Dim sTabla As String
Dim mOpcBD(2) As New "com.sun.star.beans.PropertyValue"

   'El nombre de nuestra base de datos
   sBaseDatos = "Directorio"
   'La tabla que nos interesa traer
   sTabla = "tblPaises"
   'La hoja activa
   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   'Las propiedades de la importación, primero el nombre de la base de datos
   mOpcBD(0).Name = "DatabaseName"
   mOpcBD(0).Value = sBaseDatos
   'El tipo de objeto fuente
   mOpcBD(1).Name = "SourceType"
   mOpcBD(1).Value = com.sun.star.sheet.DataImportMode.TABLE
   'El nombre del objeto fuente
   mOpcBD(2).Name = "SourceObject"
   mOpcBD(2).Value = sTabla
   'Importamos la tabla, a partir de la celda A10
   oHojaActiva.getCellRangeByName("A10").doImport( mOpcBD() )

End Sub
```

Ahora, importamos una consulta (*QUERY*) en vez de una tabla (*TABLE*).

```
Sub BasesDeDatos3()
Dim oHojaActiva As Object
Dim sBaseDatos As String
Dim sConsulta As String
Dim mOpcBD(2) As New "com.sun.star.beans.PropertyValue"

   sBaseDatos = "Directorio"
   'Ahora una consulta
   sConsulta = "qryCiudades"
   'La hoja activa
   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   mOpcBD(0).Name = "DatabaseName"
   mOpcBD(0).Value = sBaseDatos
   'El tipo de objeto fuente, una consulta
   mOpcBD(1).Name = "SourceType"
```

```
    mOpcBD(1).Value = com.sun.star.sheet.DataImportMode.QUERY
    'El nombre del objeto fuente
    mOpcBD(2).Name = "SourceObject"
    mOpcBD(2).Value = sConsulta
    'Importamos la tabla, a partir de la celda A1
    oHojaActiva.getCellRangeByName("A1").doImport( mOpcBD() )

End Sub
```

Tienes que tener la precaución de que el nombre del objeto corresponda a su tipo, si no, el método te devolverá un error:

Importamos una consulta diferente:

```
Sub BasesDeDatos4()
Dim oHojaActiva As Object
Dim sBaseDatos As String
Dim sConsulta As String
Dim mOpcBD(2) As New "com.sun.star.beans.PropertyValue"

    sBaseDatos = "Directorio"
    'Una consulta
    sConsulta = "qryContactos"
    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

    mOpcBD(0).Name = "DatabaseName"
    mOpcBD(0).Value = sBaseDatos
    mOpcBD(1).Name = "SourceType"
    mOpcBD(1).Value = com.sun.star.sheet.DataImportMode.QUERY
    mOpcBD(2).Name = "SourceObject"
    mOpcBD(2).Value = sConsulta
    oHojaActiva.getCellRangeByName("A1").doImport( mOpcBD() )

End Sub
```

Cambiemos de tipo de objeto fuente, el siguiente ejemplo, hace exactamente lo mismo que el primer ejemplo, es decir, nos importa la tabla “tblContactos”.

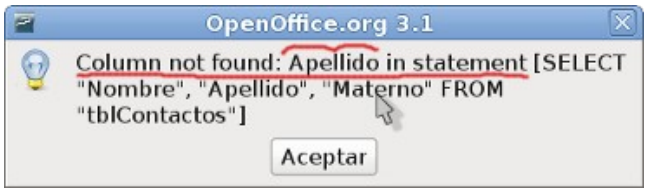
```
Sub BasesDeDatos5()
Dim oHojaActiva As Object
Dim sBaseDatos As String
Dim sSQL As String
Dim mOpcBD(2) As New "com.sun.star.beans.PropertyValue"

    sBaseDatos = "Directorio"
    'Establecemos la consulta SQL
    sSQL = "SELECT * FROM tblContactos"
    oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

    mOpcBD(0).Name = "DatabaseName"
    mOpcBD(0).Value = sBaseDatos
    mOpcBD(1).Name = "SourceType"
    'Observa como hemos cambiado el tipo de fuente a SQL
    mOpcBD(1).Value = com.sun.star.sheet.DataImportMode.SQL
    mOpcBD(2).Name = "SourceObject"
```

```
    mOpcBD(2).Value = sSQL
    oHojaActiva.getCellRangeByName("A1").doImport( mOpcBD() )

End Sub
```

Entonces, ¿cual es la diferencia?, la primera, es obvio, es que estamos usando una instrucción SQL para importar los datos, pero... ¿que es SQL?, la respuesta sencilla; SQL es un lenguaje estructurado de consultas, nos permite recuperar “casi” cualquier dato de cualquier base de datos con soporte para SQL, la respuesta compleja nos podría llevar una libro completo. En los siguiente ejemplos, me limitaré a darte muestras de instrucciones SQL que sean lo suficientemente ilustrativas del poder y versatilidad de este lenguaje. SQL es un estándar, así que, en “teoría”, cualquier documentación de el “debería” servirte. Cuando importamos tablas o consultas, estas, se importan con todo su contenido, con SQL, podemos limitarnos exclusivamente a los datos que necesitamos, en el siguiente ejemplo, solo importamos tres (*Nombre, Paterno y Materno*) campos de la tabla (*tblContactos*).

```
Sub BasesDeDatos6()
Dim oHojaActiva As Object
Dim sBaseDatos As String
Dim sSQL As String
Dim mOpcBD(2) As New "com.sun.star.beans.PropertyValue"

   sBaseDatos = "Directorio"
   'Establecemos la consulta SQL
   sSQL = "SELECT Nombre, Paterno, Materno FROM tblContactos"
   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   mOpcBD(0).Name = "DatabaseName"
   mOpcBD(0).Value = sBaseDatos
   mOpcBD(1).Name = "SourceType"
   mOpcBD(1).Value = com.sun.star.sheet.DataImportMode.SQL
   mOpcBD(2).Name = "SourceObject"
   mOpcBD(2).Value = sSQL
   oHojaActiva.getCellRangeByName("A1").doImport( mOpcBD() )

End Sub
```

Observa como en el primer ejemplo de instrucción SQL, le indicamos que queremos todo (*) con el asterisco, en este segundo ejemplo, le indicamos explícitamente que campos queremos importar, de nuevo, ten mucho cuidado con los nombres de tus campos, si el campo no existe, el método te devolverá un error.



Nota como claramente nos esta diciendo que el campo no existe, recuerda que en los nombres se distingue entre mayúsculas y minúsculas.

Las instrucciones SQL, tiene unos modificadores que se llaman “clausulas”, estas no permiten complementar, limitar, filtrar y ordenar entre otras acciones a los datos importados. En el siguiente ejemplo, ordenamos (*ORDER BY*) los datos importados.

```
Sub BasesDeDatos7()
Dim oHojaActiva As Object
Dim sBaseDatos As String
```

```
Dim sSQL As String
Dim mOpcBD(2) As New "com.sun.star.beans.PropertyValue"

   sBaseDatos = "Directorio"
   'Establecemos la consulta SQL
   sSQL = "SELECT Nombre, Paterno, Materno FROM tblContactos ORDER BY Paterno"
   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   mOpcBD(0).Name = "DatabaseName"
   mOpcBD(0).Value = sBaseDatos
   mOpcBD(1).Name = "SourceType"
   mOpcBD(1).Value = com.sun.star.sheet.DataImportMode.SQL
   mOpcBD(2).Name = "SourceObject"
   mOpcBD(2).Value = sSQL
   oHojaActiva.getCellRangeByName("A1").doImport( mOpcBD() )

End Sub
```

El nombre de los campos, casi siempre se establece de forma que sea fácil de recordar y de manipular por el programador, pero muchas veces este nombre no le dice mucho al usuario, para estos casos, tenemos una clausula para cambiar el nombre original del campo, por uno de más fácil lectura para el usuario, veamos como.

```
Sub BasesDeDatos8()
Dim oHojaActiva As Object
Dim sBaseDatos As String
Dim sSQL As String
Dim mOpcBD(2) As New "com.sun.star.beans.PropertyValue"

   sBaseDatos = "Directorio"
   'Establecemos la consulta SQL
   sSQL = "SELECT Nombre, Paterno As ""Apellido Paterno"", Materno As ""Apellido Materno"", Cumple As
Cumpleaños FROM tblContactos ORDER BY Paterno"

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   mOpcBD(0).Name = "DatabaseName"
   mOpcBD(0).Value = sBaseDatos
   mOpcBD(1).Name = "SourceType"
   mOpcBD(1).Value = com.sun.star.sheet.DataImportMode.SQL
   mOpcBD(2).Name = "SourceObject"
   mOpcBD(2).Value = sSQL
   oHojaActiva.getCellRangeByName("A1").doImport( mOpcBD() )

End Sub
```

Nota el uso de las dobles comillas, esto es indispensable cuando el nombre usado (*AS*) contenga espacios como en nuestro ejemplo, y estas, deben ser exactamente las necesarias, ni una más ni una menos, ten cuidado con su uso. Si revisas la estructura de la tabla "*tblContactos*" notaras que el campo "*Cumple*" es de tipo fecha (*Date*), para hacer consultas sobre campos de fechas, tienes que usar una sintaxis muy singular, por ejemplo, para consultar una fecha especifica, usamos.

```
sSQL = "SELECT * FROM tblContactos WHERE Cumple={ D '1974-01-15' }"
```

Observa como todo el criterio de la fecha va encerrado entre llaves, nota el uso de la letra D (de Date) y las comillas en la fecha, además, el orden tiene que ser precisamente así año-mes-día, si estableces correctamente este criterio, no tendrás problemas con el manejo de fechas.

Cambiemos de base de datos, en el siguiente ejemplo, usamos la base de datos “Videoteca”, y mostramos como podemos ordenar de forma ascendente (*ASC*) por un campo y de forma descendente (*DESC*) por otro.

```
Sub BasesDeDatos9()
Dim oHojaActiva As Object
Dim sBaseDatos As String
Dim sSQL As String
Dim mOpcBD(2) As New "com.sun.star.beans.PropertyValue"

   sBaseDatos = "Videoteca"
   'Establecemos la consulta SQL
   sSQL = "SELECT * FROM tblVideo ORDER BY Genero ASC, Director DESC"

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   mOpcBD(0).Name = "DatabaseName"
   mOpcBD(0).Value = sBaseDatos
   mOpcBD(1).Name = "SourceType"
   mOpcBD(1).Value = com.sun.star.sheet.DataImportMode.SQL
   mOpcBD(2).Name = "SourceObject"
   mOpcBD(2).Value = sSQL
   oHojaActiva.getCellRangeByName("A1").doImport( mOpcBD() )

End Sub
```

Como en los ejemplos siguientes solo cambia la instrucción SQL, solo te mostraré esta, mientras no se indique lo contrario, seguiremos usando la base de datos “Videoteca”.

En el siguiente ejemplo, filtramos los datos importados por el campo *Pais*.

```
sSQL = "SELECT * FROM tblVideo WHERE Pais='México'"
```

Siempre que quieras establecer un criterio de filtrado usa WHERE en donde estableces el *CAMPO=CRITERIO* deseado y nota también, como hacemos uso de comillas simples para el parámetro del criterio, esto, es importante.

La siguiente instrucción, nos importa todos los títulos que comiencen por la letra “V”, observa el uso de una nueva clausula *LIKE* sin signo igual y con el símbolo de porcentaje después de la letra deseada sin olvidar el uso de las comillas simples.

```
   sSQL = "SELECT * FROM tblVideo WHERE Titulo LIKE 'V%'"
```

Ahora, importamos todas las películas que “contengan” la palabra “anillos”.

```
   sSQL = "SELECT * FROM tblVideo WHERE Titulo LIKE '%anillos%'"
```

Con la siguiente instrucción, importamos todas las películas que duren entre 60 y 90 minutos ordenados por duración, tenemos una nueva clausula para importar datos entre dos valores (*BETWEEN*) junto con el operador “y” (*AND*).

```
   sSQL = "SELECT * FROM tblVideo WHERE Duracion BETWEEN 60 AND 90 ORDER BY Duracion"
```

Ahora, importamos las películas que sean de cualquiera (*OR*) de los siguientes años; 1974 o 1986 o 2000, observa que volvemos a hace uso de la forma *CAMPO=CRITERIO.*

```
sSQL = "SELECT * FROM tblVideo WHERE Año=1974 OR Año=1986 OR Año=2000"
```

Nuestro siguiente ejemplo, hace exactamente lo mismo que el anterior, solo que ahora hacemos uso de la clausula *IN* para establecer los años deseados.

```
sSQL = "SELECT * FROM tblVideo WHERE Año IN(1974,1986,2000)"
```

El siguiente ejemplo es muy interesante, nos permite obtener un listado con registros únicos (*DISTINCT*), lo que nos da la posibilidad de importar, por ejemplo, todos los directores que tenemos en nuestra videoteca, además, es una buena alternativa para encontrar campos mal capturados, por ejemplo, si algún campo tiene un solo espacio de más, con esta clausula te darás cuenta cuales de ellos pueden tener este caso, ya que los considerará como dos registros diferentes.

```
sSQL = "SELECT DISTINCT Director FROM tblVideo"
```

El lenguaje SQL, también nos permite hacer algunas operaciones sobre los campos, por ejemplo, en vez de regresar los distintos países que tenemos, solo los contamos (*COUNT*), nota que primero obtenemos los países *(DISTINCT Pais)*, y después los contamos (*COUNT*), además, le establecemos un nombre a este resultado (*AS ""Total Países""*).

```
sSQL = "SELECT COUNT(DISTINCT Pais) AS ""Total Países"" FROM tblVideo"
```

Podemos obtener, el total de minutos de nuestra videoteca, sumando (*SUM*) los minutos de duración de todas las películas.

```
sSQL = "SELECT SUM(Duracion) AS ""Total Minutos"" FROM tblVideo"
```

U obtener el promedio (*AVG*) de duración por película.

```
sSQL = "SELECT AVG(Duracion) AS ""Promedio en Minutos"" FROM tblVideo"
```

O la duración máxima.

```
sSQL = "SELECT MAX(Duracion) AS ""Duración Máxima"" FROM tblVideo"
```

O la mínima.

```
sSQL = "SELECT MIN(Duracion) AS ""Duración Mínima"" FROM tblVideo"
```

En el siguiente ejemplo, usamos una nueva clausula que nos sirve para agrupar (*GROUP BY*) el resultado por algún campo, para obtener al suma de minutos por genero, usamos la siguiente instrucción SQL.

```
sSQL = "SELECT Genero, SUM(Duracion) AS ""Duración por Genero"" FROM tblVideo GROUP BY Genero"
```

Observa como en los ejemplos para obtener la película de máxima (*MAX*) y mínima (*MIN*) duración, efectivamente obtenemos el valor, pero no sabes que título es este, para saberlo, vamos a realizar una “subconsulta”, que no es otra cosa que consultas anidadas, en donde, el

resultado de la consulta interior, es el criterio de filtro para la exterior, con en el siguiente ejemplo, ahora si, obtenemos todos los datos de la película de máxima duración en nuestra videoteca.

```
    sSQL = "SELECT * FROM tblVideo WHERE Duracion=(SELECT MAX(Duracion) FROM tblVideo)"
```

No te confundas, primero, obtenemos la máxima duración (*SELECT MAX(Duracion) FROM tblVideo*), solo el valor, después, este valor, lo establecemos como criterio (*WHERE*) de la consulta exterior, donde si devolvemos todos los campos, es muy importante el uso de los paréntesis para la consulta interior.

Los ejemplos de instrucciones SQL vistas en todos los ejemplos anteriores, son solo una muestra ínfima de su poder y versatilidad, pero creo, suficientes para tengas mucho para practicar y probar. El método *doImport*, no es el único método para consultar datos en bases de datos pero si el más sencillo.

6.8.2 Insertando nuevos datos

Las instrucciones SQL no solo sirven para importar datos, también nos sirven para insertar nuevos datos en las tablas de nuestra base de datos. En el siguiente ejemplo, importamos los países dados de alta en nuestra tabla “tblPaises” de nuestra base de datos “Directorio”.

```
Sub BasesDeDatos10()
Dim oHojaActiva As Object
Dim sBaseDatos As String
Dim sTabla As String
Dim mOpcBD(2) As New "com.sun.star.beans.PropertyValue"

   sBaseDatos = "Directorio"
   sTabla = "tblPaises"
   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()

   mOpcBD(0).Name = "DatabaseName"
   mOpcBD(0).Value = sBaseDatos
   mOpcBD(1).Name = "SourceType"
   mOpcBD(1).Value = com.sun.star.sheet.DataImportMode.TABLE
   mOpcBD(2).Name = "SourceObject"
   mOpcBD(2).Value = sTabla

   oHojaActiva.getCellRangeByName("A1").doImport( mOpcBD() )

End Sub
```

La importación anterior, tiene que haberte devuelto algo muy similar a.

	A	B	C
1	Id	Pais	LDPais
2	0	México	52
3	1	Argentina	54
4	2	España	34
5	3	Colombia	

Pero el método para importar (*doImport*), hace exactamente solo eso, importar, por lo que para insertar nuevos datos usaremos una técnica diferente.

Si revisas la estructura de la tabla “tblPaises”, observaras que esta formada por tres campos; “Id”, “Pais” y “LDPais”, el primero es un campo tipo entero, es la clave primaria y su inserción es automática, el segundo es un campo tipo texto y es requerido, es decir, no puede faltar, y el tercero, es un entero que puede o no puede estar, esto, la estructura de la tabla, el nombre de los campos, el tipo de campo y si es requerido o no, son datos indispensables para insertar nuevos datos, ya que, de no cumplir con algún requerimiento especifico o no coincidir el tipo de dato que se desea insertar con el tipo del campo de la tabla, lo más probable es que la instrucción no tenga éxito y te devuelva un error. Antes de empezar con los ejemplos propios de inserción de datos, veamos la nueva forma de acceder a nuestra base de datos, el siguiente código, te mostrará todas las bases de datos “registradas” en OpenOffice.org.

```
Sub BasesDeDatos11()
Dim oDBC As Object
Dim mNombresBD() As String
Dim co1 As Integer

   'Creamos el servicio para acceder y manipular las bases de datos
   oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
   'Obtenemos los nombres de las bases de datos registradas
   mNombresBD = oDBC.getElementNames()
   'Mostramos el nombre de cada una
   For co1 = LBound( mNombresBD ) To UBound ( mNombresBD )
         MsgBox mNombresBD(co1)
   Next

End Sub
```

Para acceder a la base de datos de nuestro interés, usamos.

```
Sub BasesDeDatos12()
Dim oDBC As Object
Dim oBD As Object
Dim sBaseDatos As String

   'El nombre de la base de datos
   sBaseDatos = "Directorio"
   'Creamos el servicio para acceder y manipular las bases de datos
   oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
   'Nos aseguramos de que exista la base de datos
   If oDBC.hasByName( sBaseDatos ) Then
         'Si existe, accedemos por el nombre
         oBD = oDBC.getByName( sBaseDatos )
         'Mostramos la ruta de la base de datos
         MsgBox ConvertFromURL( oBD.DatabaseDocument.URL )
   End If

End Sub
```

El siguiente ejemplo, nos ilustra una nueva forma de hacer una consulta (*executeQuery*), diferente al método doImport, este nuevo método, tienen muchas variantes y posibilidades, pues nos da acceso al resultado de la consulta para manipular cada registro y cada campo según nuestras necesidades.

```
Sub BasesDeDatos13()
Dim oDBC As Object
Dim oBD As Object
Dim oConexion As Object
Dim oDeclaracion As Object
Dim oResultado As Object
Dim sBaseDatos As String
```

```
Dim sSQL As String

   'El nombre de la base de datos
   sBaseDatos = "Directorio"
   sSQL = "SELECT * FROM tblPaises"
   'Creamos el servicio para acceder y manipular las bases de datos
   oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
   'Nos aseguramos de que exista la base de datos
   If oDBC.hasByName( sBaseDatos ) Then
         'Si existe, accedemos por el nombre
         oBD = oDBC.getByName( sBaseDatos )
         'Creamos una conexion a la base de datos
         oConexion = oBD.getConnection("","")
         'Creamos un objeto para las instrucciones SQL
         oDeclaracion = oConexion.createStatement()
         'Ejecutamos la consulta
         oResultado = oDeclaracion.executeQuery( sSQL)
         'Si hay resultados
         If Not IsNull( oResultado ) Then
               Do While oResultado.next
                     'Mostramos el contenido del campo
                     MsgBox oResultado.getString( 2 )
               Loop
         End If
   End If

End Sub
```

Ahora si, veamos como insertar un nuevo registro en nuestra base de datos.

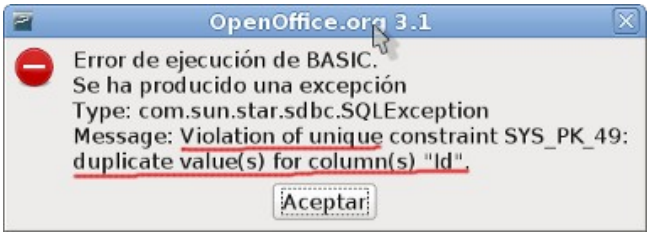
```
Sub BasesDeDatos14()
Dim oDBC As Object
Dim oBD As Object
Dim oConexion As Object
Dim oDeclaracion As Object
Dim sBaseDatos As String
Dim sSQL As String

   'El nombre de la base de datos
   sBaseDatos = "Directorio"
   sSQL = "INSERT INTO ""tblPaises"" (""Pais"") VALUES ('Honduras')"
   'Creamos el servicio para acceder y manipular las bases de datos
   oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
   'Nos aseguramos de que exista la base de datos
   If oDBC.hasByName( sBaseDatos ) Then
         'Si existe, accedemos por el nombre
         oBD = oDBC.getByName( sBaseDatos )
         'Creamos una conexion a la base de datos
         oConexion = oBD.getConnection("","")
         'Creamos un objeto para las instrucciones SQL
         oDeclaracion = oConexion.createStatement()
         'Ejecutamos la inserción de datos
         oDeclaracion.executeUpdate( sSQL)
   End If

End Sub
```

Es importante que notes que estamos usando un nuevo método (*executeUpdate*) para ejecutar la instrucción SQL. Para insertar registros, usamos una nueva instrucción SQL (*INSERT INTO*), seguida de la tabla donde insertaremos los nuevos datos, entre paréntesis, los nombres de los campos, después la clausula para indicar los valores (*VALUES*) y por ultimo, los valores a insertar entre paréntesis. Nota que en la tabla y los campos, usamos doble comilla y en los valores a insertar, si estos son texto, usamos comillas simples. Recordemos que la tabla

"*tblPaises*" esta conformada por tres campos, el primero se llama "Id" de tipo entero, que además es clave primaria y de inserción automática, por ello omitimos, tanto el campo como el valor, por que este, será automáticamente insertado, no obstante, es posible insertarlo si lo deseas, como en el siguiente ejemplo.

```
    sSQL = "INSERT INTO ""tblPaises"" (""Id"", ""Pais"") VALUES (11,'Honduras')"
```

Observa como indicamos el nombre del campo (Id) y su valor (11), pero, al ser este campo una clave primaria, estas "no admiten duplicados", por lo que si el valor indicado ya existe, te dará un error como el siguiente, donde claramente nos indica que estamos violando la unicidad del índice, intentando insertar valores duplicados para la columna (campo) Id.



Aunque este error en tiempo de ejecución es interceptable y manipulable, como lo vimos en el capitulo 4.10. Control de errores, si tienes un campo que es clave primaria y de inserción automática como el campo "Id" de nuestra tabla, la recomendación es que permitas al motor de bases de datos, se encargue de insertar el valor siguiente como en nuestro ejemplo, que tienes que ejecutar varias veces para que notes que efectivamente, inserta el valor automáticamente. Después tenemos el campo "Pais", si ejecutaste la macro varias veces y haces la consulta de la tabla o la abres desde Base, notaras que ahora, tenemos varias veces el país recién insertado, ¿es esto correcto?, claro que no, la idea de una base de datos es tener la menor cantidad de información repetida, de hecho, "no debería" haber información repetida en una base de datos. Para evitar la duplicidad de datos en otro campo que no sea la clave primaria, tienes dos posibles soluciones; una, puedes crear un *índice* en este campo, en el cual se le indica que solo contenga datos únicos con lo cual, si se intentan agregar datos duplicados, obtendrás un error interceptable en tiempo de ejecución, para ello, tienes que modificar la tabla desde Base y agregar el índice manualmente (o por código) lo cual aprenderemos más adelante, por ahora, usaremos el segundo método, que consiste en consultar si el país ya existe o no, y actuar en consecuencia, una primera aproximación a esta solución seria.

```
Sub BasesDeDatos15()
Dim oDBC As Object
Dim oBD As Object
Dim oConexion As Object
Dim oDeclaracion As Object
Dim oResultado As Object
Dim sBaseDatos As String
Dim sSQL As String
Dim sPais As String

   sPais = Trim( InputBox( "Introduce el nombre del nuevo país" ) )
   If sPais <> "" Then
         sBaseDatos = "Directorio"
         sSQL = "SELECT Pais FROM ""tblPaises"" WHERE Pais='" & sPais & "'"
         oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
         If oDBC.hasByName( sBaseDatos ) Then
               oBD = oDBC.getByName( sBaseDatos )
               oConexion = oBD.getConnection("","")
               oDeclaracion = oConexion.createStatement()
               'Verificamos que el país no exista
               oResultado = oDeclaracion.executeQuery( sSQL)
               oResultado.next()
```

```
                If oResultado.getRow = 0 Then
                      'Si no existe lo insertamos
                      sSQL = "INSERT INTO ""tblPaises"" (""Pais"") VALUES ('" & sPais & "')"
                      oDeclaracion.executeUpdate( sSQL)
                      MsgBox "El país: " & sPais & " se inserto correctamente en la base de datos"
                Else
                      oResultado.close()
                      MsgBox "El país: " & sPais & " ya existe en la base de datos"
                End If
                'Cerramos las conexiones
                oDeclaracion.close()
                oConexion.close()
                'Liberamos la memoria
                oResultado = Nothing
                oDeclaracion = Nothing
                oConexion = Nothing
          End If
    Else
          MsgBox "El campo no puede estar vacío"
    End If

  End Sub
```

Aun con esta validación, es posible que haya todavía datos duplicados cuando intentemos introducir variantes como “México”, “MÉxico”, “MEXico” o “MÉXICO”, esto es por que el campo distingue mayúsculas de minúsculas, de nuevo, tienes dos alternativas; la primero es obvia, haces la consulta y comparas (si hay resultado), el valor devuelto con el valor nuevo todo en mayúsculas o minúsculas según prefieras, la otra, es modificar la estructura de la tabla para que ignore esta distinción, salvo en contadas excepciones, esta segunda opción es más rápida, practica y segura y como siempre, tienes la ultima palabra. Modifica la macro anterior para que no distinga mayúsculas de minúsculas y permite al usuario seguir introduciendo países hasta que quiera presionando el botón *Cancelar* del cuadro de dialogo.

El ultimo campo de nuestra tabla es: “”LDPais”, que es un entero, pero el campo no es requerido, por lo que puedes omitirlo como lo hemos hecho hasta ahora, o puedes insertarlo, como en el siguiente ejemplo:

```
    sSQL = "INSERT INTO ""tblPaises"" (""Pais"", ""LDPais"") VALUES ('Venezuela',45)"
```

Observa que al ser un número, no lleva comillas, no así el campo, “todos” llevan sus comillas dobles. En “teoría”, una base de datos ideal, no “debería” tener campos vacíos, por lo que, aunque no sea obligatorio, procura dejar vacíos la menor cantidad de campos posible, en algunas ocasiones, si un campo esta casi siempre vacío, tal vez podrías plantearte la posibilidad de eliminarlo de la tabla, aunque no es muy recomendable estar eliminando o insertando campos, es mucho mejor y más eficiente, “diseñar” tus tablas, anticipando lo mejor posible, su contenido y estructura y evitar en lo posible cambios constantes durante la implementación de tu código, te evitaras algunos dolores de cabeza.

Para terminar este tema, veamos un ejemplo donde se inserta una fecha, la cual, debe ir entre comillas simples y muy importante, en el orden año-mes-día, si no, te dará un error en tiempo de ejecución.

```
  Sub BasesDeDatos16()
  Dim oDBC As Object
  Dim oBD As Object
  Dim oConexion As Object
  Dim oDeclaracion As Object
  Dim sBaseDatos As String
  Dim sSQL As String
```

```
    sBaseDatos = "Directorio"
    sSQL = "INSERT INTO ""tblContactos"" (""Nombre"",""Paterno"",""Materno"",""Cumple"",""IdCiudad"")
VALUES ('Juan','Gomez','Perez','1981-01-31',2)"
    oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
    oBD = oDBC.getByName( sBaseDatos )
    oConexion = oBD.getConnection("","")
    oDeclaracion = oConexion.createStatement()
    oDeclaracion.executeUpdate( sSQL)

    oDeclaracion.close()
    oConexion.close()
    oDeclaracion = Nothing
    oConexion = Nothing

End Sub
```

Las recomendaciones generales al insertar datos son: ten cuidado con el tipo de datos del campo destino, cuida de que no falte ningún campo requerido y cuida los campos que tienen alguna “relación” con un campo de otra tabla.

6.8.3 Actualizando datos

La actualización de datos en una tabla, es una acción recurrente en la manipulación de bases de datos. Veamos como hacerlo, el siguiente ejemplo, actualiza el campo “LDPais” de la tabla “tblPaises” pero solo el registro que cumple la condición.

```
Sub BasesDeDatos17()
Dim oDBC As Object
Dim oBD As Object
Dim oConexion As Object
Dim oDeclaracion As Object
Dim sBaseDatos As String
Dim sSQL As String

    sBaseDatos = "Directorio"
    'Construimos la instrucción de actualización
    sSQL = "UPDATE ""tblPaises"" SET ""LDPais""=57 WHERE ""Pais""='Colombia'"
    oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
    oBD = oDBC.getByName( sBaseDatos )
    oConexion = oBD.getConnection("","")
    oDeclaracion = oConexion.createStatement()
    oDeclaracion.executeUpdate( sSQL)

    oDeclaracion.close()
    oConexion.close()
    oDeclaracion = Nothing
    oConexion = Nothing

End Sub
```

Para actualizar, usamos una nueva instrucción SQL (*UPDATE*), después establecemos (*SET*), el campo a actualizar y el nuevo valor, si hay más campo se separan por comas, por último, la condición (*WHERE*) para actualizar solo los registros deseados, cuidado, si no estableces un criterio, te actualizará “todos” los registros de tu tabla. Si el criterio no devuelve ningún resultado, no veras ningún mensaje, simplemente no hará ningún cambio en la tabla.

En el siguiente ejemplo, mostramos el país y la clave lada de cada uno con la posibilidad de cambiarlo.

```
Sub BasesDeDatos18()
Dim oDBC As Object
Dim oBD As Object
Dim oConexion As Object
Dim oDeclaracion As Object
Dim oDeclaracion1 As Object
Dim oResultado AS Object
Dim sBaseDatos As String
Dim sSQL As String
Dim sInfo As String
Dim sClave As String

   sBaseDatos = "Directorio"
   sSQL = "SELECT Pais, LDPais FROM tblPaises"
   oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
   oBD = oDBC.getByName( sBaseDatos )
   oConexion = oBD.getConnection("","")
   oDeclaracion = oConexion.createStatement()
   oDeclaracion1 = oConexion.createStatement()
   oResultado = oDeclaracion.executeQuery( sSQL)

   Do While oResultado.Next
         sInfo = "El pais: " & oResultado.getString(1) & " tiene como clave Lada: " & _
                     oResultado.getString(2) & Chr(13) & Chr(13) & "Introduce la nueva clave lada"
         sClave = Val(Trim(InputBox(sInfo)))
         If sClave > 0 Then
               sSQL = "UPDATE ""tblPaises"" SET ""LDPais""=" & sClave & " WHERE ""Pais""='" &
oResultado.getString(1) & "'"
               oDeclaracion1.executeUpdate( sSQL)
         End If
   Loop

   oResultado.close()
   oDeclaracion.close()
   oDeclaracion1.close()
   oConexion.close()
   oResultado = Nothing
   oDeclaracion = Nothing
   oDeclaracion1 = Nothing
   oConexion = Nothing

End Sub
```

Observa que creamos dos declaraciones (*createStatement*), una es para la consulta de selección y la otra para la actualización, no puedes usar la misma, pues se ven afectadas una por la otra. En nuestro siguiente ejemplo, actualizamos la fecha del contacto con Id = 0.

```
Sub BasesDeDatos19()
Dim oDBC As Object
Dim oBD As Object
Dim oConexion As Object
Dim oDeclaracion As Object
Dim sBaseDatos As String
Dim sSQL As String

   sBaseDatos = "Directorio"
   'Construimos la instrucción de actualización
   sSQL = "UPDATE ""tblContactos"" SET ""Cumple""='1980-02-28' WHERE ""Id""=0"
   oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
   oBD = oDBC.getByName( sBaseDatos )
   oConexion = oBD.getConnection("","")
   oDeclaracion = oConexion.createStatement()
   oDeclaracion.executeUpdate( sSQL)
```

```
    oDeclaracion.close()
    oConexion.close()
    oDeclaracion = Nothing
    oConexion = Nothing

End Sub
```

Al igual que en la inserción, la fecha debe cumplir el orden año-mes-día y estar entre comillas simples. Cuida respetar el tipo de dato del campo a actualizar.

6.8.4 Borrando datos

Borrar datos es sumamente simple, por ello, ten mucho cuidado cuando uses esta instrucción, el siguiente ejemplo borra un país de la tabla.

```
Sub BasesDeDatos20()
Dim oDBC As Object
Dim oBD As Object
Dim oConexion As Object
Dim oDeclaracion As Object
Dim sBaseDatos As String
Dim sSQL As String

    sBaseDatos = "Directorio"
    'Construimos la instrucción de borrado
    sSQL = "DELETE FROM ""tblPaises"" WHERE ""LDPais""=57"
    oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
    oBD = oDBC.getByName( sBaseDatos )
    oConexion = oBD.getConnection("","")
    oDeclaracion = oConexion.createStatement()
    oDeclaracion.executeUpdate( sSQL)

    oDeclaracion.close()
    oConexion.close()
    oDeclaracion = Nothing
    oConexion = Nothing

End Sub
```

¡Cuidado!, si no estableces la condición, “**borraras toda el contenido de la tabla**”, es recomendable, en cualquier operación de borrado, así sea un campo, una tabla, un archivo o lo que sea, preguntarle al usuario si esta seguro de realizar la acción e informarle que esta no se puede deshacer. Las instrucciones de borrado, pueden afectar a tablas que estén “relacionadas”, tanto la actualización como el borrado, están ligados con un concepto en bases de datos que se llama “integridad referencial”, tema que sale de los propósitos de este libro, pero que si quieres profundizar en ello, tienes que conocerlo, estudiarlo, comprenderlo y aplicarlo en tus bases de datos, por ahora, con que tengas el cuidado de no dejar “huérfanos” a tus datos es más que suficiente. En el siguiente ejemplo, borramos todos los contactos que se llamen 'Juan'.

```
Sub BasesDeDatos21()
Dim oDBC As Object
Dim oBD As Object
Dim oConexion As Object
Dim oDeclaracion As Object
Dim sBaseDatos As String
Dim sSQL As String
```

```
    sBaseDatos = "Directorio"
    'Construimos la instrucción de borrado
    sSQL = "DELETE FROM ""tblContactos"" WHERE ""Nombre""='Juan'"
    oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
    oBD = oDBC.getByName( sBaseDatos )
    oConexion = oBD.getConnection("","")
    oDeclaracion = oConexion.createStatement()
    oDeclaracion.executeUpdate( sSQL)

    oDeclaracion.close()
    oConexion.close()
    oDeclaracion = Nothing
    oConexion = Nothing

End Sub
```

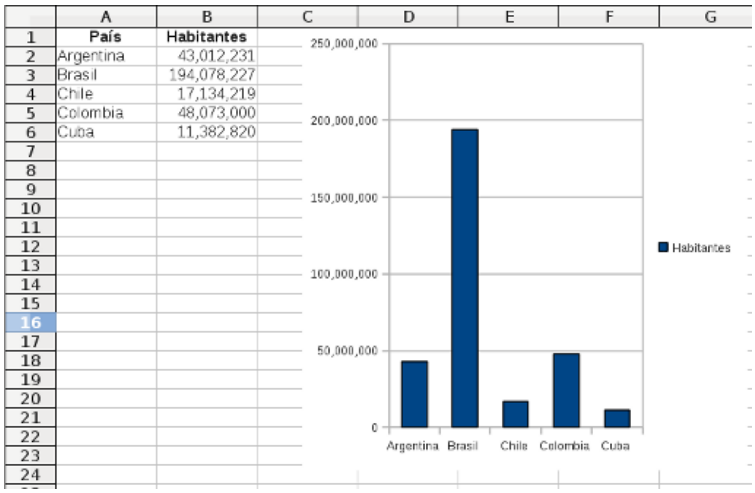
Para consultar, insertar, actualizar o borrar datos, puedes tomar los valores de las celdas de Calc, con las mismas técnicas vista a lo largo de estos apuntes, solo ten cuidado con el correcto uso de las comillas, lo mejor, es crearte una función que agregue las comillas necesarias, esto, te aseguro, te evitará muchos dolores de cabeza.

Los temas vistos en este capitulo, son solo una pequeña parte de ese enorme tema que son las bases de datos, no obstante, tienes ya, herramientas suficientes para obtener mucho provecho de tus datos y tus conocimientos.

6.9 Graficando datos

Dicen que una imagen vale más que mil palabras, la verdad es que yo prefiero las mil palabras, no obstante, daré por cierta la aseveración anterior y veremos como darle “imagen” a nuestros datos, que de eso se trata cuando hacemos gráficos.

Vamos a crear el siguiente gráfico de la población de cinco países de América Latina.



El cual creamos con la siguiente macro.

```
Sub Graficando1()
```

```
Dim oHojaActiva As Object
Dim oGraficos As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oDir As New com.sun.star.table.CellRangeAddress

   'Acceso a la hoja activa
   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'El nombre de nuestro gráfico
   sNombre = "Grafico01"
   'El tamaño y la posición del nuevo gráfico, todas las medidas
   'en centésimas de milímetro
   With oRec
         .X = 5500              'Distancia desde la izquierda de la hoja
         .Y = 0                 'Distancia desde la parte superior
         .Width = 10000         'El ancho del gráfico
         .Height = 10000        'El alto del gráfico
   End With
   'La dirección del rango de datos para el gráfico
   With oDir
         .Sheet = oHojaActiva.getRangeAddress.Sheet
         .StartColumn = 0
         .EndColumn = 1
         .StartRow = 0
         .EndRow = 5
   End With
   'Es una matriz de rangos, pues se pueden establecer más de uno
   mRangos(0) = oDir
   'Accedemos al conjunto de todos los gráficos de la hoja
   oGraficos = oHojaActiva.getCharts()
   'Verificamos que no exista el nombre
   If oGraficos.hasByName( sNombre ) Then
         MsgBox "Ya existe este nombre de gráfico, escoge otro"
   Else
         'Si no existe lo agregamos
         oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
   End If

End Sub
```

El método para agregar un nuevo gráfico es sencillo (*addNewByName*) y consta de cinco argumentos.

```
   addNewByName(Nombre, Rectangulo, Rangos, EncabezadoColumna, EncabezadoFila)
```

1. Nombre: es el nombre del objeto a nivel código, este nombre es diferente del que puede establecer el usuario en la interfaz de la aplicación, es indispensable que no exista otro gráfico con este valor.
2. Rectángulo: es una estructura *com.sun.star.awt.Rectangle* que permite establecer el tamaño y la posición del gráfico, las unidades están especificadas en centésimas de milímetro, 1000 = 1 cm
3. Rangos: es una matriz de rangos, es decir, de estructuras *com.sun.star.table.CellRangeAddress* que guardan la dirección del rango para los datos de origen del gráfico.
4. EncabezadoColumna: valor booleano, establecido en verdadero (True) sirve para indicar si la fila superior se usara como titulo de etiquetas para eje o leyenda.
5. EncabezadoFila: valor booleano, establecido en verdadero (True) sirve para indicar si se usará la columna de la izquierda como etiquetas de eje o leyenda.

Ya creado el gráfico, podemos personalizarlo completamente a nuestro gusto. En el siguiente ejemplo, creamos tres nuevos gráficos, de tres estilos diferentes: columnas, barras y circular, uno al lado de otro, teniendo como origen de datos, el mismo del ejemplo anterior.

```
Sub Graficando2()
Dim oHojaActiva As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oDir As New com.sun.star.table.CellRangeAddress

   'Acceso a la hoja activa
   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'El nombre de nuestro gráfico
   sNombre = "GraficoA"
   'El tamaño y la posición del nuevo gráfico, todas las medidas
   'en centésimas de milímetro
   With oRec
         .X = 0                        'Distancia desde la izquierda de la hoja
         .Y = 4000               'Distancia desde la parte superior
         .Width = 10000          'El ancho del gráfico
         .Height = 10000         'El alto del gráfico
   End With
   'La dirección del rango de datos para el gráfico
   With oDir
         .Sheet = oHojaActiva.getRangeAddress.Sheet
         .StartColumn = 0
         .EndColumn = 1
         .StartRow = 0
         .EndRow = 5
   End With
   'Es una matriz de rangos, pues se pueden establecer más de uno
   mRangos(0) = oDir
   'Accedemos al conjunto de todos los gráficos de la hoja
   oGraficos = oHojaActiva.getCharts()
   'Verificamos que no exista el nombre
   If oGraficos.hasByName( sNombre ) Then
         MsgBox "Ya existe este nombre de gráfico, escoge otro"
   Else
         'Si no existe lo agregamos, de forma predeterminada se crea
         'un gráfico de columnas
         oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
   End If

   'Cambiamos el nombre y la posición para un segundo gráfico
   sNombre = "GraficoB"
   With oRec
         .X = 10000
         .Y = 4000
   End With
   If oGraficos.hasByName( sNombre ) Then
         MsgBox "Ya existe este nombre de gráfico, escoge otro"
   Else
         oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
         'Accedemos al nuevo gráfico
         oGrafico = oGraficos.getBYName( sNombre ).getEmbeddedObject()
         'Establecemos que sean barras en vez de columnas
         oGrafico.getDiagram.Vertical = True
   End If

   'Volvemos a cambiar el nombre y la posición para un tercer gráfico
   sNombre = "GraficoC"
   With oRec
```

```
            .X = 20000
            .Y = 4000
    End With
    If oGraficos.hasByName( sNombre ) Then
          MsgBox "Ya existe este nombre de gráfico, escoge otro"
    Else
          oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
          oGrafico = oGraficos.getBYName( sNombre ).getEmbeddedObject()
          oGrafico.setDiagram( oGrafico.createInstance("com.sun.star.chart.PieDiagram"))
    End If

End Sub
```

Los gráficos de columnas y barras son el mismo tipo de gráfico, excepto por la orientación, uno es vertical y otro es horizontal, para el tercer gráfico, cambiamos completamente su tipo (*setDiagram*), lo establecemos circular, toma en cuenta que cada gráfico esta pensado para un determinado tipo de datos, los tipos de gráficos que puedes establecer son.

com.sun.star.chart	*Tipo*
com.sun.star.chart.BarDiagram	Barras (y columnas)
com.sun.star.chart.AreaDiagram	Áreas
com.sun.star.chart.LineDiagram	Líneas
com.sun.star.chart.PieDiagram	Circular
com.sun.star.chart.DonutDiagram	Dona
com.sun.star.chart.NetDiagram	Red
com.sun.star.chart.XYDiagram	Dispersión XY
com.sun.star.chart.StockDiagram	Stock
com.sun.star.chart.BubbleDiagram	Burbujas

Cada uno de estos tipos, tiene variantes que multiplican las posibilidades para graficar la información, por ejemplo, representaciones en 3D, agregar líneas y puntos, entre otras características que veremos más adelante.

La mayor parte de los elementos de un gráfico, son en realidad una forma (*Shape*), a la cual le puedes establecer sus propiedades, principalmente; el estilo de letra (tipo de fuente, color, negritas, tamaño, etc), el estilo de fondo (estilo, color, etc), y el estilo de borde (tipo de línea, color, ancho, etc), entre otras más, vamos a mostrar como modificar los principales elementos de un gráfico, los siguientes ejemplos manipulan el gráfico seleccionado, pero todas las propiedades que veremos, las puedes establecer perfectamente al crearlo de modo que tu gráfico este configurado completamente a tu gusto y necesidad a la primera. Vamos a hacer uso de las siguientes subrutinas para dar formato.

```
'Subrutina para formatear el texto de una forma, si no quieres cambiar alguna
'propiedad, solo pasa una cadena vacía en los textos y un 0 en los números
Sub FormatoTexto(Obj As Object, Texto As String, Fuente As String, Neg As Integer, Tam As Integer,
Color As Long)
Dim oTexto As Object
Dim mTexto(0)

   'El titulo del gráfico es algo especial
   If Obj.supportsService("com.sun.star.chart2.Title") Then
         'Requiere un objeto "Cadena Formateada" (FormattedString)
         oTexto = createUnoService("com.sun.star.comp.chart.FormattedString")
         'Y agregarse a una matriz
         mTexto(0) = oTexto
         'Establecer el texto
```

```
            Obj.setText( mTexto )
      Else
            oTexto = Obj
      End If
      'Cambiamos el formato del texto
      With oTexto
            If Texto <> "" Then .String = Texto
            If Fuente <> "" Then .CharFontName = Fuente
            If Neg > 0 Then .CharWeight = Neg
            If Tam > 0 Then .CharHeight = Tam
            If Color > 0 Then .CharColor = Color
      End With

End Sub


'Subrutina para cambiar el fondo de una forma,
'Estilo 0 = Ninguno, 1 = Color, 2 = Gradiente, 3 = Trama, 4 = Mapa de bits
'Color puede ser un número o una cadena con el nombre del gradiente, trama o mapa de bits
Sub FormatoFondo(Obj As Object, Estilo As Integer, Color)
      With Obj
            .FillBackground = True
            .FillStyle = Estilo
            Select Case Estilo
                  Case 1
                        If Color > 0 Then .FillColor = Color
                  Case 2
                        .FillGradientName = Color
                  Case 3
                        .FillHachName = Color
                  Case 4
                        .FillBitmapName = Color
            End Select
      End With

End Sub


'Subrutina para cambiar la línea de una forma, si no quieres línea,
'establece el estilo en 0
Sub FormatoLinea(Obj As Object, Estilo As Integer, Color As Long, Ancho As Integer)
      With Obj
            .LineStyle = Estilo
            If Color > 0 Then .LineColor = Color
            If Ancho > 0 Then .LineWidth = Ancho
      End With

End Sub


'Función para devolver el gráfico activo
'devuelve NULL si no es un gráfico
Function getGrafico() As Object
Dim oHojaActiva As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim oSel As Object
Dim sNombre As String
Dim sInfo As String

      oSel = ThisComponent.getCurrentSelection()
      If oSel.getImplementationName = "com.sun.star.drawing.SvxShapeCollection" Then
            oSel = oSel.getByIndex(0)
            If oSel.supportsService("com.sun.star.drawing.OLE2Shape") Then
                  sNombre = oSel.PersistName
                  oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
```

```
            oGraficos = oHojaActiva.getCharts()
            If oGraficos.hasByName( sNombre ) Then
                  getGrafico = oGraficos.getByName( sNombre ).getEmbeddedObject()
            End If
      End If
   End If

End Function
```

En el siguiente ejemplo, agregamos y damos formato al titulo y al subtitulo del gráfico, observa que si no existe el titulo lo creamos, el subtitulo siempre existe, solo hay que determinar si lo mostramos o no, como se indica en los comentarios.

```
Sub Graficando3()
Dim oGrafico As Object
Dim oTitulo As Object
Dim oSubTitulo As Object

   'Accedemos al gráfico seleccionado
   oGrafico = getGrafico()
   If Not IsNull( oGrafico ) Then

         'Si hay un gráfico, seleccionamos el título
         oTitulo = oGrafico.getTitleObject()
         If IsNull(oTitulo) Then
               'Si no existe el título lo creamos
               oTitulo = createUnoService("com.sun.star.chart2.Title")
         End If
         'Le damos formato al texto del título
         Call FormatoTexto( oTitulo, "Habitantes Por País", "Liberation Serif", 150, 15, RGB(255,0,0)
)
         'Le damos formato al fondo del titulo
         Call FormatoFondo( oTitulo, 1, RGB(220,220,220) )
         'Le damos formato a la línea
         Call FormatoLinea( oTitulo, 1, RGB(100,50,25), 100 )
         'Si estableces la línea, tal vez sea buena idea, establecer los margenes
         'al texto para que no se vea tan pegado
         With oTitulo
               .ParaLeftMargin = 300
               .ParaRightMargin = 300
               .ParaTopMargin = 200
               .ParaBottomMargin = 200
         End With
         'Y lo establecemos
         oGrafico.setTitleObject( oTitulo )

         'Establecemos que se muestre el subtitulo
         oGrafico.HasSubTitle = True
         'Accedemos al subtitulo
         oSubTitulo = oGrafico.SubTitle
         'Le damos formato al texto, fondo y línea
         Call FormatoTexto( oSubTitulo, "América Latina", "Liberation Serif", 150, 12, RGB(255,150,0)
)
         Call FormatoFondo( oSubTitulo, 1, RGB(240,240,240) )
         Call FormatoLinea( oSubTitulo, 2, RGB(100,50,25), 50 )
   Else
         MsgBox "Selecciona un gráfico"
   End If

End Sub
```

Los margenes del borde al texto, solo existen en el título, no así en el subtítulo. Si usas normalmente un conjunto de fuentes, podrías usar una matriz de texto (*string*) con los nombres de las fuentes que más uses, por ejemplo.

```
Dim mFuentes(4) As String

   mFuentes(0) = "Liberation Serif"
   mFuentes(1) = "Liberation Sans"
   mFuentes(2) = "Liberation Mono"
   mFuentes(3) = "FreeMono"
   mFuentes(4) = "FreeSans"
```

Y mira que bien nos va quedando, bueno, no tan bien, ya sabes que soy muy malo para el diseño, así que no te me pongas exigente.

Ahora, cambiaremos el formato de la leyenda, el circulo (bueno, intento de circulo) rojo en la imagen anterior. Nota que en la llamada a la subrutina *FormatoTexto*, el argumento *Texto,* que nos sirve para establecer el título y el subtítulo, ahora, le pasamos una cadena vacía, esto es muy importante, pues la leyenda no implementa la propiedad *String*, por lo que si intentas establecerla, te dará un error en tiempo de ejecución, que claro, puedes interceptar y manipular.

```
Sub Graficando4()
Dim oGrafico As Object
Dim oLeyenda As Object

   oGrafico = getGrafico()
   If Not IsNull( oGrafico ) Then
         'Accedemos a la leyenda
         oLeyenda = oGrafico.Legend()
         'Le damos formato al texto, fondo y línea
         Call FormatoTexto( oLeyenda, "", "Liberation Sans", 150, 10, RGB(255,150,100) )
         Call FormatoFondo( oLeyenda, 1, RGB(240,240,240) )
         Call FormatoLinea( oLeyenda, 3, RGB(100,50,25), 50 )
   Else
         MsgBox "Selecciona un gráfico"
   End If

End Sub
```

En algunos casos, como en este ejemplo, no tiene mucho sentido mostrar la leyenda, pues es una sola serie de datos, así que mejor la ocultamos y establecemos el titulo de el eje X y el eje Y, así como su formato.

```
Sub Graficando5()
Dim oGrafico As Object
Dim oTituloEje As Object

   oGrafico = getGrafico()
   If Not IsNull( oGrafico ) Then
         'Ocultamos la leyenda
         oGrafico.HasLegend = False
         'Accedemos al titulo del eje X
         oTituloEje = oGrafico.getDiagram.XAxisTitle
         'Establecemos que se muestre
         oGrafico.getDiagram.HasXAxisTitle = True
         Call FormatoTexto( oTituloEje, "Países", "Liberation Sans", 150, 11, RGB(100,150,100) )
         Call FormatoFondo( oTituloEje, 1, RGB(240,240,240) )
         Call FormatoLinea( oTituloEje, 1, RGB(200,200,200), 20 )

         'Accedemos al titulo del eje Y
         oTituloEje = oGrafico.getDiagram.YAxisTitle
         'Establecemos que se muestre
         oGrafico.getDiagram.HasYAxisTitle = True
         Call FormatoTexto( oTituloEje, "Habitantes", "Liberation Sans", 150, 11, RGB(100,150,100) )
         Call FormatoFondo( oTituloEje, 1, RGB(240,240,240) )
         Call FormatoLinea( oTituloEje, 1, RGB(200,200,200), 20 )
   Else
         MsgBox "Selecciona un gráfico"
   End If

End Sub
```

El punto importante, es la forma en que accedemos a los títulos de los ejes (getDiagram), estos están contenidos “dentro” del gráfico, pero también “dentro” de un forma que se llama diagrama, que es propiamente el área donde se muestra el gráfico.

```
Sub Graficando6()
Dim oGrafico As Object
Dim oEje As Object

   oGrafico = getGrafico()
   If Not IsNull( oGrafico ) Then
         'Accedemos al eje X
         oEje = oGrafico.getDiagram.getXAxis
         Call FormatoTexto( oEje, "", "Liberation Sans", 150, 10, RGB(50,50,50) )
         Call FormatoLinea( oEje, 1, RGB(0,0,255), 30 )
         'Rotamos 60º el texto
         oEje.TextRotation = 6000
         'Accedemos al eje Y
         oEje = oGrafico.getDiagram.getYAxis
         Call FormatoTexto( oEje, "", "Liberation Sans", 150, 10, RGB(50,50,50) )
         Call FormatoLinea( oEje, 1, RGB(0,0,255), 30 )
         'Establecemos el limite superior del eje
         oEje.Max = 200000000
         'Establecemos el intervalo superior
         oEje.StepMain = 50000000
         'El número de marcas secundarias
         oEje.StepHelpCount = 5
         'Líneas del eje principal
         oEje = oGrafico.getDiagram.YMainGrid
         Call FormatoLinea( oEje, 1, RGB(255,0,0), 50 )
         'Mostramos las líneas secundarias
```

```
        oGrafico.getDiagram.HasYAxisHelpGrid = True
        oEje = oGrafico.getDiagram.YHelpGrid
        Call FormatoLinea( oEje, 1, RGB(150,0,0), 25 )
   Else
        MsgBox "Selecciona un gráfico"
   End If

End Sub
```

Modificamos las propiedades de toda el área del gráfico.

```
Sub Graficando7()
Dim oGrafico As Object
Dim oArea As Object

   oGrafico = getGrafico()
   If Not IsNull( oGrafico ) Then
        'Seleccionamos el área del gráfico
        oArea = oGrafico.getArea()
        'Establecemos el fondo en gradiente y su nombre
        Call FormatoFondo( oArea, 2, "Radial red/yellow" )
        Call FormatoLinea( oArea, 1, RGB(50,255,50), 20 )
   Else
        MsgBox "Selecciona un gráfico"
   End If

End Sub
```

Ahora, solo del fondo del gráfico, el área efectiva donde se muestran los datos.

```
Sub Graficando8()
Dim oGrafico As Object
Dim oFondo As Object

   oGrafico = getGrafico()
   If Not IsNull( oGrafico ) Then
        'Seleccionamos el fondo del gráfico
        oFondo = oGrafico.getDiagram.getWall
        Call FormatoFondo( oFondo, 2, 9 )
   Else
        MsgBox "Selecciona un gráfico"
   End If

End Sub
```

Modificamos la serie de datos, por ahora, solo tenemos una.

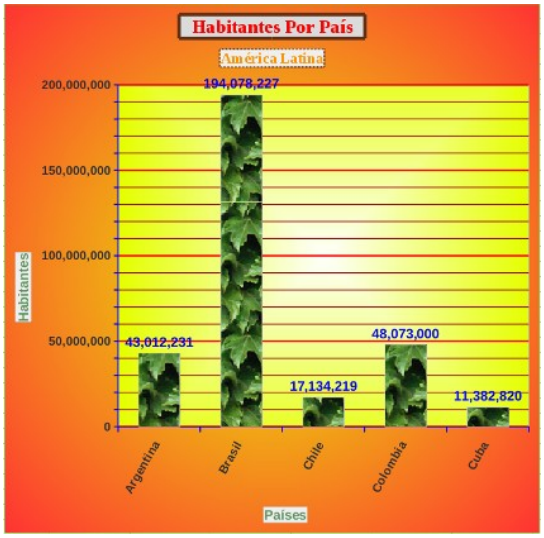
```
Sub Graficando9()
Dim oGrafico As Object
Dim oDatos As Object

   oGrafico = getGrafico()
   If Not IsNull( oGrafico ) Then
        'La primer serie de datos
        oDatos = oGrafico.getDiagram.getDataRowProperties(0)
        Call FormatoTexto( oDatos, "", "Liberation Sans", 150, 11, RGB(0,0,250) )
        Call FormatoFondo( oDatos, 4, 5 )
        'Establecemos que se muestren los valores de cada punto
        oDatos.DataCaption = 1
   Else
        MsgBox "Selecciona un gráfico"
```

```
    End If

End Sub
```

Para cambiar de tamaño un gráfico, tienes que hacerlo como si fuera una forma (*shape*) como te muestro en el siguiente ejemplo.

```
Sub Graficando10()
Dim oGrafico As Object
Dim oSel As Object
Dim oTam As New com.sun.star.awt.Size

   oGrafico = getGrafico()
   If Not IsNull( oGrafico ) Then
         'Aquí, repetimos lo que hace la función que nos regresa el gráfico
         'esto es por que ya estamos seguros de que es un gráfico y por que,
         'para cambiar de tamaño, hay que hacerlo como si fuera una forma (shape)
         oSel = ThisComponent.getCurrentSelection()
         oSel = oSel.getByIndex(0)
         'Establecemos el nuevo tamaño
         oTam.Width = 15000
         oTam.Height = 15000
         oSel.setSize( oTam )
   Else
         MsgBox "Selecciona un gráfico"
   End If

End Sub
```

Si has probado cada una de las macros de ejemplo sobre el mismo gráfico, tienes que tener algo así.



Si lo sé, esta horrible, al fin que no es curso de diseño.

La siguiente macro, cambiara el rango de datos origen, algo sumamente necesario para tener realmente un gráfico dinámico y podamos actualizarlo cuando sea necesario.

```
Sub Graficando11()
Dim oGrafico As Object
Dim oHojaActiva As Object
Dim mRangos(0)
```

```
Dim oDir As New com.sun.star.table.CellRangeAddress

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   'Accedemos al gráfico
   oGrafico = getGrafico2()
   If Not IsNull( oGrafico ) Then
         'El nuevo rango de datos
         With oDir
               .Sheet = oHojaActiva.getRangeAddress.Sheet
               .StartColumn = 0
               .EndColumn = 1
               .StartRow = 0
               .EndRow = 10
         End With
         mRangos(0) = oDir
         'Establecemos el nuevo rango
         oGrafico.setRanges( mRangos )
   Else
         MsgBox "Selecciona un gráfico"
   End If

End Sub
```

Nota que estamos usando una segunda versión de la función para regresar el gráfico seleccionado, la función es casi idéntica.

```
Function getGrafico2() As Object
Dim oHojaActiva As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim oSel As Object
Dim sNombre As String
Dim sInfo As String

   oSel = ThisComponent.getCurrentSelection()
   If oSel.getImplementationName = "com.sun.star.drawing.SvxShapeCollection" Then
         oSel = oSel.getByIndex(0)
         If oSel.supportsService("com.sun.star.drawing.OLE2Shape") Then
               sNombre = oSel.PersistName
               oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
               oGraficos = oHojaActiva.getCharts()
               If oGraficos.hasByName( sNombre ) Then
                     getGrafico2 = oGraficos.getByName( sNombre )
               End If
         End If
   End If

End Function
```

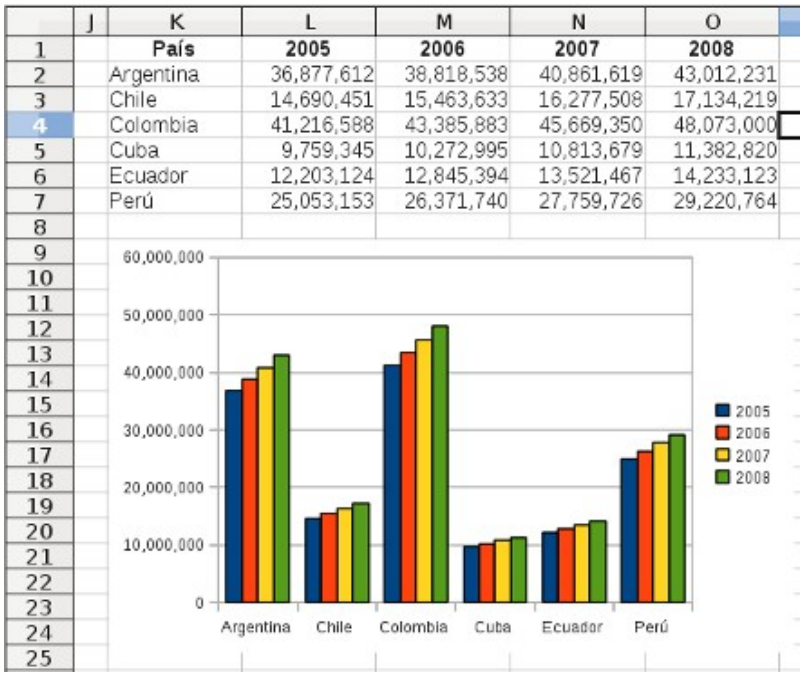
La diferencia, es que la primera te devuelve el objeto (no me gusta la palabra pero así es) “embebido” (*getEmbeddedObject*), con el cual tenemos acceso a todos los objetos dentro del gráfico, esta segunda forma, accede directamente al gráfico (*getByName*).

Veamos algunos ejemplos más de creación de gráficos, cuando grafiques, como ya lo mencionamos, debes de cuidar la correspondencia de tus datos con el tipo de gráfico, así mismo, cuando personalices un gráfico, asegurate de que el tipo de gráfico es correcto, por ejemplo, puedes establecerle ejes a un gráfico circular, pero dejaría de ser un gráfico circular y te aseguro que no obtendrías el resultado previsto.

El siguiente ejemplo, modifica nuestro gráfico para que se vea en 3D.

```
Sub Graficando12()
Dim oGrafico As Object

   oGrafico = getGrafico()
   If Not IsNull( oGrafico ) Then
         'Establecemos el gráfico en 3D
         oGrafico.getDiagram.Dim3D = True
         'Mostramos cilindros en vez de columnas
         oGrafico.getDiagram.SolidType = 1
   Else
         MsgBox "Selecciona un gráfico"
   End If

End Sub
```

Observa el siguiente gráfico, para obtenerlo, es indispensable que el cursor este en “una sola celda” de los datos, lo demás se calcula, tanto el rango de datos como la posición.

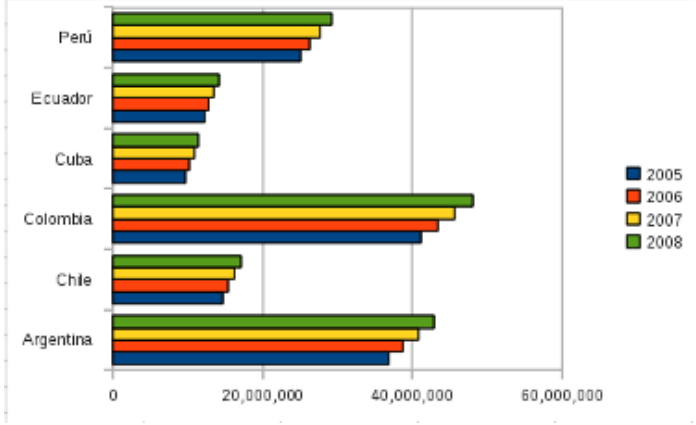



```
Sub Graficando13()
Dim oHojaActiva As Object
Dim oSel As Object
Dim oCursor As Object
Dim oGraficos As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oCelda As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oSel = ThisComponent.getCurrentController.getSelection()
   If oSel.getImplementationName = "ScCellObj" Then
         oCursor = oHojaActiva.createCursorByRange( oSel )
         'Expandimos el cursor a la región actual
         oCursor.collapseToCurrentRegion()
         mRangos(0) = oCursor.getRangeAddress
         sNombre = "Grafico10"
         'Celda para la posición del gráfico
```

```
        oCelda = oHojaActiva.getCellByPosition( oCursor.getRangeAddress.StartColumn,
oCursor.getRangeAddress.EndRow + 2 )
        With oRec
              .X = oCelda.Position.X
              .Y = oCelda.Position.Y
              .Width = 11500          'El ancho del gráfico
              .Height = 7000          'El alto del gráfico
        End With
        oGraficos = oHojaActiva.getCharts()
        If oGraficos.hasByName( sNombre ) Then
             MsgBox "Ya existe este nombre de gráfico, escoge otro"
        Else
              oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
        End If
   Else
        MsgBox "Selecciona solo una celda con datos"
   End If

End Sub
```

El mismo gráfico, pero ahora lo hacemos de barras.

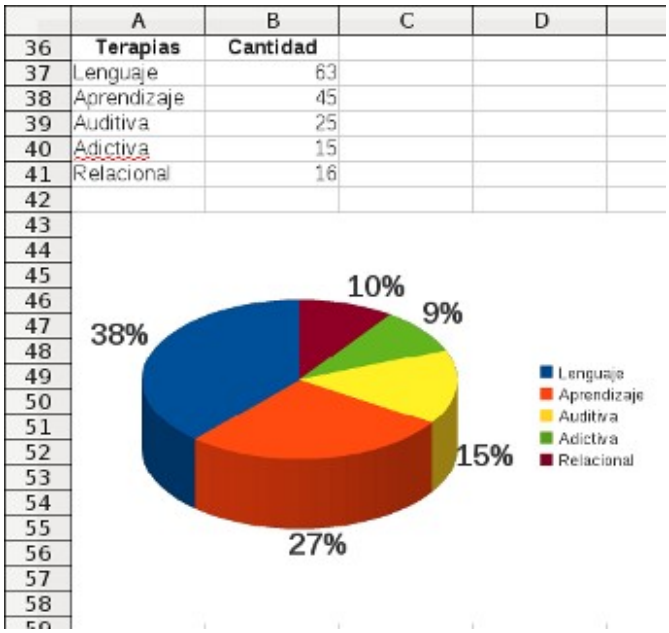
País	2005	2006	2007	2008
Argentina	36,877,612	38,818,538	40,861,619	43,012,231
Chile	14,690,451	15,463,633	16,277,508	17,134,219
Colombia	41,216,588	43,385,883	45,669,350	48,073,000
Cuba	9,759,345	10,272,995	10,813,679	11,382,820
Ecuador	12,203,124	12,845,394	13,521,467	14,233,123
Perú	25,053,153	26,371,740	27,759,726	29,220,764




```
Sub Graficando14()
Dim oHojaActiva As Object
Dim oSel As Object
Dim oCursor As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oCelda As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oSel = ThisComponent.getCurrentController.getSelection()
   If oSel.getImplementationName = "ScCellObj" Then
        oCursor = oHojaActiva.createCursorByRange( oSel )
        'Expandimos el cursor a la región actual
        oCursor.collapseToCurrentRegion()
```

```
        mRangos(0) = oCursor.getRangeAddress
        sNombre = "Grafico10"
        'Celda para la posición del gráfico
        oCelda = oHojaActiva.getCellByPosition( oCursor.getRangeAddress.StartColumn,
oCursor.getRangeAddress.EndRow + 2 )
        With oRec
              .X = oCelda.Position.X
              .Y = oCelda.Position.Y
              .Width = 11500         'El ancho del gráfico
              .Height = 7000         'El alto del gráfico
        End With
        oGraficos = oHojaActiva.getCharts()
        If oGraficos.hasByName( sNombre ) Then
             MsgBox "Ya existe este nombre de gráfico, escoge otro"
        Else
              oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
              oGrafico = oGraficos.getByName( sNombre )
              oGrafico.getEmbeddedObject.getDiagram.Vertical = True
        End If
   Else
        MsgBox "Selecciona solo una celda con datos"
   End If

End Sub
```

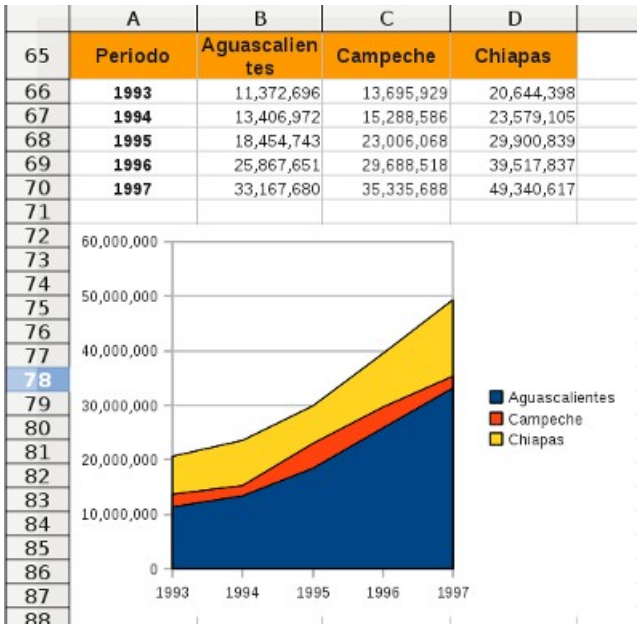
Los gráficos circulares sirven para darnos una imagen de la relación de cada punto respecto al total.




```
Sub Graficando15()
Dim oHojaActiva As Object
Dim oSel As Object
Dim oCursor As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oCelda As Object
Dim oDatos As Object
```

```
   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oSel = ThisComponent.getCurrentController.getSelection()
   If oSel.getImplementationName = "ScCellObj" Then
         oCursor = oHojaActiva.createCursorByRange( oSel )
         oCursor.collapseToCurrentRegion()
         mRangos(0) = oCursor.getRangeAddress
         sNombre = "Grafico15"
         oCelda = oHojaActiva.getCellByPosition( oCursor.getRangeAddress.StartColumn,
oCursor.getRangeAddress.EndRow + 2 )
         With oRec
               .X = oCelda.Position.X
               .Y = oCelda.Position.Y
               .Width = 10000
               .Height = 7000
         End With
         oGraficos = oHojaActiva.getCharts()
         If oGraficos.hasByName( sNombre ) Then
               MsgBox "Ya existe este nombre de gráfico, escoge otro"
         Else
               oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
               oGrafico = oGraficos.getByName( sNombre )
               'Cambiamos el tipo de gráfico
               oGrafico.getEmbeddedObject.setDiagram(
oGrafico.getEmbeddedObject.createInstance("com.sun.star.chart.PieDiagram") )
               oDatos = oGrafico.getEmbeddedObject.getDiagram.getDataRowProperties(0)
               oGrafico.getEmbeddedObject.getDiagram.Dim3D = True
               oGrafico.HasColumnHeaders = True
               oGrafico.HasRowHeaders = True
               'Hay que reasignar el rango de datos, lo pierde al cambiar de tipo de gráfico
               oGrafico.setRanges( mRangos )
               oDatos.DataCaption = 2
               oDatos.LabelPlacement = 0
               Call FormatoTexto( oDatos, "", "Liberation Sans", 150, 15, RGB(55,55,55) )
         End If
   Else
         MsgBox "Selecciona solo una celda con datos"
   End If

End Sub
```

Los gráficos de área, destacan la magnitud de un cambio en el tiempo.




```
Sub Graficando16()
Dim oHojaActiva As Object
Dim oSel As Object
Dim oCursor As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oCelda As Object
Dim oDatos As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oSel = ThisComponent.getCurrentController.getSelection()
   If oSel.getImplementationName = "ScCellObj" Then
         oCursor = oHojaActiva.createCursorByRange( oSel )
         oCursor.collapseToCurrentRegion()
         mRangos(0) = oCursor.getRangeAddress
         sNombre = "Grafico16"
         oCelda = oHojaActiva.getCellByPosition( oCursor.getRangeAddress.StartColumn,
oCursor.getRangeAddress.EndRow + 2 )
         With oRec
               .X = oCelda.Position.X
               .Y = oCelda.Position.Y
               .Width = 10000
               .Height = 7000
         End With
         oGraficos = oHojaActiva.getCharts()
         If oGraficos.hasByName( sNombre ) Then
               MsgBox "Ya existe este nombre de gráfico, escoge otro"
         Else
               oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
               oGrafico = oGraficos.getByName( sNombre )
               'Cambiamos el tipo de gráfico
               oGrafico.getEmbeddedObject.setDiagram(
oGrafico.getEmbeddedObject.createInstance("com.sun.star.chart.AreaDiagram") )
         End If
   Else
         MsgBox "Selecciona solo una celda con datos"
   End If
```

```
End Sub
```

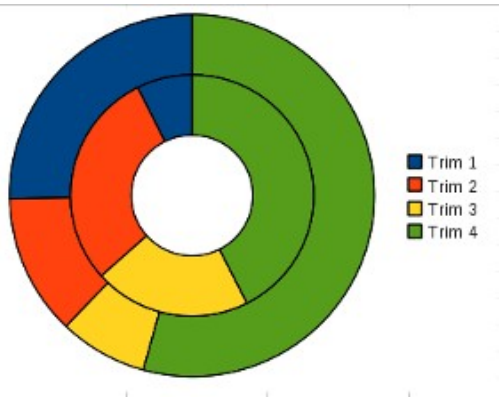
Los gráficos de líneas muestran principalmente los cambios de valor en el tiempo y su relación con otros valores.


```
Sub Graficando17()
Dim oHojaActiva As Object
Dim oSel As Object
Dim oCursor As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oCelda As Object
Dim oDatos As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oSel = ThisComponent.getCurrentController.getSelection()
   If oSel.getImplementationName = "ScCellObj" Then
         oCursor = oHojaActiva.createCursorByRange( oSel )
         oCursor.collapseToCurrentRegion()
         mRangos(0) = oCursor.getRangeAddress
         sNombre = "Grafico17"
         oCelda = oHojaActiva.getCellByPosition( oCursor.getRangeAddress.StartColumn,
oCursor.getRangeAddress.EndRow + 2 )
         With oRec
               .X = oCelda.Position.X
               .Y = oCelda.Position.Y
               .Width = 12000
               .Height = 7000
         End With
         oGraficos = oHojaActiva.getCharts()
         If oGraficos.hasByName( sNombre ) Then
               MsgBox "Ya existe este nombre de gráfico, escoge otro"
         Else
               oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
               oGrafico = oGraficos.getByName( sNombre )
```

```
            'Cambiamos el tipo de gráfico
            oGrafico.getEmbeddedObject.setDiagram(
oGrafico.getEmbeddedObject.createInstance("com.sun.star.chart.LineDiagram") )
            oGrafico.getEmbeddedObject.getDiagram.SymbolType = 1
      End If
   Else
      MsgBox "Selecciona solo una celda con datos"
   End If

End Sub
```

Los gráficos de anillo, son parecidos a los circulares, pero pueden representar más de una serie de datos, aun así, creo, no son una buena elección.

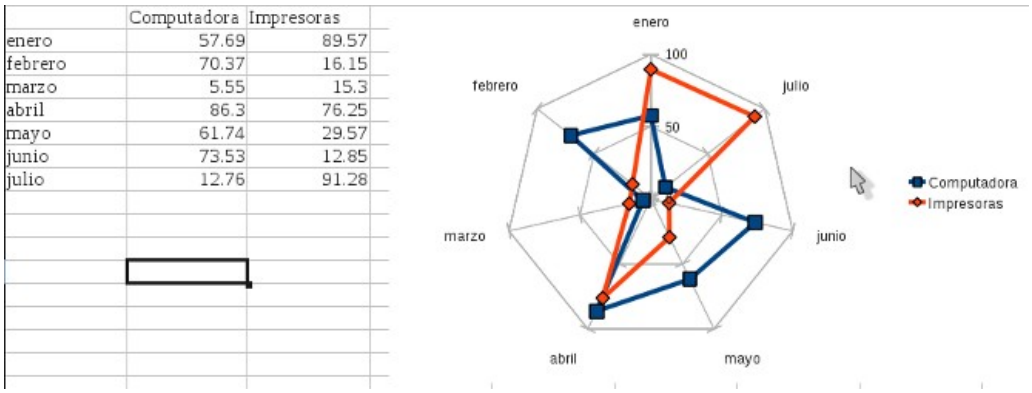
Trimestre	Juan	Pedro
Trim 1	38,163	17,426
Trim 2	18,985	68,971
Trim 3	11,823	49,214
Trim 4	81,978	100,388




```
Sub Graficando18()
Dim oHojaActiva As Object
Dim oSel As Object
Dim oCursor As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oCelda As Object
Dim oDatos As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oSel = ThisComponent.getCurrentController.getSelection()
   If oSel.getImplementationName = "ScCellObj" Then
      oCursor = oHojaActiva.createCursorByRange( oSel )
      oCursor.collapseToCurrentRegion()
      mRangos(0) = oCursor.getRangeAddress
      sNombre = "Grafico18"
      oCelda = oHojaActiva.getCellByPosition( oCursor.getRangeAddress.StartColumn,
oCursor.getRangeAddress.EndRow + 2 )
      With oRec
         .X = oCelda.Position.X
         .Y = oCelda.Position.Y
         .Width = 12000
         .Height = 7000
      End With
      oGraficos = oHojaActiva.getCharts()
      If oGraficos.hasByName( sNombre ) Then
         MsgBox "Ya existe este nombre de gráfico, escoge otro"
      Else
         oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
         oGrafico = oGraficos.getByName( sNombre )
```

```
            'Cambiamos el tipo de gráfico
            oGrafico.getEmbeddedObject.setDiagram(
oGrafico.getEmbeddedObject.createInstance("com.sun.star.chart.DonutDiagram") )
      End If
   Else
      MsgBox "Selecciona solo una celda con datos"
   End If

End Sub
```

Lo confieso, no le hayo mucho sentido a un gráfico radial, pero ahí esta el ejemplo.

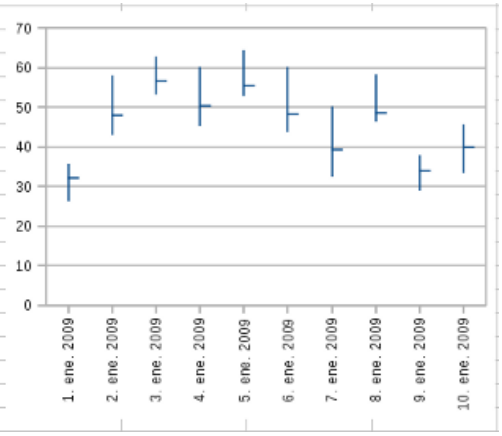



```
Sub Graficando19()
Dim oHojaActiva As Object
Dim oSel As Object
Dim oCursor As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oCelda As Object
Dim oDatos As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oSel = ThisComponent.getCurrentController.getSelection()
   If oSel.getImplementationName = "ScCellObj" Then
      oCursor = oHojaActiva.createCursorByRange( oSel )
      oCursor.collapseToCurrentRegion()
      mRangos(0) = oCursor.getRangeAddress
      sNombre = "Grafico19"
      oCelda = oHojaActiva.getCellByPosition( oCursor.getRangeAddress.StartColumn,
oCursor.getRangeAddress.EndRow + 2 )
      With oRec
         .X = oCelda.Position.X
         .Y = oCelda.Position.Y
         .Width = 12000
         .Height = 7000
      End With
      oGraficos = oHojaActiva.getCharts()
      If oGraficos.hasByName( sNombre ) Then
         MsgBox "Ya existe este nombre de gráfico, escoge otro"
      Else
         oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
         oGrafico = oGraficos.getByName( sNombre )
         'Cambiamos el tipo de gráfico
         oGrafico.getEmbeddedObject.setDiagram(
oGrafico.getEmbeddedObject.createInstance("com.sun.star.chart.NetDiagram") )
```

```
        End If
    Else
        MsgBox "Selecciona solo una celda con datos"
    End If

End Sub
```

Para un gráfico de stock, el orden de los datos es importante.

Día	Mínimo	Máximo	Cierre
1. ene. 2009	26.25	35.73	32.25
2. ene. 2009	43.02	58.06	48
3. ene. 2009	53.25	62.83	56.68
4. ene. 2009	45.28	60.2	50.4
5. ene. 2009	52.84	64.45	55.5
6. ene. 2009	43.77	60.18	48.22
7. ene. 2009	32.41	50.19	39.29
8. ene. 2009	46.41	58.31	48.52
9. ene. 2009	28.86	37.9	34.04
10. ene. 2009	33.41	45.65	39.85

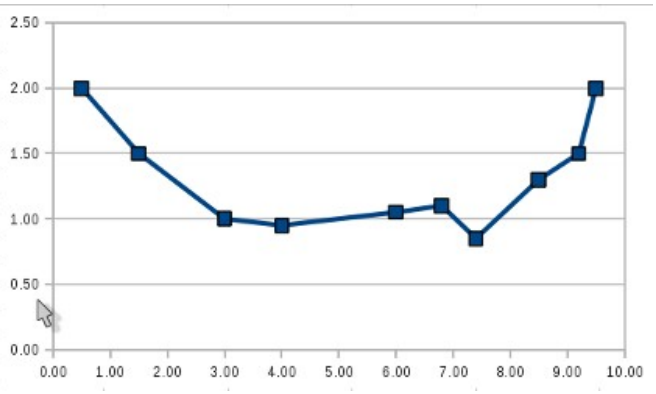



```
Sub Graficando20()
Dim oHojaActiva As Object
Dim oSel As Object
Dim oCursor As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oCelda As Object
Dim oDatos As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oSel = ThisComponent.getCurrentController.getSelection()
   If oSel.getImplementationName = "ScCellObj" Then
        oCursor = oHojaActiva.createCursorByRange( oSel )
        oCursor.collapseToCurrentRegion()
        mRangos(0) = oCursor.getRangeAddress
        sNombre = "Grafico20"
        oCelda = oHojaActiva.getCellByPosition( oCursor.getRangeAddress.StartColumn,
oCursor.getRangeAddress.EndRow + 2 )
        With oRec
            .X = oCelda.Position.X
            .Y = oCelda.Position.Y
            .Width = 12000
            .Height = 7000
        End With
        oGraficos = oHojaActiva.getCharts()
        If oGraficos.hasByName( sNombre ) Then
            MsgBox "Ya existe este nombre de gráfico, escoge otro"
        Else
            oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
            oGrafico = oGraficos.getByName( sNombre )
            'Cambiamos el tipo de gráfico
            oGrafico.getEmbeddedObject.setDiagram(
oGrafico.getEmbeddedObject.createInstance("com.sun.star.chart.StockDiagram") )
        End If
```

```
   Else
         MsgBox "Selecciona solo una celda con datos"
   End If

End Sub
```

El gráfico XY (dispersión) muestra la relación de un valor en función de otro.

Punto	Distancia	Altura
1	0.50	2.00
2	1.50	1.50
3	3.00	1.00
4	4.00	0.95
5	6.00	1.05
6	6.80	1.10
7	7.40	0.85
8	8.50	1.30
9	9.20	1.50
10	9.50	2.00




```
Sub Graficando21()
Dim oHojaActiva As Object
Dim oSel As Object
Dim oCursor As Object
Dim oGraficos As Object
Dim oGrafico As Object
Dim mRangos(0)
Dim sNombre As String
Dim oRec As New com.sun.star.awt.Rectangle
Dim oCelda As Object
Dim oDatos As Object

   oHojaActiva = ThisComponent.getCurrentController.getActiveSheet()
   oSel = ThisComponent.getCurrentController.getSelection()
   If oSel.getImplementationName = "ScCellObj" Then
         oCursor = oHojaActiva.createCursorByRange( oSel )
         oCursor.collapseToCurrentRegion()
         mRangos(0) = oCursor.getRangeAddress
         sNombre = "Grafico21"
         oCelda = oHojaActiva.getCellByPosition( oCursor.getRangeAddress.StartColumn,
oCursor.getRangeAddress.EndRow + 2 )
         With oRec
               .X = oCelda.Position.X
               .Y = oCelda.Position.Y
               .Width = 12000
               .Height = 7000
         End With
         oGraficos = oHojaActiva.getCharts()
         If oGraficos.hasByName( sNombre ) Then
               MsgBox "Ya existe este nombre de gráfico, escoge otro"
         Else
               oGraficos.addNewByName(sNombre, oRec, mRangos, True, True)
               oGrafico = oGraficos.getByName( sNombre ).getEmbeddedObject
               'Cambiamos el tipo de gráfico
               oGrafico.setDiagram( oGrafico.createInstance("com.sun.star.chart.XYDiagram") )
               oGrafico.HasLegend = False
         End If
   Else
         MsgBox "Selecciona solo una celda con datos"
   End If
```

```
End Sub
```

Cada tipo de gráfico tiene sus particularidades que tendrás que considerar a la hora de graficar, la recomendación general es, inmediatamente que agregues el gráfico, cambia su tipo al que necesites y solo al final, estableces todas las propiedades que quieras.

Si sumas la importación de bases de datos, con el gráfico de datos, tienes una combinación bastante poderosa y eficiente para representar tus datos en informes.

6.10 Trabajando con elementos gráficos

No se si estarás de acuerdo conmigo, pero poco a poco se ha ido diluyendo la frontera entre aplicaciones, es decir, entre un procesador de textos y una hoja de calculo por ejemplo, en uno y en otro podemos hacer muchas tareas similares, soy de la opinión de seguir usando cada aplicación especifica para la tarea que fue diseñada, no obstante, podemos hacer uso de esas características extras, tan variadas y ricas con que cuentan las aplicaciones actuales, lo cual aprenderemos en este capitulo.

6.10.1 Trabajando con imágenes

Calc, soporta la inclusión de una amplia variedad de formatos de imágenes, en el siguiente ejemplo, insertamos una imagen PGN en la hoja activa.

```
Sub Imagenes1()
Dim oDoc As Object
Dim oPaginaDibujo As Object
Dim oImagen As Object
Dim sRuta As String
Dim oTam As New com.sun.star.awt.Size

   'La ruta de la imagen
   sRuta = ConvertToURL("/home/mau/globo.png")
   oDoc = ThisComponent
   'Pagina de dibujo de la hoja activa
   oPaginaDibujo = oDoc.getCurrentController.getActiveSheet.getDrawPage()
   'Para crear y manipular imagenes
   oImagen = oDoc.createInstance( "com.sun.star.drawing.GraphicObjectShape" )
   'Establecemos la ruta de la imagen
   oImagen.GraphicURL = sRuta
   'La agregamos a la página de dibujo, por ende, al conjunto de formas
   oPaginaDibujo.add( oImagen )
   'Establecemos el tamaño de la imagen, siempre establece un tamaño, si no
   'se insertará con un tamaño mínimo casi invisible
   'la unidad es centésimas de milímetro
   oTam.Width = 10000
   oTam.Height = 7500
```

```
    oImagen.setSize( oTam )

End Sub
```

Ahora permitimos al usuario seleccionar una imagen, el método para abrir un archivo lo usamos casi al principio de estos apuntes, pero aquí lo recordamos.

```
Sub Imagenes2()
Dim oDlgAbrirArchivo As Object
Dim mArchivo()
Dim sRuta As String
Dim oPaginaDibujo As Object
Dim oImagen As Object
Dim oTam As New com.sun.star.awt.Size

   'Creamos el servicio necesario
   oDlgAbrirArchivo = CreateUnoService ("com.sun.star.ui.dialogs.FilePicker")
   'Establecemos los filtros de archivo
   oDlgAbrirArchivo.appendFilter( "Todos los formatos", "*.png;*.jpg;*.jpge;*.bmp;*.tiff")
   oDlgAbrirArchivo.appendFilter( "Imagenes PNG", "*.png")
   oDlgAbrirArchivo.appendFilter( "Imagenes JPG", "*.jpg")
   'Establecemos el titulo del cuadro de dialogo
   oDlgAbrirArchivo.setTitle("Selecciona la imagen")
   'Con el metodo .Execute() mostramos el cuadro de dialogo
   'Si el usuario presiona Abrir el metodo devuelve 1 que podemos evaluar como Verdadero (True)
   'Si presiona Cancelar devuelve 0
   If oDlgAbrirArchivo.Execute() Then
         'De forma predeterminada, solo se puede seleccionar un archivo
         'pero devuelve una matriz de todos modos con la ruta completa
         'del archivo en formato URL
         mArchivo() = oDlgAbrirArchivo.getFiles()
         'El primer elemento de la matriz es el archivo seleccionado
         sRuta = mArchivo(0)
         'Insertamos la imagen
         oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
         oImagen = ThisComponent.createInstance( "com.sun.star.drawing.GraphicObjectShape" )
         oImagen.GraphicURL = sRuta
         oPaginaDibujo.add( oImagen )
         oTam.Width = 10000
         oTam.Height = 7500
         oImagen.setSize( oTam )
   Else
         'Si el usuario presiona Cancelar
         MsgBox "Proceso cancelado"
   End If

End Sub
```

El método para abrir archivos, podrías convertirla en una función para que te devuelva el nombre o nombres de los archivos abiertos, aquí te muestro una primer forma que nos servirá para nuestros siguientes ejemplos, pero tu puedes mejorarla, por ejemplo, pasándole el título del dialogo y los filtros que soporte.

```
'Función para devolver la imagen a insertar
Function SelecionarImagen() As String
Dim oDlgAbrirArchivo As Object
Dim mArchivos()

   oDlgAbrirArchivo = CreateUnoService ("com.sun.star.ui.dialogs.FilePicker")
   oDlgAbrirArchivo.appendFilter( "Todos los formatos de imagen", "*.png;*.jpg;*.jpge;*.bmp;*.tiff")
   oDlgAbrirArchivo.appendFilter( "Imagenes PNG", "*.png")
   oDlgAbrirArchivo.appendFilter( "Imagenes JPG", "*.jpg")
```

```
    oDlgAbrirArchivo.setTitle("Selecciona la imagen a insertar")
    If oDlgAbrirArchivo.Execute() Then
          mArchivos() = oDlgAbrirArchivo.getFiles()
          SelecionarImagen = mArchivos(0)
    End If

End Function
```

Comprueba que funciona como se espera.

```
Sub Imagenes3()
Dim sRuta As String

   sRuta = SelecionarImagen()
   MsgBox sRuta

End Sub
```

La siguiente subrutina, inserta la imagen pasada como argumento en la hoja activa, puedes mejorarla pasándole la hoja donde se insertará, la posición y el tamaño, por ahora, para nuestro ejemplo, así nos sirve.

```
'Subrutina para insertar la imagen en la hoja activa
Sub InsertaImagen(RutaImagen As String)
Dim oDoc As Object
Dim oPaginaDibujo As Object
Dim oImagen As Object
Dim oTam As New com.sun.star.awt.Size

   oDoc = ThisComponent
   oPaginaDibujo = oDoc.getCurrentController.getActiveSheet.getDrawPage()
   oImagen = oDoc.createInstance( "com.sun.star.drawing.GraphicObjectShape" )
   oImagen.GraphicURL = RutaImagen
   oPaginaDibujo.add( oImagen )
   oTam.Width = 10000
   oTam.Height = 7500
   oImagen.setSize( oTam )

End Sub
```

Y la probamos.

```
Sub Imagenes4()
Dim sRuta As String

   sRuta = SelecionarImagen()
   If sRuta <> "" Then
         Call InsertarImagen( sRuta )
   End If

End Sub
```

La subrutina la puedes convertir en función y devolver una referencia a la imagen insertada, de este modo, puedes seguir manipulándola. Nota que en la función no establecemos el tamaño de la imagen, por lo que es importante que lo hagas después de llamar a esta función.

```
'Función para insertar la imagen en la hoja activa, devuelve la imagen
'nota que no establecemos el tamaño
Function getImagen(RutaImagen As String) As Object
```

```
Dim oDoc As Object
Dim oPaginaDibujo As Object
Dim oImagen As Object

   oDoc = ThisComponent
   oPaginaDibujo = oDoc.getCurrentController.getActiveSheet.getDrawPage()
   oImagen = oDoc.createInstance( "com.sun.star.drawing.GraphicObjectShape" )
   oImagen.GraphicURL = RutaImagen
   oPaginaDibujo.add( oImagen )
   getImagen = oImagen

End Function
```

Y la prueba.

```
Sub Imagenes5()
Dim sRuta As String
Dim oImagen As Object
Dim oTam As New com.sun.star.awt.Size

   sRuta = SelecionarImagen()
   If sRuta <> "" Then
         oImagen = getImagen( sRuta )
         oTam.Width = 10000
         oTam.Height = 7500
         oImagen.setSize( oTam )
   End If

End Sub
```

Y ya encarrerados, crea la subrutina para establecer el tamaño, que servirá para cualquier objeto que soporte estas propiedades.

```
Sub CambiaTam( Obj As Object, Ancho As Long, Alto As Long )
Dim oTam As New com.sun.star.awt.Size

   oTam.Width = Ancho
   oTam.Height = Alto
   Obj.setSize( oTam )

End Sub
```

Hasta ahora, las imágenes insertadas, siempre lo hacen ancladas (*Anchor*) a la celda A1, vamos a cambiar esto y por consiguiente la posición de la imagen.

```
Sub Imagenes6()
Dim sRuta As String
Dim oImagen As Object
Dim oCelda As Object

   sRuta = SelecionarImagen()
   If sRuta <> "" Then
         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 10000, 7500 )
         'Establecemos la celda de anclaje, al modificar esta se modifica la posición
         oCelda = ThisComponent.getCurrentController.getActiveSheet.getCellByPosition( 4,9 )
         oImagen.Anchor = oCelda
   End If

End Sub
```

El ancla también la puedes establecer a la hoja.

```
Sub Imagenes7()
Dim sRuta As String
Dim oImagen As Object

   sRuta = SelecionarImagen()
   If sRuta <> "" Then
         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 10000, 7500 )
         'Establecemos la hoja como ancla
         oImagen.Anchor = ThisComponent.getCurrentController.getActiveSheet
   End If

End Sub
```

Como ya lo comprobamos, si cambias el ancla de la imagen a una celda, la imagen cambia a la posición de dicha celda, pero también puedes establecer esta posición, no importando si el ancla esta a la hoja o a una celda.

```
Sub Imagenes8()
Dim sRuta As String
Dim oImagen As Object
Dim oPos As New com.sun.star.awt.Point
   sRuta = SelecionarImagen()
   If sRuta <> "" Then
         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 10000, 7500 )
         'Establecemos la posición de la imagen
         oPos.X = 15000
         oPos.y = 5000
         oImagen.setPosition( oPos )
   End If

End Sub
```

Algunas propiedades interesantes; cuando insertas o eliminas filas o columnas, la imagen se verá afectada en su tamaño, puedes evitarlo protegiendo su tamaño (*SizeProtect*), también puedes evitar que se mueva (*MoveProtect*) y que se imprima (*Printable*).

```
Sub Imagenes9()
Dim sRuta As String
Dim oImagen As Object

   sRuta = SelecionarImagen()
   If sRuta <> "" Then
         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 10000, 7500 )

         With oImagen
               'Nombramos la imagen
               .Name = "Imagen09"
               'Evitamos que la muevan
               .MoveProtect = True
               'Que cambie su tamaño
               .SizeProtect = True
               'Que se imprima
               .Printable = False
               'Puedes reflejar la imagen
               .IsMirrored = True
```

```
        End With
    End If

End Sub
```

El nombre asignado a la imagen (*Name*), es el que puede establecer con el menú contextual de la imagen, así mismo, es el nombre que aparecerá en el navegador, es importante que lo asignes, sobre todo si manejas muchas imágenes para que se muestre en el navegador.

Puedes tener varias imágenes con el mismo nombre. A las imágenes, les puedes cambiar el modo de color (*GraphicColorMode*), según la siguiente enumeración.

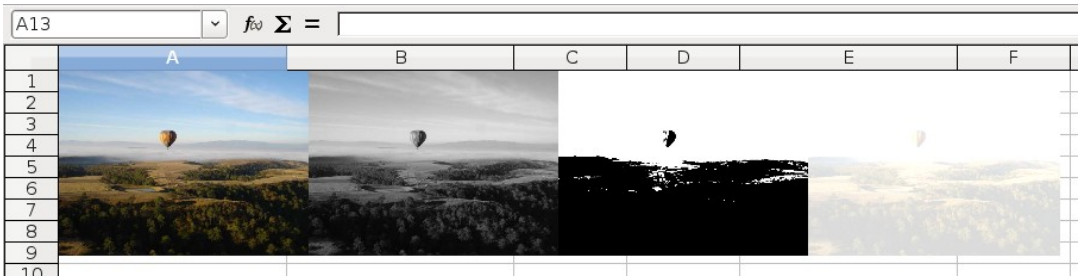
com.sun.star.drawing.ColorMode	*Valor*	*Valor en Interfaz*
com.sun.star.drawing.ColorMode.STANDARD	0	Predeterminado
com.sun.star.drawing.ColorMode.GREYS	1	Escala de grises
com.sun.star.drawing.ColorMode.MONO	2	Blanco y negro
com.sun.star.drawing.ColorMode.WATERMARK	3	Filigrana

El siguiente ejemplo, inserta la imagen varias veces son los diferentes modos.

```
Sub Imagenes10()
Dim sRuta As String
Dim oImagen As Object
Dim oPos As New com.sun.star.awt.Point

   sRuta = SelecionarImagen()
   If sRuta <> "" Then
         'Insertamos la imagen normal
         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 5000, 3750 )

         'Insertamos la misma imagen
         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 5000, 3750 )
         Call CambiaPos( oImagen, 5000, 0 )
         'Cambiamos el modo de color a escala de grises
         oImagen.GraphicColorMode = com.sun.star.drawing.ColorMode.GREYS

         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 5000, 3750 )
         Call CambiaPos( oImagen, 10000, 0 )
         'Cambiamos el modo de color a blanco y negro
         oImagen.GraphicColorMode = com.sun.star.drawing.ColorMode.MONO

         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 5000, 3750 )
         Call CambiaPos( oImagen, 15000, 0 )
         'Cambiamos el modo de color a marca de agua
         oImagen.GraphicColorMode = com.sun.star.drawing.ColorMode.WATERMARK

   End If

End Sub

'Cambia la posición del objeto
Sub CambiaPos( Obj As Object, X As Long, Y As Long )
Dim oPos As New com.sun.star.awt.Point

   oPos.X = X
   oPos.Y = Y
```

```
   Obj.setPosition( oPos )
End Sub
```

El resultado.



Puedes cambiar la transparencia (*Transparency*) de la imagen, este valor va de 0 a 100, donde 0 es el valor predeterminado, sin transparencia y 100 totalmente transparente, si estableces este valor en 100 y pierdes la selección, no veras la imagen por lo que procura no establecerlo tan alto.

```
Sub Imagenes11()
Dim sRuta As String
Dim oImagen As Object
Dim oPos As New com.sun.star.awt.Point

   sRuta = SelecionarImagen()
   If sRuta <> "" Then
         'Insertamos la imagen normal
         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 5000, 3750 )

         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 5000, 3750 )
         Call CambiaPos( oImagen, 5000, 0 )
         'Cambiamos el nivel de transparencia
         oImagen.Transparency = 25

         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 5000, 3750 )
         Call CambiaPos( oImagen, 10000, 0 )
         oImagen.Transparency = 50

         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 5000, 3750 )
         Call CambiaPos( oImagen, 15000, 0 )
         oImagen.Transparency = 75

   End If

End Sub
```

Puedes manipular completamente los colores de una imagen (si sabes por que yo no), tanto la luminosidad (*AdjustLuminance*), como el contraste (*AdjustContrast*), el canal gamma (*Gamma*). La luminosidad y el contraste, toman valores de 0 a 100, el canal gamma, de 0.1 a 10.

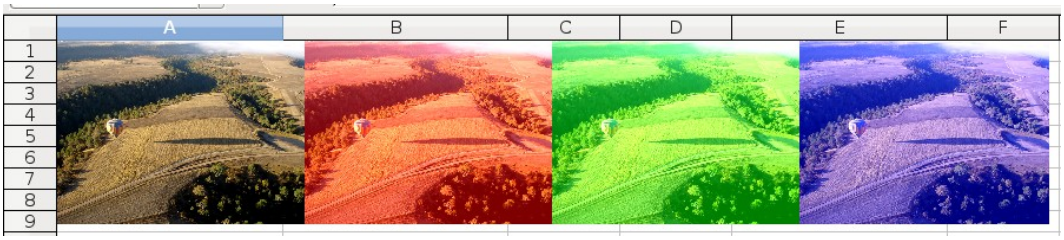
```
Sub Imagenes12()
Dim sRuta As String
Dim oImagen As Object
Dim oPos As New com.sun.star.awt.Point
```

```
    sRuta = SelecionarImagen()
    If sRuta <> "" Then
          'Insertamos la imagen normal
          oImagen = getImagen( sRuta )
          Call CambiaTam( oImagen, 5000, 3750 )

          oImagen = getImagen( sRuta )
          Call CambiaTam( oImagen, 5000, 3750 )
          Call CambiaPos( oImagen, 5000, 0 )
          'Cambiamos el nivel de luminosidad
          oImagen.AdjustLuminance = 50

          oImagen = getImagen( sRuta )
          Call CambiaTam( oImagen, 5000, 3750 )
          Call CambiaPos( oImagen, 10000, 0 )
          'Cambiamos el contraste
          oImagen.AdjustContrast = 50

          oImagen = getImagen( sRuta )
          Call CambiaTam( oImagen, 5000, 3750 )
          Call CambiaPos( oImagen, 15000, 0 )
          'Cambiamos el canal gamma, que no se que sea, pero cambia
          oImagen.Gamma = 5

    End If

End Sub
```

Y por supuesto, también puedes cambiar los canales: rojo (*AdjustRed*), verde (*AdjustGreen*) y azul (*AdjustBlue*), de la siguiente manera.

```
Sub Imagenes13()
Dim sRuta As String
Dim oImagen As Object
Dim oPos As New com.sun.star.awt.Point

    sRuta = SelecionarImagen()
    If sRuta <> "" Then
          'Insertamos la imagen normal
          oImagen = getImagen( sRuta )
          Call CambiaTam( oImagen, 5000, 3750 )

          oImagen = getImagen( sRuta )
          Call CambiaTam( oImagen, 5000, 3750 )
          Call CambiaPos( oImagen, 5000, 0 )
          'Cambiamos el nivel de rojo
          oImagen.AdjustRed = 50

          oImagen = getImagen( sRuta )
          Call CambiaTam( oImagen, 5000, 3750 )
          Call CambiaPos( oImagen, 10000, 0 )
          'Cambiamos el nivel de verde
          oImagen.AdjustGreen = 50

          oImagen = getImagen( sRuta )
          Call CambiaTam( oImagen, 5000, 3750 )
          Call CambiaPos( oImagen, 15000, 0 )
          'Cambiamos el nivel de azul
          oImagen.AdjustBlue = 50

    End If

End Sub
```

Mira que divertido queda.



Las imágenes insertadas tienen propiedades particulares, como las que hemos visto hasta ahora, pero al ser consideradas formas (*shapes*) comparten con ellas muchas de sus propiedades, por ejemplo, la posibilidad de agregarles texto, como en:

```
Sub Imagenes14()
Dim sRuta As String
Dim oImagen As Object

   sRuta = SelecionarImagen()
   If sRuta <> "" Then
         oImagen = getImagen( sRuta )
         Call CambiaTam( oImagen, 10000, 7500 )

         'Insertamos texto en la imagen y le cambiamos sus propiedades
         With oImagen
               .String = "Apulco, Hidalgo" & Chr(13) & "México"
               .CharColor = RGB(255,255,255)
               .CharHeight = 30
               .CharWeight = 150
               .ParaAdjust = 3
         End With

   End If

End Sub
```

Otras propiedades las veremos en el siguiente apartado, para terminar este tema, en todos los ejemplos anteriores, al insertar la imagen, la estás vinculando, por lo que si cambias el nombre de la imagen origen, la mueves o borras, cuando abras tu archivo, te quedará solo un marco vacío, para insertar la imagen incrustada en el documento, usa el siguiente código.

```
Sub Imagenes15()
```

```
Dim sRuta As String
Dim oImagen As Object
Dim oForma As Object
Dim oGP As Object
Dim mOpc(0) As New com.sun.star.beans.PropertyValue

   sRuta = SelecionarImagen()
   If sRuta <> "" Then
         'Agregamos una forma a la hoja activa
         oForma = ThisComponent.createInstance("com.sun.star.drawing.GraphicObjectShape")
         ThisComponent.getCurrentController.getActiveSheet.getDrawPage().add(oForma)
         'Servicio para manipulacion de gráficos
         oGP = createUnoService("com.sun.star.graphic.GraphicProvider")
         'Establecemos la ruta
         mOpc(0).Name  = "URL"
         mOpc(0).Value = sRuta
         'Trae la imagen y la carga en la forma
         oForma.Graphic = oGP.queryGraphic( mOpc )
         'Cambiamos el tamaño
         Call CambiaTam( oForma, 10000, 7500 )
   End If

End Sub
```

El cual, por supuesto, puedes convertir en una función o subrutina para llamarla cuando quieras, pero esa, es tu tarea. El siguiente ejemplo es muy divertido, toma la selección y determina si es o no una imagen, solicita una ruta y nombre donde guardar y la guarda si el usuario no cancela la operación en formato “jpg”, puedes exportar a otros formatos diferentes.

```
Sub Imagenes16()
Dim sRuta As String
Dim oSel As Object
Dim oGP As Object
Dim mOpc(1) As New com.sun.star.beans.PropertyValue

   'La selección actual
   oSel = ThisComponent.getCurrentController.getSelection
   iF oSel.getImplementationName = "com.sun.star.drawing.SvxShapeCollection" Then
         'Si es una forma, siempre es la primera
         oSel = oSel.getByIndex(0)
         If oSel.supportsService("com.sun.star.drawing.GraphicObjectShape") Then
               'Si es una imagen, obtenemos la ruta y nombre para guardar
               sRuta = RutaGuardarImagen()
               If sRuta <> "" Then
                     oGP = createUnoService("com.sun.star.graphic.GraphicProvider")
                     'Establecemos las propiedades
                     mOpc(0).Name = "URL"
                     mOpc(0).Value = sRuta
                     mOpc(1).Name = "MimeType"
                     mOpc(1).Value = "image/jpeg"
                     'Guardamos la imagen
                     oGP.storeGraphic( oSel.Graphic, mOpc )
               Else
                     MsgBox "Proceso cancelado"
               End If
         Else
               MsgBox "La selección es una forma pero no una imagen"
         End If
   Else
         MsgBox "La selección no es una imagen"
   End If

End Sub
```

La función para devolver la ruta es.

```
'Función para devolver la ruta y nombre del archivo a guardar
'Puedes mejorarla pasándole los filtros que quieras
Function RutaGuardarImagen() As String
Dim oDlgGuardarArchivo As Object
Dim mArchivo()
Dim mDlgOpciones()

   mDlgOpciones() = Array(2)
   oDlgGuardarArchivo = CreateUnoService ("com.sun.star.ui.dialogs.FilePicker")
   oDlgGuardarArchivo.setTitle("Guardar como")
   oDlgGuardarArchivo.Initialize ( mDlgOpciones() )
   oDlgGuardarArchivo.AppendFilter( "Imagen JPG (.jpg)", "*.jpg" )
   If oDlgGuardarArchivo.Execute() Then
         mArchivo() = oDlgGuardarArchivo.getFiles()
         RutaGuardarImagen = mArchivo(0)
   End If

End Function
```

Y con esto terminamos el tema de imágenes, que puedes complementar perfectamente con los conocimientos de nuestro próximo apartado, pues muchas de las propiedades y métodos que veremos, son soportados por las imágenes.

6.10.2 Trabajando con autoformas

Con las herramientas de dibujo incluidas en la aplicación, se podría dibujar casi cualquier cosa, el limite, es tu imaginación, veamos, porqué. Cada hoja de nuestro archivo, tiene una "página de dibujo virtual" (*DrawPage*), donde están todos los elementos gráficos, para acceder a esta hoja, usamos.

```
Sub AutoFormas1()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   'Accedemos a la página de dibujo
   oPaginaDibujo = oHojaActiva.getDrawPage()

   'Mostramos el número de elementos en la página
   MsgBox oPaginaDibujo.getCount

End Sub
```

La cuenta de los objetos gráficos incluye las notas de las celdas, por lo que si te muestra un número y aparentemente no hay nada en la hoja, tal vez este sea tu caso, otras veces, hay elementos como imágenes que tienen un tamaño mínimo y no se notan o están posicionadas en zonas de la hoja muy separadas. El siguiente ejemplo, cambia el tamaño y la posición de "todos" los objetos gráficos de la hoja y los posiciona en la esquina superior izquierda de la hoja, las subrutinas "*CambiaTam*" y "*CambiaPos*", ya las hemos usado anteriormente.

```
Sub AutoFormas2()
```

```
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim co1 As Long

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   'Accedemos a la página de dibujo
   oPaginaDibujo = oHojaActiva.getDrawPage()
   'Iteramos en cada forma
   For co1 = 0 To oPaginaDibujo.getCount - 1
         oForma = oPaginaDibujo.getByIndex(co1)
         'Cambiamos su tamaño
         Call CambiaTam( oForma, 3000, 3000 )
         'Y su posición
         Call CambiaPos( oForma, 0, 0 )
   Next

End Sub
```

Si en tu hoja hay notas, estas se cambiaran de tamaño pero el cambio de posición solo se verá reflejado cuando muestres la nota, no cuando pongas el cursor sobre la celda, que siempre muestra las notas cerca de esta, si no cuando la muestres permanentemente.

Para agregar una forma a la página de dibujo y por consiguiente a la hoja activa (puede ser a la hoja que quieras, no necesariamente la activa), primero creas una instancia de la forma que quieras, bueno, casi de la que quieras por que en las hojas de calculo no puedes agregar todos los tipos de formas, por ejemplo, una polilínea no puedes agregarla, lo cual si puedes hacer en Draw o Impress, entonces, después de agregar la forma soportada en Calc, le estableces sus propiedades y al final la agregas a la página de dibujo. Como con las imágenes, es importante que al menos cambies el tamaño de la nueva forma, si no, se creará del tamaño mínimo, la nueva forma se crea con las propiedades predeterminadas, las cuales, aprenderemos a cambiar aquí. En el siguiente ejemplo, agregamos un rectángulo a la hoja.

```
Sub AutoFormas3()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   'Creamos un rectangulo
   oForma = ThisComponent.createInstance("com.sun.star.drawing.RectangleShape")
   Call CambiaTam( oForma, 5000, 3000 )
   oPaginaDibujo.add( oForma )

End Sub
```

¿Adivina que pasa si estableces el mismo alto y ancho?, es obvia la respuesta, en vez de un rectángulo, tienes un cuadrado, lo mismo pasa con las elipses, si establece el mismo ancho y alto, obtienes un circulo.

```
Sub AutoFormas4()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   'Creamos un circulo
   oForma = ThisComponent.createInstance("com.sun.star.drawing.EllipseShape")
   Call CambiaTam( oForma, 5000, 5000 )
   oPaginaDibujo.add( oForma )
```

```
End Sub
```

La mayoría de las formas, comparten casi todas las mismas propiedades (línea, relleno, texto, sombra, etc), veamos las principales que comparten entre si y después las particulares de algunas.

6.10.2.1 Principales propiedades de línea.

```
Sub AutoFormas5()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   'Creamos una elipse
   oForma = ThisComponent.createInstance("com.sun.star.drawing.EllipseShape")
   Call CambiaTam( oForma, 10000, 5000 )
   With oForma
         'El estilo de línea
         .LineStyle = com.sun.star.drawing.LineStyle.SOLID
         'El color de la línea
         .LineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
         'La transparencia de la línea
         .LineTransparence = 50
         'El ancho de la línea
         .LineWidth = 500
   End With
   oPaginaDibujo.add( oForma )

End Sub
```

El estilo de línea, esta determinado por la siguiente enumeración.

com.sun.star.drawing.LineStyle	*Valor*	*Valor en Interfaz*
com.sun.star.drawing.LineStyle.NONE	0	Oculta
com.sun.star.drawing.LineStyle.SOLID	1	Solida
com.sun.star.drawing.LineStyle.DASH	2	Guiones

El color (*LineColor*) de la línea es un valor tipo largo (*long*), la transparencia (LineTransparence) puede tomar valores de 0 (menos transparencia) a 100 (más transparencia), si establece el valor en cero la línea no se verá como si establecieras el estilo en oculta (*NONE*), el ancho de la línea se establece en centésimas de milímetro y se reparte de forma equidistante del dentro hacia afuera y dentro de la forma, si estableces este valor en cero, no desaparece completamente, queda visible aun, por ello, mejor usa el estilo si lo que quieres es no mostrar la línea. Si estableces el estilo en guiones (*DASH*), puedes establecer el estilo de este de dos maneras, la primera más sencilla, estableces el nombre de estilo como en.

```
Sub AutoFormas6()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   'Seleccionamos el primero objeto de dibujo
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el estilo en guiones
   oForma.LineStyle = com.sun.star.drawing.LineStyle.DASH
```

```
    'Establecemos el estilo
    oForma.LineDashName = "Fine Dashed"

End Sub
```

Los nombres que puedes establecer son:

Si las vas a usar seguido, lo más practico es crear una matriz con los nombres.

```
Sub AutoFormas7()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim mEstilos()

   mEstilos = Array("Ultrafine Dashed","Fine Dashed","Ultrafine 2 Dots 3 Dashes","Fine Dotted","Line
with Fine Dots","Fine Dashed (var)","3 Dashes 3 Dots (var)","Ultrafine Dotted (var)","Line Style
9","2 Dots 1 Dash")
   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   'Seleccionamos el primero objeto de dibujo
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el estilo en guiones
   oForma.LineStyle = com.sun.star.drawing.LineStyle.DASH
   'Establecemos el estilo
   oForma.LineDashName = mEstilos(4)

End Sub
```

Recuerda que los estilos de línea, son completamente personalizables desde la interfaz del usuario, por lo que estos nombres pueden cambiar, el nombre debe corresponder exactamente, incluso sus mayúsculas, minúsculas, números y espacios, con el nombre de la interfaz, de lo contrario, te dará un error en tiempo de ejecución. Si quieres asegurate de que no haya error, puedes usar la otra forma de establecer el estilo, creando un estilo completamente nuevo y personalizado, el cual, esta estructurado de la siguiente manera.

1) Número de puntos (*Dots*)
2) Número de guiones (*Dashes*)
3) Ancho del punto (*DotLen*)
4) Distancia en elementos (*Distance*)
5) Ancho del guión (*DashLen*)

Para crear un estilo similar al anterior, usamos el siguiente código.

```
Sub AutoFormas8()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim oLineaGuion As Object

   oLineaGuion = createUnoStruct("com.sun.star.drawing.LineDash")
```

```
    'Todas las medidas en centésimas de milímetro
    With oLineaGuion
          .Style = 0
          .Dots = 3
          .DotLen = 500
          .Dashes = 2
          .DashLen = 2000
          .Distance = 250
    End With
    oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
    oForma = oPaginaDibujo.getByIndex(0)
    oForma.LineStyle = com.sun.star.drawing.LineStyle.DASH
    'Establecemos el estilo
    oForma.LineDash = oLineaGuion

End Sub
```

Observa que también aquí tenemos un estilo (*Style*), de acuerdo a la enumeración.

com.sun.star.drawing.DashStyle	*Valor*	*Valor en Interfaz*
com.sun.star.drawing.DashStyle.RECT	0	Rectángulo
com.sun.star.drawing.DashStyle.ROUND	1	Punto
com.sun.star.drawing.DashStyle.RECTRELATIVE	2	Rectángulo, relativo a la longitud de la línea
com.sun.star.drawing.DashStyle.ROUNDRELATIVE	3	Punto, relativo a la longitud de la línea

Para que notes las diferencias de estos estilos, establece la línea en un ancho bastante visible y nota como cambia.

6.10.2.2 Principales propiedades de relleno

Ahora veamos las principales propiedades de relleno. La primera es, sin relleno.

```
Sub AutoFormas9()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el relleno en invisible
   oForma.FillStyle = 0

End Sub
```

El estilo de fondo (FillStyle), esta determinado por la enumeración.

com.sun.star.drawing.FillStyle	*Valor*	*Valor en Interfaz*
com.sun.star.drawing.FillStyle.NONE	0	Invisible
com.sun.star.drawing.FillStyle.SOLID	1	Color
com.sun.star.drawing.FillStyle.GRADIENT	2	Gradiente
com.sun.star.drawing.FillStyle.HATCH	3	Trama
com.sun.star.drawing.FillStyle.BITMAP	4	Bitmap (Imagen)

Establecemos un color aleatorio en la primer forma de la hoja.

```
Sub AutoFormas10()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el relleno en color
   oForma.FillStyle = com.sun.star.drawing.FillStyle.SOLID
   'Un color aleatorio
   oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )

End Sub
```

Puedes establecer el porcentaje (0 a 100) de transparencia, un valor de 100, será similar a establecer el estilo de fondo en invisible (*NONE*), la diferencia será que si lo estableces con el estilo, el fondo se “quita” y puedes seleccionar las celdas que estén debajo de la forma, si lo haces con la transparencia, al dar clic dentro de la forma, seleccionas la forma.

```
Sub AutoFormas11()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el relleno en color
   oForma.FillStyle = com.sun.star.drawing.FillStyle.SOLID
   'Un color aleatorio
   oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   'La transparencia del color
   oForma.FillTransparence = 50

End Sub
```

Para establecer el fondo en un gradiente, usamos.

```
Sub AutoFormas12()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el relleno en gradiente
   oForma.FillStyle = com.sun.star.drawing.FillStyle.GRADIENT
   'El nombre del gradiente
   oForma.FillGradientName = "Rectangular red/white"

End Sub
```

El nombre es el mismo usado por la interfaz del usuario pero en ingles y debe ser exacto, si no, te dará un error en tiempo de ejecución, los gradientes predeterminados si no los ha cambiado el usuario son:

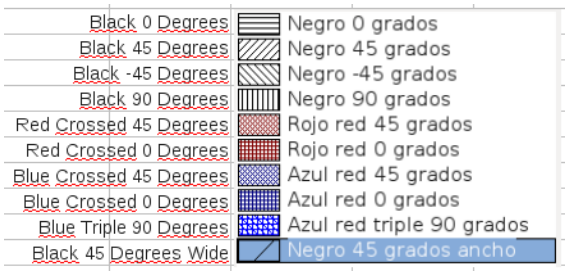
Con una matriz para los nombres, es más sencillo establecerlos.

```
Sub AutoFormas13()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim mGradiente()

   'Matriz con los nombres de los gradientes
   mGradiente = Array("Gradient 1","Gradient 2","Gradient 3","Gradient 4","Gradient 5","Gradient
6","Linear blue/white","Radial green/black","Rectangular red/white","Square yellow/white","Linear
magenta/green","Linear yellow/brown","Radial red/yellow","Ellipsoid blue grey/light blue","Axial light
red/white")
   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el relleno en gradiente
   oForma.FillStyle = com.sun.star.drawing.FillStyle.GRADIENT
   'El nombre del gradiente
   oForma.FillGradientName = mGradiente(4)

End Sub
```

Cambiamos de tipo de fondo y establecemos el estilo en trama (*HATCH*).

```
Sub AutoFormas14()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el relleno en trama
   oForma.FillStyle = com.sun.star.drawing.FillStyle.HATCH
   'El nombre de la trama
   oForma.FillHatchName = "Black 45 Degrees Wide"

End Sub
```

Al igual que con los nombres de los gradientes, estos deben establecerse exactamente como se muestran a continuación y, también, son susceptibles de ser modificados por el usuario desde la interfaz del usuario.




```
Sub AutoFormas15()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim mTramas()

   'Matriz con los nombres de las tramas predeterminadas
   mTramas = Array("Black 0 Degrees","Black 45 Degrees","Black -45 Degrees","Black 90 Degrees","Red
Crossed 45 Degrees","Red Crossed 0 Degrees","Blue Crossed 45 Degrees","Blue Crossed 0 Degrees","Blue
Triple 90 Degrees","Black 45 Degrees Wide")
   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el relleno en trama
   oForma.FillStyle = com.sun.star.drawing.FillStyle.HATCH
   'El nombre de la trama
   oForma.FillHatchName = mTramas(5)

End Sub
```

Puedes combinar el uso de una trama con un fondo de color, procura establecer un color de fondo claro, para que se distinga la trama.

```
Sub AutoFormas16()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el relleno en trama
   oForma.FillStyle = com.sun.star.drawing.FillStyle.HATCH
   'El nombre de la trama
   oForma.FillHatchName = "Black 45 Degrees Wide"
   'Tambien con color
   oForma.FillBackground = True
   oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )

End Sub
```

Para establecer una imagen como fondo de una forma, tienes que cambiar el estilo y establecer el nombre de la imagen, que como en los demás estilos, debe estar escrito correctamente y puede ser cambiado por el usuario desde la interfaz.

```
Sub AutoFormas17()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim mImagen()
```

```
   mImagen =
Array("Empty","Sky","Aqua","Coarse","Space","Metal","Wet","Marble","Linen","Stone","Space
Metal","Pebbles","Wall","Red Wall","Pattern","Leaves","Lawn
Artificial","Daisy","Orange","Fiery","Roses")
   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el relleno en Imagen
   oForma.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
   'El nombre de la imagen
   oForma.FillBitmapName = mImagen(10)

End Sub
```

Estos nombres corresponden a.

Puedes establecer una imagen, y al mismo tiempo el nivel de transparencia.

```
Sub AutoFormas18()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim mImagen()

   mImagen =
Array("Empty","Sky","Aqua","Coarse","Space","Metal","Wet","Marble","Linen","Stone","Space
Metal","Pebbles","Wall","Red Wall","Pattern","Leaves","Lawn
Artificial","Daisy","Orange","Fiery","Roses")
   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos el relleno en Imagen
   oForma.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
   'El nombre de la imagen
   oForma.FillBitmapName = mImagen(15)
   oForma.FillTransparence = 50

End Sub
```

Puedes establecer la imagen, desde un archivo de imagen.

```
Sub AutoFormas19()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   oForma.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
   'El nombre de la imagen, tomado de un archivo
   oForma.FillBitmapURL = ConvertToURL( "/home/mau/fondo.jpg" )
   'Es modo con que se muestra
   oForma.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT

End Sub
```

El modo de imagen, esta determinado por la enumeración.

com.sun.star.drawing.BitmapMode	*Valor*	*Valor en Interfaz*
com.sun.star.drawing.BitmapMode.REPEAT	0	Repetir
com.sun.star.drawing.BitmapMode.STRETCH	1	Ajustar
com.sun.star.drawing.BitmapMode.NO_REPEAT	2	No repetir

La imagen no necesariamente tiene que estar en tu equipo, puede estar al otro lado del mundo, si la ruta no existe no te dará error pero obvio, no te mostrará la imagen.

```
Sub AutoFormas20()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   oForma.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
   'El nombre de la imagen, tomado de un archivo de internet
   oForma.FillBitmapURL = ConvertToURL( "http://www.universolibre.org/archivos/favicon.png" )
End Sub
```

6.10.2.3 Principales propiedades de sombra

Ahora veamos las principales propiedades de la sombra de una forma.

```
Sub AutoFormas21()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos que queremos mostrar la sombra
   oForma.Shadow = True
   'Establecemos el color de la sombra
   oForma.ShadowColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   'Establecemos el nivel de transparencia de la sombra
   oForma.ShadowTransparence = 50

End Sub
```

Si no quieres mostrarla solo establece la sombra (*Shadow*) en falso (*False*), el color y la transparencia tienen las mismas particularidades ya vistas. Puedes posicionar la imagen, en

relación a la forma, donde quieras, por ejemplo, para establecer la sombra a 3 milímetros de la forma, usamos.

```
Sub AutoFormas22()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   'Establecemos que queremos mostrar la sombra
   oForma.Shadow = True
   'Establecemos el color de la sombra
   oForma.ShadowColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   'Establecemos la distancia X - Y de la sombra en centésimas de milímetro
   oForma.ShadowXDistance = 300
   oForma.ShadowYDistance = 300

End Sub
```

Si establecemos la posición (*ShadowXDistance y ShadowYDistance*) en 0, la sombra queda “detrás” de la forma y es “como” si la ocultaras, para establecer la forma en otra posición de la forma, puedes establecer estas propiedades en valores negativos, por ejemplo, para posicionarla en el extremo inferior izquierdo, estableces los valores en.

```
Sub AutoFormas23()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   oForma.Shadow = True
   oForma.ShadowColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   'Establecemos la distancia X - Y de la sombra en centésimas de milímetro
   oForma.ShadowXDistance = -500
   oForma.ShadowYDistance = 300

End Sub
```

6.10.2.4 Otras propiedades de las autoformas

Podemos establecer si la forma se va a imprimir o no, si se puede mover o no y si se puede cambiar el tamaño o no, con las siguientes propiedades.

```
Sub AutoFormas24()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)

   'Establecemos que no se imprima la forma
   oForma.Printable = False
   'Que no se pueda mover
   oForma.MoveProtect = True
   'Que no cambie el tamaño
   oForma.SizeProtect = True

End Sub
```

Podemos establecer el ángulo de rotación de la forma. La unidad de este valor son centésimas de grado y el sentido de rotación es inverso al giro de las manecillas del reloj.

```
Sub AutoFormas25()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)

   'Establecemos el ángulo de rotación en 45º
   'unidades en centésimas de grado
   oForma.RotateAngle = 4500

End Sub
```

También podemos modificar la inclinación de la forma, la unidad también son centésimas de grado, pero no todas las formas lo soportan, este valor solo puede estar comprendido entre 0º y 89º.

```
Sub AutoFormas26()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)

   'Establecemos el ángulo de inclinación
   oForma.ShearAngle = 1500

End Sub
```

También podemos cambiar el orden de las formas, es decir, quien esta delante de quien, esta propiedad, algunas formas especiales como rombos o corazones no la soportan, el siguiente ejemplo intercambia la posición de las dos primeras formas, asegurate de tener al menos dos formas en tu hoja y de preferencia formas estándar, rectángulos o círculos para que veas el efecto.

```
Sub AutoFormas27()
Dim oPaginaDibujo As Object
Dim oForma1 As Object
Dim oForma2 As Object
Dim Pos1 As Integer
Dim Pos2 As Integer

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma1 = oPaginaDibujo.getByIndex(0)
   oForma2 = oPaginaDibujo.getByIndex(1)

   Pos1 = oForma1.ZOrder
   Pos2 = oForma2.ZOrder

   oForma1.ZOrder = Pos2
   oForma2.ZOrder = Pos1

End Sub
```

Puedes cambiar el nombre de la forma, es importante que lo establezcas para que puedas verlo en el navegador (F5) de contenido del archivo.

```
Sub AutoFormas28()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)

   'Establecemos el nombre de la forma
   oForma.Name = "Circulo Azul"

End Sub
```

6.10.2.5 Agrupando y desagrupando formas

Para agrupar formas, usamos el siguiente código.

```
Sub AutoFormas29()
Dim oPaginaDibujo As Object
Dim oGrupoFormas As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oGrupoFormas = createUnoService("com.sun.star.drawing.ShapeCollection")
   'Necesitamos al menos dos formas para agrupar
   If oPaginaDibujo.getCount >= 2 Then
         'Agregamos las dos primeras
         oGrupoFormas.add( oPaginaDibujo.getByIndex(0) )
         oGrupoFormas.add( oPaginaDibujo.getByIndex(1) )
         'Las agrupamos
         oPaginaDibujo.group( oGrupoFormas )
   Else
         MsgBox "Agrega más formas para poder agrupar"
   End If

End Sub
```

Si el elemento a agregar no existe te dará un error en tiempo de ejecución. En un grupo puedes editar cada elemento que lo contiene de la siguiente manera.

```
Sub AutoFormas30()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim co1 As Integer

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   'Accedemos a la primer forma de la hoja
   oForma = oPaginaDibujo.getByIndex(0)
   'Checamos que sea un grupo
   If oForma.supportsService("com.sun.star.drawing.GroupShape") Then
         'Recorremos todas las formas que contiene
         For co1 = 0 To oForma.getCount - 1
               'Cambiamos el color y el tamaño de cada forma aleatoriamente
               oForma.getByindex(co1).FillColor =  RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
               Call CambiaTam( oForma.getByindex(co1), Rnd()*10000+1000, Rnd()*5000+1000)
         Next
   End If

End Sub
```

El ejemplo anterior, te dará un error en un caso determinado, tu tarea es averiguar en cual, puedes deducirlo a partir de lo que se comenta en el ejemplo 32 y mejorar este ejemplo para que funcione en todos los casos.

Para desagrupar un grupo, usamos el siguiente código.

```
Sub AutoFormas31()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = oPaginaDibujo.getByIndex(0)
   If oForma.supportsService("com.sun.star.drawing.GroupShape") Then
         'Si es un grupo, lo desagrupamos
         oPaginaDibujo.ungroup( oForma )
   End If

End Sub
```

Toma en cuenta que cuando desagrupas, el número de elementos que contenía el grupo, se suma a la cuenta de formas de la página de dibujo, por eso, este número varía en función de agrupar o desagrupar formas, toma en cuenta esto cuando trates de recorrer todas las formas de la página de dibujo, el siguiente ejemplo, te desagrupa “todos” los grupos existentes.

```
Sub AutoFormas32()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim co1 As Long
Dim bHayGrupo As Boolean

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   If oPaginaDibujo.getCount > 0 Then
         Do
               oForma = oPaginaDibujo.getByIndex(co1)
               If oForma.supportsService("com.sun.star.drawing.GroupShape") Then
                     oPaginaDibujo.ungroup( oForma )
                     'Reiniciamos la cuenta para empezar de cero
                     co1 = 0
               Else
                     co1 = co1 + 1
               End If
         'Cuando co1 = número de formas, significa que recorrió todas
         'las formas sin encontrar más grupos, por lo que salimos.
         Loop While co1 < oPaginaDibujo.getCount
   End If

End Sub
```

6.10.2.6 Trabajando con FontWork

Para insertar un texto con FontWork, usamos.

```
Sub FontWork1()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim mProp(0) As New com.sun.star.beans.PropertyValue
Dim mOpc(1) As New com.sun.star.beans.PropertyValue
```

```
    oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
    'Agregamos una forma personalizada (CustomShape)
    oForma = ThisComponent.createInstance("com.sun.star.drawing.CustomShape")

    'Cambiamos algunas propiedades conocidas
    Call CambiaTam( oForma, 15000, 4000 )
    oForma.FillStyle = com.sun.star.drawing.FillStyle.SOLID
    oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )

    oPaginaDibujo.add( oForma )
    'Establecemos el texto de la forma
    oForma.String = "OpenOffice.org Basic"

    'Esta propiedad es importante, le decimos que use una ruta para el texto
    'que es lo que le da su particularidad a FontWork
    mProp(0).Name = "TextPath"
    mProp(0).Value = True

    'Establecemos el tipo de FontWork
    mOpc(0).Name = "Type"
    mOpc(0).Value = "fontwork-wave"
    'Establecemos las propiedades de la ruta del texto
    mOpc(1).Name = "TextPath"
    mOpc(1).value = mProp()
    'Aplicamos estas propiedades personalizadas
    oForma.setPropertyValue("CustomShapeGeometry", mOpc() )

End Sub
```

La mayoría de las propiedades con las formas FontWork, se establecen como una matriz de propiedades, el tipo (*Type*) de la forma, determinará el estilo del FontWork insertado, este valor, que es una cadena (string), debe estar “exactamente” escrito, si no, no te dará ningún error pero la forma no se insertará, tienes cuarenta estilos para escoger, si combinas el tamaño, el color y estilo de línea, el color y estilo de relleno, tienes muchas, muchas posibilidades para personalizar las formas. En el siguiente ejemplo, insertamos una forma, pero seleccionamos aleatoriamente el tipo del FontWork.

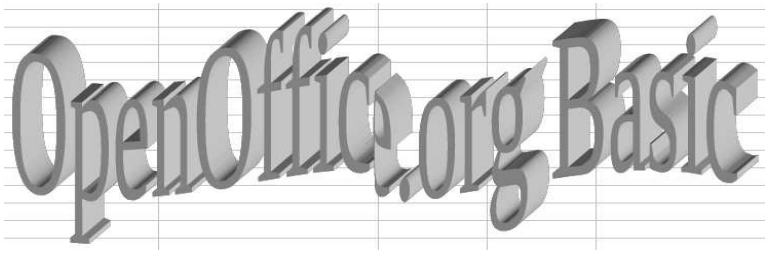
```
Sub FontWork2()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim mProp(0) As New com.sun.star.beans.PropertyValue
Dim mOpc(1) As New com.sun.star.beans.PropertyValue
Dim mTipoFW()

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = ThisComponent.createInstance("com.sun.star.drawing.CustomShape")

   mTipoFW = Array("fontwork-plain-text","fontwork-wave","fontwork-inflate","fontwork-
stop","fontwork-curve-up","fontwork-curve-down","fontwork-triangle-up","fontwork-triangle-
down","fontwork-fade-right","fontwork-fade-left","fontwork-fade-up","fontwork-fade-down","fontwork-
slant-up","fontwork-slant-down","fontwork-fade-up-and-right","fontwork-fade-up-and-left","fontwork-
chevron-up","fontwork-chevron-down","fontwork-arch-up-curve","fontwork-arch-down-curve","fontwork-
arch-left-curve","fontwork-arch-right-curve","fontwork-circle-curve","fontwork-open-circle-
curve","fontwork-arch-up-pour","fontwork-arch-down-pour","fontwork-arch-left-pour","fontwork-arch-
right-pour","fontwork-circle-pour","fontwork-open-circle-pour","mso-spt142","mso-spt143","mso-
spt157","mso-spt158","mso-spt159","mso-spt161","mso-spt162","mso-spt163","mso-spt164","mso-
spt165","mso-spt166","mso-spt167","mso-spt174","mso-spt175")

   Call CambiaTam( oForma, 15000, 4000 )
   oForma.FillStyle = com.sun.star.drawing.FillStyle.SOLID
   oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
```

```
    oPaginaDibujo.add( oForma )
    oForma.String = "OpenOffice.org Basic"

    mProp(0).Name = "TextPath"
    mProp(0).Value = True

    'Establecemos el tipo de FontWork aleatoriamente
    mOpc(0).Name = "Type"
    mOpc(0).Value = mTipoFW( Rnd()*UBound(mTipoFW) )
    mOpc(1).Name = "TextPath"
    mOpc(1).value = mProp()
    oForma.setPropertyValue("CustomShapeGeometry", mOpc() )

End Sub
```

En el siguiente ejemplo, insertamos una forma FontWork en 3D.

```
Sub FontWork3()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim mProp(0) As New com.sun.star.beans.PropertyValue
Dim mPropEx(0) As New com.sun.star.beans.PropertyValue
Dim mOpc(2) As New com.sun.star.beans.PropertyValue
Dim mTipoFW()

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = ThisComponent.createInstance("com.sun.star.drawing.CustomShape")

   mTipoFW = Array("fontwork-plain-text","fontwork-wave","fontwork-inflate","fontwork-
stop","fontwork-curve-up","fontwork-curve-down","fontwork-triangle-up","fontwork-triangle-
down","fontwork-fade-right","fontwork-fade-left","fontwork-fade-up","fontwork-fade-down","fontwork-
slant-up","fontwork-slant-down","fontwork-fade-up-and-right","fontwork-fade-up-and-left","fontwork-
chevron-up","fontwork-chevron-down","fontwork-arch-up-curve","fontwork-arch-down-curve","fontwork-
arch-left-curve","fontwork-arch-right-curve","fontwork-circle-curve","fontwork-open-circle-
curve","fontwork-arch-up-pour","fontwork-arch-down-pour","fontwork-arch-left-pour","fontwork-arch-
right-pour","fontwork-circle-pour","fontwork-open-circle-pour","mso-spt142","mso-spt143","mso-
spt157","mso-spt158","mso-spt159","mso-spt161","mso-spt162","mso-spt163","mso-spt164","mso-
spt165","mso-spt166","mso-spt167","mso-spt174","mso-spt175")

   Call CambiaTam( oForma, 15000, 4000 )
   oForma.FillStyle = com.sun.star.drawing.FillStyle.SOLID
   oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )

   oPaginaDibujo.add( oForma )
   oForma.String = "OpenOffice.org Basic"

   mProp(0).Name = "TextPath"
   mProp(0).Value = True
   mPropEx(0).Name = "Extrusion"
   mPropEx(0).Value = True

   'Establecemos el tipo de FontWork aleatoriamente
   mOpc(0).Name = "Type"
   mOpc(0).Value = mTipoFW( CInt(Rnd()*UBound(mTipoFW)) )
   mOpc(1).Name = "TextPath"
   mOpc(1).value = mProp()
   mOpc(2).Name = "Extrusion"
   mOpc(2).value = mPropEx()
   oForma.setPropertyValue("CustomShapeGeometry", mOpc() )

End Sub
```

Y mira que linda quedo, claro, te confieso que hice varios intentos hasta que salio esta que me gusto.



Tanto las propiedades de la ruta del texto (*TextPath*), como de la configuración en 3D (*Extrusion*), tienes varias propiedades más para personalizarse completamente, describirlas todas sale del propósito de este tema, pero esperamos abordarlas a profundidad en Draw.

6.10.2.7 Propiedades particulares de algunas formas

Los rectángulos y los marcos de texto, son formas muy similares, los dos soportan casi las mismas propiedades, por ejemplo, establecer el radio de las esquinas y personalizar el comportamiento del texto que contiene.

```
Sub AutoFormasEspeciales1()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   'Creamos un marco de texto
   oForma = ThisComponent.createInstance("com.sun.star.drawing.TextShape")
   Call CambiaTam( oForma, 10000, 5000 )
   oPaginaDibujo.add( oForma )
   oForma.setString("Marco con texto" & Chr(13) & "Otra línea")
   oForma.FillStyle = com.sun.star.drawing.FillStyle.SOLID
   oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   'Establecemos el radio de las esquinas
   oForma.CornerRadius = 1000
   'Establecemos que el texto se adapte al tamaño de la forma
   oForma.TextFitToSize = 1
End Sub
```

Reemplaza TextShape por RectangleShape y veras que sigue funcionando.

Las líneas tienes propiedades únicas, como la posibilidad de establecer el inicio y el final de forma diferente. El siguiente ejemplo agrega una línea ¿de?, tu tarea es decirme que distancia tiene la línea.

```
Sub AutoFormasEspeciales2()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   'Creamos una línea
   oForma = ThisComponent.createInstance("com.sun.star.drawing.LineShape")
   Call CambiaTam( oForma, 10000, 5000 )
   oPaginaDibujo.add( oForma )
   oForma.LineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   oForma.LineWidth = 200

End Sub
```

En el siguiente ejemplo, agregamos una línea vertical y otra horizontal.

```
Sub AutoFormasEspeciales3()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   'Creamos una línea horizontal
   oForma = ThisComponent.createInstance("com.sun.star.drawing.LineShape")
   Call CambiaTam( oForma, 5000, 0 )
   oPaginaDibujo.add( oForma )
   oForma.LineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   oForma.LineWidth = 200

   'Ahora una vertical
   oForma = ThisComponent.createInstance("com.sun.star.drawing.LineShape")
   Call CambiaTam( oForma, 0, 5000 )
   oPaginaDibujo.add( oForma )
   oForma.LineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   oForma.LineWidth = 200

End Sub
```

Observa como hemos logrado dibujar una línea horizontal, tan solo pasándole el valor X para cambiar el tamaño, como valor Y establecemos 0, y, lo contrario para la línea vertical.

En el siguiente ejemplo, establecemos el inicio y el final de la línea, con un cuadro a 45º, lo que te permite acotar otros elementos.

```
Sub AutoFormasEspeciales4()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = ThisComponent.createInstance("com.sun.star.drawing.LineShape")
   Call CambiaTam( oForma, 10000, 0 )
   oPaginaDibujo.add( oForma )
   oForma.LineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   oForma.LineWidth = 300
   'Establecemos el ancho de inicio y final de línea
   oForma.LineStartWidth = 1000
   oForma.LineEndWidth = 1000
   'Establecemos el tipo de inicio y fin de línea
   oForma.LineStartName = "Square 45"
   oForma.LineEndName = "Square 45"

End Sub
```

Ahora, establecemos flechas como final de línea.

```
Sub AutoFormasEspeciales5()
Dim oPaginaDibujo As Object
Dim oForma As Object

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = ThisComponent.createInstance("com.sun.star.drawing.LineShape")
   Call CambiaTam( oForma, 10000, 0 )
   oPaginaDibujo.add( oForma )
   oForma.LineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   oForma.LineWidth = 300
   'Establecemos el ancho de inicio y final de línea
```

```
    oForma.LineStartWidth = 1000
    oForma.LineEndWidth = 1000
    'Establecemos el tipo de inicio y fin de línea
    oForma.LineStartName = "Arrow"
    oForma.LineEndName = "Arrow"

End Sub
```

Para terminar este tema, veamos como insertar otros tipos de formas, solo algunos.

```
Sub AutoFormasEspeciales6()
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim mOpc(0) As New com.sun.star.beans.PropertyValue

    oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
    oForma = ThisComponent.createInstance("com.sun.star.drawing.CustomShape")
    Call CambiaTam( oForma, 5000, 5000 )
    oPaginaDibujo.add( oForma )
    oForma.LineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
    oForma.LineWidth = 300
    oForma.FillStyle = com.sun.star.drawing.FillStyle.SOLID
    oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )

    'Agregamos un hexágono
    mOpc(0).Name = "Type"
    mOpc(0).Value = "hexagon"
    oForma.setPropertyValue("CustomShapeGeometry", mOpc() )

    oForma = ThisComponent.createInstance("com.sun.star.drawing.CustomShape")
    Call CambiaTam( oForma, 5000, 5000 )
    oPaginaDibujo.add( oForma )
    oForma.LineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
    oForma.LineWidth = 300
    oForma.FillStyle = com.sun.star.drawing.FillStyle.SOLID
    oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )

    'Agregamos un corazón
    mOpc(0).Name = "Type"
    mOpc(0).Value = "heart"
    oForma.setPropertyValue("CustomShapeGeometry", mOpc() )

    oForma = ThisComponent.createInstance("com.sun.star.drawing.CustomShape")
    Call CambiaTam( oForma, 5000, 5000 )
    oPaginaDibujo.add( oForma )
    oForma.LineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
    oForma.LineWidth = 300
    oForma.FillStyle = com.sun.star.drawing.FillStyle.SOLID
    oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )

    'Agregamos el símbolo de ordenar en un diagrama de flujo
    mOpc(0).Name = "Type"
    mOpc(0).Value = "flowchart-sort"
    oForma.setPropertyValue("CustomShapeGeometry", mOpc() )

End Sub
```

Como ya lo notaste, hay que indicarle el nombre de la forma especial que queremos agregar, la cantidad de formas es grande, compruebalo tu mismo con el siguiente código donde agregamos una forma aleatoria de las diferentes doscientas un posibles.

```
Sub AutoFormasEspeciales7()
```

```
Dim oPaginaDibujo As Object
Dim oForma As Object
Dim mOpc(0) As New com.sun.star.beans.PropertyValue
Dim mTipoFormas

   mTipoFormas = Array("rectangle", "round-rectangle", "ellipse", "diamond", "isosceles-triangle",
"right-triangle", "parallelogram", "trapezoid", "hexagon", "octagon", "cross", "star5", "right-
arrow", "mso-spt14", "pentagon-right", "cube", "mso-spt17", "mso-spt18", "mso-spt19", "mso-spt20",
"mso-spt21", "can", "ring", "mso-spt24", "mso-spt25", "mso-spt26", "mso-spt27", "mso-spt28", "mso-
spt29", "mso-spt30", "mso-spt31", "mso-spt32", "mso-spt33", "mso-spt34", "mso-spt35", "mso-spt36",
"mso-spt37", "mso-spt38", "mso-spt39", "mso-spt40", "mso-spt41", "mso-spt42", "mso-spt43", "mso-
spt44", "mso-spt45", "mso-spt46", "line-callout-1", "line-callout-2", "mso-spt49", "mso-spt50", "mso-
spt51", "mso-spt52", "mso-spt53", "mso-spt54", "chevron", "pentagon", "forbidden", "star8", "mso-
spt59", "mso-spt60", "rectangular-callout", "round-rectangular-callout", "round-callout", "mso-
spt64", "paper", "left-arrow", "down-arrow", "up-arrow", "left-right-arrow", "up-down-arrow", "mso-
spt71", "bang", "lightning", "heart", "mso-spt75", "quad-arrow", "left-arrow-callout", "right-arrow-
callout", "up-arrow-callout", "down-arrow-callout", "left-right-arrow-callout", "up-down-arrow-
callout", "quad-arrow-callout", "quad-bevel", "left-bracket", "right-bracket", "left-brace", "right-
brace", "mso-spt89", "mso-spt90", "mso-spt91", "star24", "striped-right-arrow", "notched-right-
arrow", "block-arc", "smiley", "vertical-scroll", "horizontal-scroll", "circular-arrow", "mso-
spt100", "mso-spt101", "mso-spt102", "mso-spt103", "mso-spt104", "mso-spt105", "cloud-callout", "mso-
spt107", "mso-spt108", "flowchart-process", "flowchart-decision", "flowchart-data", "flowchart-
predefined-process", "flowchart-internal-storage", "flowchart-document", "flowchart-multidocument",
"flowchart-terminator", "flowchart-preparation", "flowchart-manual-input", "flowchart-manual-
operation", "flowchart-connector", "flowchart-card", "flowchart-punched-tape", "flowchart-summing-
junction", "flowchart-or", "flowchart-collate", "flowchart-sort", "flowchart-extract", "flowchart-
merge", "mso-spt129", "flowchart-stored-data", "flowchart-sequential-access", "flowchart-magnetic-
disk", "flowchart-direct-access-storage", "flowchart-display", "flowchart-delay", "fontwork-plain-
text", "fontwork-stop", "fontwork-triangle-up", "fontwork-triangle-down", "fontwork-chevron-up",
"fontwork-chevron-down", "mso-spt142", "mso-spt143", "fontwork-arch-up-curve", "fontwork-arch-down-
curve", "fontwork-circle-curve", "fontwork-open-circle-curve", "fontwork-arch-up-pour", "fontwork-
arch-down-pour", "fontwork-circle-pour", "fontwork-open-circle-pour", "fontwork-curve-up", "fontwork-
curve-down", "fontwork-fade-up-and-right", "fontwork-fade-up-and-left", "fontwork-wave", "mso-
spt157", "mso-spt158", "mso-spt159", "fontwork-inflate", "mso-spt161", "mso-spt162", "mso-spt163",
"mso-spt164", "mso-spt165", "mso-spt166", "mso-spt167", "fontwork-fade-right", "fontwork-fade-left",
"fontwork-fade-up", "fontwork-fade-down", "fontwork-slant-up", "fontwork-slant-down", "mso-spt174",
"mso-spt175", "flowchart-alternate-process", "flowchart-off-page-connector", "mso-spt178", "mso-
spt179", "mso-spt180", "line-callout-3", "mso-spt182", "sun", "moon", "bracket-pair", "brace-pair",
"star4", "mso-spt188", "mso-spt189", "mso-spt190", "mso-spt191", "mso-spt192", "mso-spt193", "mso-
spt194", "mso-spt195", "mso-spt196", "mso-spt197", "mso-spt198", "mso-spt199", "mso-spt200", "mso-
spt201", "mso-spt202" )

   oPaginaDibujo = ThisComponent.getCurrentController.getActiveSheet.getDrawPage()
   oForma = ThisComponent.createInstance("com.sun.star.drawing.CustomShape")
   Call CambiaTam( oForma, 5000, 5000 )
   oPaginaDibujo.add( oForma )
   oForma.LineColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   oForma.LineWidth = 300
   oForma.FillStyle = com.sun.star.drawing.FillStyle.SOLID
   oForma.FillColor = RGB( Rnd()*255,Rnd()*255,Rnd()*255 )
   'Agregamos una forma aleatoria
   mOpc(0).Name = "Type"
   mOpc(0).Value = mTipoFormas( CInt(Rnd()*UBound(mTipoFormas)) )
   oForma.setPropertyValue("CustomShapeGeometry", mOpc() )

End Sub
```

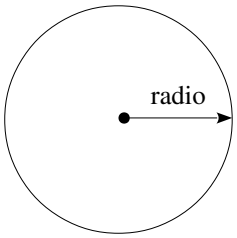
6.11 Funciones personalizadas

La posibilidad de implementar nuestras propias funciones, es una de las características más ricas de una hoja de calculo, para todo este capitulo, asumo que no tienes

problemas con la implementación de formulas y funciones en Calc, es muy útil, que tengas muy claro, que las funciones siempre devuelven un valor, aun, y cuando este sea un error, que estas pueden o no necesitar argumentos, que los argumentos son de un tipo y el valor devuelto puede ser de otro, por ejemplo, supongamos las triviales formulas para obtener el diámetro, el perímetro y el área de un circulo

Donde:

r = radio
D = Diámetro
P = Perímetro
A = Área



$$D = 2 * r$$

$$P = \Pi * D$$

$$A = \Pi * r^2$$

En Calc, suponiendo que el radio este en la celda A2 las formulas serían.

Diámetro = A2 * 2
Perímetro = PI() * A2 * 2
Área = PI() * POTENCIA(A2;2)

En OOo Basic, estas funciones podrían quedar así:

```
Option Explicit

'Función para obtener el diámetro de un circulo
Function DiametroCirculo( Radio As Single ) As Double

   DiametroCirculo = Radio * 2

End Function


'Función para obtener el perímetro de un circulo
Function PerimetroCirculo( Radio As Single ) As Double
Const PI As Single = 3.1416

   PerimetroCirculo = Radio * 2 * PI

End Function


'Función para obtener el área de un circulo
Function AreaCirculo( Radio As Single ) As Double
Const PI As Single = 3.1416

   AreaCirculo = PI * Radio ^ 2

End Function
```

Estas funciones, se usan exactamente igual que cualquier otra función de Calc, es decir, escribiendo su nombre correctamente y pasándole los argumentos correctos. Observa la siguiente imagen, donde estamos calculando estos valores, por los dos métodos, con formulas y funciones incorporadas de Calc, y con nuestras funciones personalizadas.

La forma como establecemos los argumentos y el valor devuelto de la función, no te son desconocidos, los hemos venido usando a lo largo de los temas vistos, la diferencia, es que, ahora, solo “llamábamos” a las funciones desde otras macros, ahora, las estamos usando desde la hoja de calculo.

Pero las funciones personalizadas son mucho más versátiles, nota como en las tres formulas, usamos el mismo argumento, solo el *radio*, podemos crear nuestra función para que nos

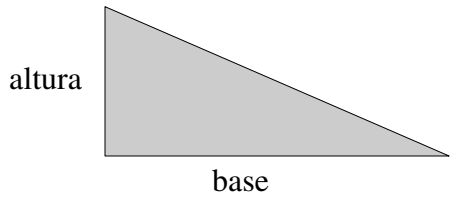
devuelva el valor deseado de los tres, simplemente pasándole un segundo argumento para indicárselo.

```
'Función para devolver un dato de un círculo
'QueDato puede tener los valores
' 1 = Diámetro
' 2 = Perímetro
' 3 = Área
Function Circulo( Radio As Single, QueDato As Byte ) As Double
Const PI As Single = 3.141592
Dim dTmp As Double

   Select Case QueDato
         Case 1
               dTmp = Radio * 2
         Case 2
               dTmp = Radio * 2 * PI
         Case 3
               dTmp = PI * Radio ^ 2
   End Select

   Circulo = dTmp

End Function
```

Cuando le pasas un argumento a una función desde una hoja de calculo (generalmente una referencia a una o varias celdas), lo que realmente toma la función es el valor de estas celdas. Las funciones personalizadas, tienen las mismas consideraciones presentes en las funciones incorporadas de Calc, principalmente, las relacionadas con el correcto paso de argumentos y el tipo de valor devuelto. Nuestro siguiente ejemplo también es muy simple, vamos a obtener el área de un triángulo, cuya formula es.

Donde:

b = Base
h = Altura
A = Área



$$A = \frac{b * h}{2}$$

Nuestra función para calcular el área, quedaría así.

```
'Función para calcular el área de un triángulo
Function AreaTriangulo( B As Single, H As Single ) As Double

   AreaTriangulo = (B * H) / 2

End Function
```

El asunto cambia bastante cuando lo que conocemos son la longitud de los lados del triángulo, para estos casos, usamos la famosa formula de Herón. Te pongo la formula y antes de que veas la respuesta, trata de hacerla tu mismo.

Donde:

$$A = \sqrt{S * (S-a) * (S-b) * (S-c)}$$

a, b y c = son los lados del triángulo
S = Semiperímetro, es decir, el perímetro entre dos
A = Área

Como función.

```
'Función para calcular el área de un triángulo
'con la formula de Herón
Function AreaTrianguloHeron( a As Single, b As Single, c As Single ) As Double
Dim S As Double

   S = ( a + b + c ) / 2
   AreaTrianguloHeron = Sqr( S*(S-a)*(S-b)*(S-c) )

End Function
```

Por supuesto, no solo puedes manejar números como argumentos, en el siguiente ejemplo, usamos una fecha como argumento, el planteamiento es el siguiente; se contrato un servicio anual, queremos mostrar en otra celda, el número de días que faltan para renovar el servicio, no importa el número de años que lleva contratado, solo queremos saber cuantos días faltan para la próxima renovación, una primera aproximación sería.

Celda A2 = Fecha de contratación
Celda B2 = Días que faltan para renovar
La formula de la columna B es:

=FECHA(SI(FECHA(AÑO(HOY());MES(A2);DÍA(A2))>HOY();AÑO(HOY());AÑO(HOY())
+1);MES(A2);DÍA(A2))-HOY()

Esta formula ya es un poco más compleja, nos pide un poco más de esfuerzo para comprenderla, pero es un buen ejemplo de lo que hay que hacer cuando queremos hacer una función personalizada, es decir, tenemos que tener muy claro que es lo que queremos obtener y que datos de origen tenemos, en este caso, solo tenemos una fecha, la fecha de contratación del servicio, ¿que es lo que haces para saber cuantos días faltan para la renovación?, cuidado, no es una pregunta ligera, recuerda que el trabajo de programador, es resolver problemas, y los problemas, entre más sencillos, son más fáciles de resolver, así que, tomate con calma la respuesta de la pregunta. En la solución de la función comentamos paso a paso la respuesta.

```
'Función para saber cuantos días faltan para renovar
Function DiasParaRenovar( FechaInicial As Date ) As Integer
Dim FechaActual As Date
Dim iDiferencia As Integer

   'Lo primero que necesitamos es tener la misma fecha inicial
   'mismo día y mes, pero con el año actual, observa como obtenemos
   'el año actual, la función Now, devuelve la fecha actual, con Year obtenemos el año
   FechaActual = DateSerial( Year(Now()), Month(FechaInicial), Day(FechaInicial) )

   'El siguiente paso, es saber si esta fecha actual, es mayor o menor al día de hoy
   If FechaActual < Now() Then
         'Si es menor, significa que la renovación ya paso, por lo que se tiene
         'que aumentar un año a la fecha actual para obtener la correcta de renovación
         FechaActual = DateSerial( Year(Now())+1, Month(FechaInicial), Day(FechaInicial) )
   End If

   'Restamos la fecha futura con el día de hoy, para saber cuantos días faltan
   'Usamos Fix para eliminar las horas y evitar el redondeo
   iDiferencia = FechaActual - Fix(Now())
   'Asignamos el resultado
   DiasParaRenovar = iDiferencia

End Function
```

Comparando, aquí es bastante notable la comodidad de una función personalizada:

=FECHA(SI(FECHA(AÑO(HOY());MES(A2);DÍA(A2))>HOY();AÑO(HOY());AÑO(HOY()) +1);MES(A2);DÍA(A2))-HOY()
=DIASPARARENOVAR(A2)

En nuestro siguiente ejemplo, tenemos como argumentos números y regresamos un texto, la tarea es mostrar el valor de un ángulo, configurado correctamente, por ejemplo: 125º 45' 35'', lo que logramos con la siguiente función.

```
'Función para dar formato a valores sexagesimales
Function AnguloFormateado( Gra As Integer, Min As Byte, Seg As Byte ) As String
Dim sTmp As String

   sTmp = Str(Gra) & "º " & Str(Min) & "' " & Str(Seg) & "''"
   AnguloFormateado = sTmp

End Function
```

Observa como estamos regresando una cadena (*String*). Tu tarea es hacer la operación inversa, a partir de la cadena, regresar cualquiera de los tres valores, grados, minutos o segundos, según el argumento pasado, te pongo el esqueleto de la función y tú la desarrollas.

```
'Función para devolver un valor de una ángulo
'El argumento QueValor, puede tomar los valores
' 1 = Devuelve los grados
' 2 = Devuelve los minutos
' 3 = Devuelve los segundos
Function DatoAngulo( Angulo As String, QueValor As Byte ) As Integer

   'Esta es tú tarea

End Function
```

Otro ejemplo donde tenemos como argumento un número (Byte) y regresamos una cadena (String), es el siguiente.

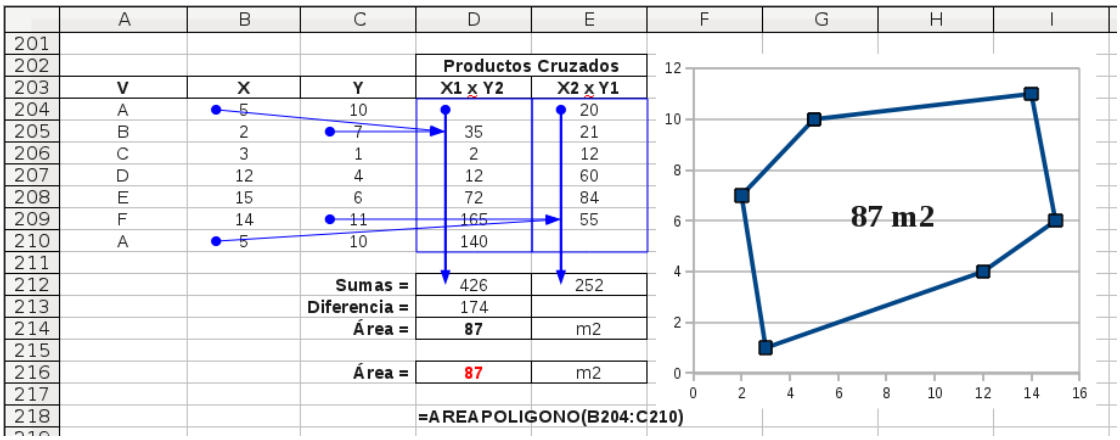
```
'Función para convertir un número en texto
Function NumeroTexto( Num As Byte ) As String
Dim sTmp As String

   Select Case Num
         Case 0 : sTmp = "Cero"
         Case 1 : sTmp = "Uno"
         Case 2 : sTmp = "Dos"
         Case 3 : sTmp = "Tres"
         Case 4 : sTmp = "Cuatro"
         Case 5 : sTmp = "Cinco"
         Case 6 : sTmp = "Seis"
         Case 7 : sTmp = "Siete"
         Case 8 : sTmp = "Ocho"
         Case 9 : sTmp = "Nueve"
   End Select
   NumeroTexto = sTmp

End Function
```

¿Y para que crees que sirve eso?..., exactamente, para hacer la famosa función que convierte números a letras, sumamente útil y muy usada en áreas administrativas, te invito a que intentes desarrollarla, dada la extensión de la misma, no la incluimos aquí, pero si en los archivos de ejemplo que acompañan a este libro.

Hasta ahora, todos los argumentos que le hemos pasado a nuestras funciones, han sido celdas individuales, pero también puedes pasarle rangos de celdas, donde, lo único que tienes que tener en cuenta, es que los valores de este rango, los tomará como una matriz de dos dimensiones, nuestra versión de la función SUMA, es la siguiente.

```
'Función para sumar los valores de un rango
Function SumarRango( Rango ) As Double
Dim dTmp As Double
Dim co1 As Long, co2 As Long

   'Iteramos en la primer dimensión de la matriz
   For co1 = LBound( Rango,1 ) To UBound( Rango,1 )
         'Iteramos en la segunda dimensión de la matriz
         For co2 = LBound( Rango,2 ) To UBound( Rango,2 )
               'Vamos sumando los valores
               dTmp = dTmp + Rango( co1, co2 )
         Next co2
   Next co1
   'Asignamos el resultado
   SumarRango = dTmp

End Function
```

Observa como no establecemos el tipo del argumento Rango, esto es por que, al ser una matriz, esta, forzosamente tiene que ser tipo variante (*Variant*). Veamos un ejemplo practico donde pasamos como argumento un rango de celdas. En topografía, se puede calcular el área de un terreno o de cualquier poligonal cerrada cualquiera, donde se cuenta con sus coordenadas, por varios métodos, uno de ellos se llama por “Productos Cruzados”, cuyo algoritmo, espero sea bastante claro en la siguiente imagen.



En la celda D214, tenemos el calculo del área, pero usando todo el desarrollo del método, en la celda D216 (en rojo), tenemos el mismo resultado, pero usando la función mostrada en la celda D218, cuyo código es el siguiente.

```
'Función para obtener el área de un polígono irregular
'por medio de coordenadas y el método de productos cruzados
Function AreaPoligono( Rango ) As Double
Dim Suma1 As Double
Dim Suma2 As Double
Dim co1 As Long

   'Calculamos las suma de los productos cruzados
   For co1 = LBound( Rango,1 ) To UBound( Rango,1 ) - 1
         Suma1 = Suma1 + Rango(co1,1) * Rango(co1+1,2)
```

```
        Suma2 = Suma2 + Rango(co1+1,1) * Rango(co1,2)
    Next co1
    'Asignamos el resultado
    AreaPoligono = Abs(Suma1 - Suma2) / 2

End Function
```

Muy importante, observa que para acceder a la segunda dimensión de la matriz, estamos empezando en 1, no se cual sea la razón de que al acceder a los valores de un rango de celdas por medio de una función, el limite inferior sea siempre 1 (por ahora), es la única excepción que me he encontrado con el uso de matrices, recuerdalo.

6.12 Configuración global de Calc

En este capitulo, veremos algunas opciones que afectan a la configuración global de Calc, por ejemplo lo que queremos ver y lo que no.

```
Sub ConfigurarCalc1()
Dim oDoc As Object
Dim oCC As Object

   oDoc = ThisComponent
   oCC = oDoc.getCurrentController()

   'Mostramos las formulas en vez de su resultado
   'el valor "normal" es falso (False)
   oCC.showFormulas = True

   'Ocultamos los valores cero
   'el valor normal es verdadero (True)
   oCC.showZeroValues = False

   'Podemos automáticamente diferenciar los valores de
   'formulas con esta propiedad, los valores los pone
   'en azul y las formulas en verde, pero ten cuidado
   'no podrás establecer el color de fuente mientras
   'esta propiedad este en verdadera (True)
   oCC.IsValueHighlightingEnabled = True

   'Esta propiedad, solo oculta el pequeño cuadro rojo
   'que nos indica que una celda tiene una nota, la nota
   'sigue existiendo, establecelo en verdadero para volver
   'a ver este pequeño cuadro
   oCC.showNotes = False

End Sub
```

También podemos mostrar u ocultar otros elementos de la hoja de calculo, en todos los ejemplos siguientes, alternamos el valor de las propiedades, es decir, si están visibles se ocultan y viceversa:

```
Sub ConfigurarCalc2()
Dim oDoc As Object
Dim oCC As Object
```

```
    oDoc = ThisComponent
    oCC = oDoc.getCurrentController()

    'Ocultamos o mostramos las barras de desplazamiento
    'tanto vertical como horizontal
    oCC.HasVerticalScrollBar = Not oCC.HasVerticalScrollBar
    oCC.HasHorizontalScrollBar = Not oCC.HasHorizontalScrollBar

    'Las etiqueas de las hojas
    oCC.HasSheetTabs = Not oCC.HasSheetTabs

    'Los encabezados de filas y columnas
    oCC.HasColumnRowHeaders = Not oCC.HasColumnRowHeaders

    'Las líneas de división de las celdas
    oCC.showGrid = Not oCC.showGrid

    'Las líneas de ayuda cuando se mueve un objeto gráfico
    oCC.showHelpLines = Not oCC.ShowHelpLines

    'El icono de anclaje cuando se selecciona un objeto gráfico
    oCC.showAnchor = Not oCC.ShowAnchor

    'Los saltos de página
    oCC.showPageBreaks = Not oCC.ShowPageBreaks

    'Los objetos e imagenes
    oCC.showObjects = Not oCC.showObjects

    'Los gráficos
    oCC.showCharts = Not oCC.showCharts

    'Las formas
    oCC.showDrawing = Not oCC.showDrawing

    'Los símbolos de esquema y agrupamiento
    oCC.IsOutlineSymbolsSet = Not oCC.IsOutlineSymbolsSet

End Sub
```

Ten cuidado con ocultar algunos elementos, por ejemplo, si ocultas las etiquetas de las hojas, el usuario puede cambiarse aun con la combinación de teclas CTRL+RePag (AvPag), los saltos de página se ocultan, pero siguen existiendo, así como los gráficos, objetos, formas y símbolos de esquema. En el siguiente ejemplo, cambiamos el color de las líneas de división de las celdas y establecemos el zoom con un porcentaje.

```
Sub ConfigurarCalc3()
Dim oDoc As Object
Dim oCC As Object

    oDoc = ThisComponent
    oCC = oDoc.getCurrentController()

    'Puedes cambiar el color de las líneas de división de celdas
    'el valor predeterminado es gris claro = 12632256
    oCC.gridColor = RGB(Rnd()*255,Rnd()*255,Rnd()*255)

    'Establecer el tipo de Zoom
    ' 1 = Ancho de página
    ' 2 = Ajustar al ancho y alto
    ' 3 = Valor
    oCC.ZoomType = 3
```

```
    'Si lo establece en 3 (valor), puedes establecer
    'el porcentaje de zoom entre 20% y 400%
    oCC.ZoomValue = 130

End Sub
```

Si estableces el porcentaje fuera de sus limites (20% a 400%), este se ajustará al más cercano. Las siguientes propiedades, solo tiene efecto sobre una sola hoja, pero me parece que son interesantes de ver. Podemos consultar y establecer la primer fila y columna visible.

```
Sub ConfigurarCalc4()
Dim oDoc As Object
Dim oCC As Object

    oDoc = ThisComponent
    oCC = oDoc.getCurrentController()

    'La primer fila y columna visible actuales
    MsgBox "Fila: " & oCC.getFirstVisibleRow & Chr(13) & "Columna: " & oCC.getFirstVisibleColumn

    'Lo interesante, es que podemos establecerlas para que el usuario
    'vea lo que queramos que vea
    oCC.setFirstVisibleRow( 50 )
    oCC.setFirstVisibleColumn( 10 )

End Sub
```

Podemos obtener el rango visible, como mostramos en el ejemplo, este rango se ve afectado por el nivel de zoom establecido en la hoja.

```
Sub ConfigurarCalc5()
Dim oDoc As Object
Dim oCC As Object
Dim oRangoVisible As Object

    oDoc = ThisComponent
    oCC = oDoc.getCurrentController()

    'Obtenemos el rango visible
    oRangoVisible = oCC.getVisibleRange()
    'Mostramos su dirección
    MsgBox DireccionRango( oRangoVisible )

    'Cambiamos el zoom
    oCC.ZoomType = 3
    oCC.ZoomValue = oCC.ZoomValue + 25

    'Volvemos a consultar el rango visible
    oRangoVisible = oCC.getVisibleRange()
    MsgBox DireccionRango( oRangoVisible )

End Sub


Function DireccionRango( DirRango As Object) As String
Dim oHA As Object
Dim sTmp As String

    oHA = ThisComponent.getCurrentController.getActiveSheet
    sTmp = oHA.getColumns.getByIndex(DirRango.StartColumn).getName() & _
                DirRango.StartRow + 1 & ":" & _
```

```
                oHA.getColumns.getByIndex(DirRango.EndColumn).getName() & _
                DirRango.EndRow + 1

    DireccionRango = sTmp

End Function
```

Podemos saber si la ventana esta dividida.

```
Sub ConfigurarCalc6()
Dim oDoc As Object
Dim oCC As Object
Dim sInfo As String

    oDoc = ThisComponent
    oCC = oDoc.getCurrentController()

    'Consultamos si la ventana esta dividida
    If oCC.getIsWindowSplit Then
          'Mostramos la información
          sInfo = "La ventana esta dividida a: " & Chr(13) & _
                      oCC.getSplitHorizontal & " pixeles de la izquierda y" & Chr(13) & _
                      oCC.getSplitVertical  & " pixeles de la parte superior" & Chr(13) & _
                      "En la columna: " &  oCC.getSplitColumn & " y en la fila: " & oCC.getSplitRow
          MsgBox sInfo
    Else
          MsgBox "La ventana no esta dividida"
    End If

End Sub
```

Por supuesto podemos establecer la división donde quieras.

```
Sub ConfigurarCalc7()
Dim oDoc As Object
Dim oCC As Object
Dim sInfo As String

    oDoc = ThisComponent
    oCC = oDoc.getCurrentController()
    'Si no esta dividida la dividimos
    If Not oCC.getIsWindowSplit Then
          'Las unidades son pixeles
          oCC.splitAtPosition( 500,300 )
    End If

End Sub
```

Si quieres establecer solo una división, establece la que no quieras en cero, por ejemplo, si solo quieres la división vertical usas.

```
          oCC.splitAtPosition( 500,0 )
```

Para la horizontal.

```
          oCC.splitAtPosition( 0,300 )
```

Para eliminar la división.

```
        oCC.splitAtPosition( 0,0 )
```

También podemos saber si la ventana esta fija.

```
Sub ConfigurarCalc8()
Dim oDoc As Object
Dim oCC As Object
Dim sInfo As String

   oDoc = ThisComponent
   oCC = oDoc.getCurrentController()
   'Consultamos si la ventana esta fija
   If oCC.hasFrozenPanes Then
         MsgBox "La ventana esta fija"
   Else
         MsgBox "La ventana no esta fija"
   End If

End Sub
```

Y establecer la posición de esta, aquí se usa la columna y fila deseada.

```
Sub ConfigurarCalc9()
Dim oDoc As Object
Dim oCC As Object
Dim sInfo As String

   oDoc = ThisComponent
   oCC = oDoc.getCurrentController()
   'Establecemos la división en la columna 5 y fila 10
   oCC.freezeAtPosition( 5, 10)

End Sub
```

Al igual que con la ventana dividida, si quiere establecer solo un sentido, establece el otro en cero, si estableces los dos argumentos en ceros, no eliminaras la división, si que la ventana se fijara en la posición actual del cursor de celda.

```
oCC.freezeAtPosition( 0, 0)
```

Si quieres quitarla, usa el mismo método visto anteriormente.

```
oCC.splitAtPosition( 0,0 )
```

Si divides la ventana por cualquiera de los dos métodos vistos, estás pueden tener 2 o cuatro partes, cada una de estas partes se llama panel, y puedes tener acceso a cada rango de cada panel, como se muestra en el siguiente ejemplo.

```
Sub ConfigurarCalc10()
Dim oDoc As Object
Dim oCC As Object
Dim oPanel As Object
Dim col As Byte
Dim sInfo As String

   oDoc = ThisComponent
   oCC = oDoc.getCurrentController()
```

```
    'La ventans debe estar fija o dividida
    If oCC.getIsWindowSplit Or oCC.hasFrozenPanes Then
          For co1 = 0 To oCC.getCount - 1
                'Obtenemos acceso a cada panel
                oPanel = oCC.getByIndex( co1 )
                'Mostramos su dirección
                MsgBox "La dirección del Panel " & co1+1 & " es: " & DireccionRango2(
oPanel.getReferredCells )
          Next co1
    End If
End Sub


Function DireccionRango2(Rango As Object) As String
Dim sTmp As String

    Select Case Rango.getImplementationName()
          Case "ScCellObj"
                sTmp = Rango.getSpreadsheet.getName() & "." & _
                            Rango.getColumns().getByIndex(0).getName() & _
                            Rango.getCellAddress.Row + 1
          Case "ScCellRangeObj", "ScCellCursorObj"
                sTmp = Rango.getSpreadsheet.getName() & "." & _
                            Rango.getColumns().getByIndex(0).getName() & _
                            Rango.getRangeAddress.StartRow + 1 & ":" & _
                            Rango.getColumns().getByIndex(Rango.getColumns().getCount()-1).getName() &
_
                            Rango.getRangeAddress.EndRow + 1
          Case "ScCellRangesObj"
                sTmp = Join( Split(Rango.getRangeAddressesAsString(),";"),Chr(13) )
    End Select

    DireccionRango2 = sTmp

End Function
```

La función anterior ya la hemos usado anteriormente, aquí la repetimos por si no la tienes a mano, y con esto damos por terminado un nuevo tema.

7 Trabajando con formularios

Los formularios nos permiten interactuar con el usuario por medio de controles que son objetos con características especiales y la capacidad de poder asociarles la ejecución de macros. El formulario y la mayoría de los controles tienen propiedades especiales para enlazarse, visualizar y controlar bases de datos, aunque se pueden usar sin ellas. El uso de formularios permite automatizar aun más, múltiples tareas, también pueden servir para ayudar al usuario con la manipulación de datos, sobre todo, a los que tienen poca experiencia con la hoja de calculo ya que nos permite convertir una hoja de Calc, en una nueva y completamente personalizada interfaz de usuario. Asumo que no tienes problemas, como usuario, con el uso de los formularios, es decir, que sabes agregar los controles a la interfaz del usuario y manipular sus propiedades en tiempo de diseño, aquí, veremos como manipular las propiedades más importantes de estos, en tiempo de ejecución, es decir, con código OOo Basic.

7.1 Formularios (Forms)

Los formularios y sus controles se colocan sobre la capa de dibujo, por lo que a través de esta hay que acceder a ellos como en el siguiente ejemplo donde mostramos los nombres de los formularios existentes en la hoja activa.

```
Sub Formularios1()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim co1 As Integer

   'El archivo desde donde se llama esta macro
   oDoc = ThisComponent
   'La hoja activa
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   'La página de dibujo
   oPaginaDibujo = oHojaActiva.getDrawPage()
   'Todos los formularios
   oFormularios = oPaginaDibujo.getForms()
   'Iteramos en cada uno mostrando su nombre
   For co1 = 0 To oFormularios.getCount() - 1
        MsgBox oFormularios.getByIndex(co1).getName()
   Next co1

End Sub
```

Como en el ejemplo anterior, puedes acceder a un formulario por su índice o por su nombre, en los dos casos el formulario “debe” de existir, debes de validarlo como en.

```
Sub Formularios2()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim iNum As Integer
Dim sNombre As String

   'El archivo desde donde se llama esta macro
   oDoc = ThisComponent
   'La hoja activa
```

```
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()
    'La página de dibujo
    oPaginaDibujo = oHojaActiva.getDrawPage()
    'Todos los formularios
    oFormularios = oPaginaDibujo.getForms()

    iNum = 1
    'Nos aseguramos de que existe el número de formulario
    If iNum < oFormularios.getCount() Then
         MsgBox oFormularios.getByIndex(iNum).getName()
    End If

    sNombre = "Directorio"
    'Nos aseguramos de que existe el nombre del formulario
    If oFormularios.hasByName(sNombre) Then
         MsgBox oFormularios.getByName(sNombre).getName()
    End If

End Sub
```

Puedes cambiar el nombre del formulario.

```
Sub Formularios3()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object

    oDoc = ThisComponent
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()
    oPaginaDibujo = oHojaActiva.getDrawPage()
    oFormularios = oPaginaDibujo.getForms()
    oFormulario = oFormularios.getByIndex(0)

    MsgBox oFormulario.getName()
    'Cambiamos el nombre del primer formulario
    oFormulario.setName("Nuevo Formulario")

End Sub
```

Puedes cambiar el origen de datos del formulario.

```
Sub Formularios4()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object

    oDoc = ThisComponent
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()
    oPaginaDibujo = oHojaActiva.getDrawPage()
    oFormularios = oPaginaDibujo.getForms()
    oFormulario = oFormularios.getByIndex(0)

    MsgBox oFormulario.DataSourceName
    'Cambiamos la fuente de datos
    oFormulario.DataSourceName = "Directorio"

End Sub
```

Aquí debes de tener algunas consideraciones importantes; los nombres de las fuentes de datos, son nombres de bases de datos registradas en OOo y ya vistas en el capitulo de Bases de datos, por lo que no las repetiremos aquí, si la fuente de datos no existe, no te dará ningún error, pero claro, no obtendrás nada de datos, ten cuidado al cambiar la fuente de datos, no solo el formulario puede estar vinculado con esta, los controles del formulario, también pueden estar vinculados a ella, por lo que perderás cualquier vinculo a los datos, lo más común, es enlazar una vez al inicio de la configuración de un formulario o para corregir desajustes realizados por los usuarios. Además de la fuente de datos puedes establecer el tipo de contenido y el contenido.

```
Sub Formularios5()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()
   oFormulario = oFormularios.getByIndex(0)

   'Establecemos la base de datos fuente
   oFormulario.DataSourceName = "Directorio"
   'Como origen una tabla
   oFormulario.CommandType = 0
   'El nombre de la tabla
   oFormulario.Command = "tblCiudades"

End Sub
```

En el siguiente ejemplo establecemos una consulta como origen de datos.

```
Sub Formularios6()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()
   oFormulario = oFormularios.getByIndex(0)

   'Establecemos la base de datos fuente
   oFormulario.DataSourceName = "Directorio"
   'Como origen una consulta
   oFormulario.CommandType = 1
   'El nombre de la consulta
   oFormulario.Command = "qryCiudades"

End Sub
```

Por ultimo, podemos establecer un comando SQL como origen de datos.

```
Sub Formularios7()
Dim oDoc As Object
Dim oHojaActiva As Object
```

```
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()
   oFormulario = oFormularios.getByIndex(0)

   'Establecemos la base de datos fuente
   oFormulario.DataSourceName = "Directorio"
   'Como origen una instrucción SQL
   oFormulario.CommandType = 2
   'La instrucción SQL
   oFormulario.Command = "SELECT * FROM tblContactos"

End Sub
```

Todas las precauciones y particularidades vistas en las bases de datos, son aplicables a los formularios, por lo que te recomiendo las tengas siempre presentes.

Puedes agregar nuevos formularios a la colección.

```
Sub Formularios8()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oNuevoFormulario As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()

   'Creamos un nuevo formulario
   oNuevoFormulario = oDoc.createInstance( "com.sun.star.form.component.Form" )
   'Lo agregamos a los formularios
   oFormularios.insertByName("MiDirectorio", oNuevoFormulario)

End Sub
```

Puedes agregar formularios con el mismo nombre y no te dará ningún error, pero no es lo común, es mejor que verifiques que el nombre no exista como ya se vio y agregues nombres únicos a la colección de formularios para su fácil identificación.

7.2 Etiquetas (Label)

Las etiquetas son los controles más simples, pues su función es solo mostrar un texto al usuario sin que este pueda modificarlo, generalmente un titulo, un mensaje o el encabezado de un campo. Se les puede asociar un evento, por ejemplo, el clic del ratón, pero no es usual que se haga, generalmente son controles estáticos. El siguiente ejemplo, agrega una etiqueta al formulario llamado “MiDirectorio”.

```
Sub Etiquetas1()
Dim oDoc As Object
```

```
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim oNuevaEtiqueta As Object
Dim oNuevaEtiquetaModelo As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()
   oFormulario = oFormularios.getByName( "MiDirectorio" )

   'Creamos una nueva forma de control
   oNuevaEtiqueta = oDoc.createInstance("com.sun.star.drawing.ControlShape")

   'Cambiamos su tamaño y posición
   Call CambiaTam( oNuevaEtiqueta, 5000, 1500 )
   Call CambiaPos( oNuevaEtiqueta, 1000, 1000 )

   'Creamos una nueva etiqueta
   oNuevaEtiquetaModelo = oDoc.createInstance("com.sun.star.form.component.FixedText")
   'Establecemos su nombres
   oNuevaEtiquetaModelo.Name = "lblId"
   'El texto que verá el usuario
   oNuevaEtiquetaModelo.Label = "Clave"

   'Conectamos la forma con el nuevo control
   oNuevaEtiqueta.Control = oNuevaEtiquetaModelo

   'Lo agregamos al formulario, el primer argumento, en este caso 0, es el índice que tendrá el nuevo
control agregado
   oFormulario.insertByIndex(0, oNuevaEtiquetaModelo)

   'Agregamos la forma a la página de dibujo para que sea visible
   oPaginaDibujo.add( oNuevaEtiqueta )

End Sub

'Macro para cambiar la posición de un objeto
Sub CambiaPos( Obj As Object, X As Long, Y As Long )
Dim oPos As New com.sun.star.awt.Point

   oPos.X = X
   oPos.Y = Y
   Obj.setPosition( oPos )

End Sub

'Macro para cambiar el tamaño de un objeto
Sub CambiaTam( Obj As Object, Ancho As Long, Alto As Long )
Dim oTam As New com.sun.star.awt.Size

   oTam.Width = Ancho
   oTam.Height = Alto
   Obj.setSize( oTam )

End Sub
```

Las subrutinas para cambiar de posición y tamaño, son las mismas usadas en capítulos anteriores. Al ser una forma (shape), los controles soportan la mayoría de las propiedades vistas anteriormente (color, línea, relleno, fuente, etc), por lo que no las repetiremos aquí y nos centraremos en la forma de manipular las propiedades particulares de cada control desde su “modelo”, por ejemplo, para acceder a la etiqueta creada en el ejemplo anterior, usamos el siguiente código.

```
Sub Etiquetas2()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim olblClave As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()
   oFormulario = oFormularios.getByName( "MiDirectorio" )

   'Referencia al control llamado lblId
   olblClave = oFormulario.getByName("lblId")
   'Cambiamos el texto que ve el usuario
   olblClave.Label = "Nuevo texto"

End Sub
```

Otras propiedades que puedes manipular en las etiquetas son.

```
Sub Etiquetas3()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim olblClave As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()
   oFormulario = oFormularios.getByName( "MiDirectorio" )

   'Referencia al control llamado lblId
   olblClave = oFormulario.getByName("lblId")

   With olblClave
         .FontName = "Linux Biolinum"            'La fuente
         .FontHeight = 18                        'El tamaño de fuente
         .BackgroundColor = RGB(200,200,200)     'El color de fondo
         .Border = 2                             'El tipo de borde (2 = plano )
         .BorderColor = RGB(255,0,0)             'El color del borde (solo efectivo si Border = 2)
         .TextColor = RGB(0,0,200)               'El color del texto
         .Align = 1                              'Alineación horizontal
         .VerticalAlign = 1                      'Alineación vertical
   End With

End Sub
```

El borde (Border), también puede ser; sin borde(0) o 3D (1), la alienación horizontal puede ser a la izquierda (0), al centro (1) o a la derecha (2), la vertical, puede tomar los valores, arriba (0), en medio (1) o abajo (2). La mayoría de los controles, cuentan con una propiedad para activarlos o desactivarlos (Enabled), es decir, para permitir su interacción con el usuario o no, en el siguiente ejemplo, se invierta el valor de esta propiedad de la misma etiqueta.

```
Sub Etiquetas4()
Dim oDoc As Object
```

```
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim olblClave As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()
   oFormulario = oFormularios.getByName( "MiDirectorio" )

   'Referencia al control llamado lblId
   olblClave = oFormulario.getByName("lblId")
   'Invertimos el valor de la propiedad
   olblClave.Enabled = Not olblClave.Enabled

End Sub
```

Esta propiedad es más útil en otro tipo de controles, que en las etiquetas.

7.3 Cuadros de texto (TextBox)

Los cuadro de texto (TextBox) son controles que nos permiten interactuar con el usuario, tanto para mostrar información, como para permitirle modificarla, en el siguiente ejemplo, agregamos un nuevo cuadro de texto a nuestro formulario.

```
Sub CuadroTexto1()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim oCuadroTexto As Object
Dim oCuadroTextoModelo As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()
   oFormulario = oFormularios.getByName( "MiDirectorio" )

   'Creamos una nueva forma de control
   oCuadroTexto = oDoc.createInstance("com.sun.star.drawing.ControlShape")

   'Cambiamos su tamaño y posicion
   Call CambiaTam( oCuadroTexto, 10000, 800 )
   Call CambiaPos( oCuadroTexto, 2000, 2000 )

   'Creamos un nuevo cuadro de texto
   oCuadroTextoModelo = oDoc.createInstance("com.sun.star.form.component.TextField")
   'Establecemos su nombre
   oCuadroTextoModelo.Name = "txtNombre"

   'Conectamos la forma con el nuevo control
   oCuadroTexto.Control = oCuadroTextoModelo

   'Lo agregamos al formulario
```

```
    oFormulario.insertByIndex(0, oCuadroTextoModelo)

    'Agregamos la forma a la página de dibujo para que sea visible
    oPaginaDibujo.add( oCuadroTexto )

End Sub
```

Los cuadros de texto, comparten algunas propiedades con las etiquetas.

```
Sub CuadroTexto2()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim otxtNombre As Object

    oDoc = ThisComponent
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()
    oPaginaDibujo = oHojaActiva.getDrawPage()
    oFormularios = oPaginaDibujo.getForms()
    oFormulario = oFormularios.getByName( "MiDirectorio" )

    'Referencia al control llamado txtNombre
    otxtNombre = oFormulario.getByName("txtNombre")

    With otxtNombre
          .FontName = "Linux Biolinum"            'La fuente
          .FontHeight = 14                        'El tamaño de fuente
          .BackgroundColor = RGB(230,230,230)     'El color de fondo
          .Border = 1                             'El tipo de borde (1 = 3D )
          .TextColor = RGB(0,100,200)             'El color del texto
          .Align = 0                              'Alineación horizontal (izquierda)
    End With

End Sub
```

El principal uso de estos controles es recuperar su contenido o establecerlo.

```
Sub CuadroTexto3()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim otxtNombre As Object
Dim sInfo As String

    oDoc = ThisComponent
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()
    oPaginaDibujo = oHojaActiva.getDrawPage()
    oFormularios = oPaginaDibujo.getForms()
    oFormulario = oFormularios.getByName( "MiDirectorio" )

    'Referencia al control llamado txtNombre
    otxtNombre = oFormulario.getByName("txtNombre")
    'Recuperamos el texto del control
    sInfo = otxtNombre.Text
    'Lo mostramos
    MsgBox sInfo
    'Solicitamos un nuevo contenido
    sInfo = InputBox("Escribe un nuevo contenido para el control")
```

```
    'Y lo establecemos
    otxtNombre.Text = sInfo

End Sub
```

También puedes usar la propiedad String para recuperar o establecer el texto del control. Los cuadros de texto, cuentan con muchas más propiedades.

```
Sub CuadroTexto4()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim otxtNombre As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()
   oFormulario = oFormularios.getByName( "MiDirectorio" )

   'Referencia al control llamado txtNombre
   otxtNombre = oFormulario.getByName("txtNombre")

   With otxtNombre
         .EchoChar = 42         'El carácter de contraseña 42 = *
         .MaxTextLen = 20       'Longitud máxima aceptada
         .MultiLine = True           'Si es o no multilínea
         .HScroll = False       'Si se muestra o no la barra de desplazamiento horizontal
         .VScroll = True        'Si se muestra o no la barra de desplazamiento vertical
         .Printable = True           'Si es imprime o no el control
         .ReadOnly = False           'Si es o no de solo lectura
         .DataField = "Nombre"  'Establece el campo de la base de datos mostrada
   End With

End Sub
```

Algunas de estas propiedades se contraponen, por ejemplo, si estableces el control en multilínea (MultiLine), el carácter de contraseña (EchoChar) no tiene efecto como tampoco las barras de desplazamiento (Hscroll y Vscroll), por supuesto, si estableces el control en solo lectura (ReadOnly), el usuario no podrá escribir en él. El campo vinculado (DataField), solo será efectivo si previamente el formulario se relacionó con una base de datos y con una fuente de datos (tabla, consulta o instrucción SQL), además, el nombre del campo **"debe existir"** en el origen de datos. Si quieres quitar la relación, establece esta propiedad en vacía.

Puedes vincular el control con una celda de cualquier hoja, esto te permite reflejar cualquier cambio que se haga en uno en el otro, solo puedes establecer el campo vinculado (DataField) o la celda vinculada (ValueBinding), tendrá efecto el ultimo que establezcas como en nuestro ejemplo.

```
Sub CuadroTexto5()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim otxtNombre As Object
Dim oDirCeldaVinculada As Object
Dim mOpc(0) As New "com.sun.star.beans.NamedValue"
Dim oVinculo As Object
```

```
    oDoc = ThisComponent
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()
    oPaginaDibujo = oHojaActiva.getDrawPage()
    oFormularios = oPaginaDibujo.getForms()
    oFormulario = oFormularios.getByName( "MiDirectorio" )

    'Referencia al control llamado txtNombre
    otxtNombre = oFormulario.getByName("txtNombre")
    'Obtenemos la dirección de la celda B5
    oDirCeldaVinculada = oHojaActiva.getCellByPosition(1,4).getCellAddress
    'Creamos la propiedad para vincular
    mOpc(0).Name = "BoundCell"
    mOpc(0).Value = oDirCeldaVinculada
    'Creamos la instancia de la celda a vincular
    oVinculo = oDoc.createInstanceWithArguments("com.sun.star.table.CellValueBinding", mOpc())
    'Y la vinculamos al cuadro de texto
    otxtNombre.setValueBinding( oVinculo )

End Sub
```

Puedes activar y desactivar (Enabled) el control, toma en cuenta que activarlo es diferente de establecerlo en solo lectura (ReadOnly).

```
Sub CuadroTexto6()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim otxtNombre As Object

    oDoc = ThisComponent
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()
    oPaginaDibujo = oHojaActiva.getDrawPage()
    oFormularios = oPaginaDibujo.getForms()
    oFormulario = oFormularios.getByName( "MiDirectorio" )

    'Referencia al control llamado txtNombre
    otxtNombre = oFormulario.getByName("txtNombre")
    'Activamos y desactivamos el control
    otxtNombre.Enabled = Not otxtNombre.Enabled

End Sub
```

Este control, es posible convertirlo en cuadro de texto enriquecido, capaz de mostrar texto con diferentes formatos.

```
Sub CuadroTexto7()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim otxtNombre As Object
Dim oCursor As Object

    oDoc = ThisComponent
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()
    oPaginaDibujo = oHojaActiva.getDrawPage()
    oFormularios = oPaginaDibujo.getForms()
    oFormulario = oFormularios.getByName( "MiDirectorio" )
```

```
    'Referencia al control llamado txtNombre
    otxtNombre = oFormulario.getByName("txtNombre")
    'Establecemos que sea multilínea y que soporte texto enriquecido
    otxtNombre.Multiline = True
    otxtNombre.RichText = True
    'Creamos un cursor
    oCursor = otxtNombre.createTextCursor()
    'Insertamos una línea de texto
    otxtNombre.insertString( oCursor, "Primera línea", False)
    'Insertamos un salto de parrafo
    otxtNombre.insertControlCharacter( oCursor, 0, False)
    'Insertamos otra línea de texto
    otxtNombre.insertString( oCursor, "Segunda línea", False)

End Sub
```

Una tarea muy común al trabajar con controles, es enviarles el foco, es decir, mover el cursor a este, para ello, tenemos que aprender un nuevo concepto, hasta ahora para acceder a los controles de un formulario, nos ha bastado como medio el mismo formulario, a este modo de acceso se le llama “modelo”, pero para usar otras propiedades y métodos, solo es posible hacer a través de la “vista” del control, para entrar al modo “vista, usamos el siguiente código.

```
Sub CuadroTexto8()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim otxtNombre As Object
Dim otxtNombreVista As Object

   oDoc = ThisComponent
   oHojaActiva = oDoc.getCurrentController.getActiveSheet()
   oPaginaDibujo = oHojaActiva.getDrawPage()
   oFormularios = oPaginaDibujo.getForms()
   oFormulario = oFormularios.getByName( "MiDirectorio" )

   otxtNombre = oFormulario.getByName("txtNombre")
   'Accedemos a la vista del control
   otxtNombreVista = oDoc.getCurrentController.getControl( otxtNombre )
   'Le enviamos el foco
   otxtNombreVista.setFocus()

End Sub
```

Observa como accedemos al modo “vista” del control, a través del controlador (*getCurrentController*) del documento. Desde el modo “vista”, puedes acceder también al contenido del control con su propiedad *Text*, pero algunas otras propiedades solo están disponibles en este modo, como posibilidad de ocultar o mostrar el control.

```
Sub CuadroTexto9()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim otxtNombre As Object
Dim otxtNombreVista As Object

   oDoc = ThisComponent
```

```
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()
    oPaginaDibujo = oHojaActiva.getDrawPage()
    oFormularios = oPaginaDibujo.getForms()
    oFormulario = oFormularios.getByName( "MiDirectorio" )

    otxtNombre = oFormulario.getByName("txtNombre")
    'Accedemos a la vista del control
    otxtNombreVista = oDoc.getCurrentController.getControl( otxtNombre )
    'Alternamos entre mostrar y ocultar el control
    otxtNombreVista.setVisible( Not otxtNombreVista.isVisible )

End Sub
```

Toma en cuenta que si el formulario esta en modo “diseño”, el control de todos modos se mostrará, este método, solo tiene efecto en tiempo de “ejecución”. Casi todos los controles que veremos, tienen su modo “vista” y la posibilidad de enviar el foco y ocultarlos.

7.4 Casilla de verificación (CheckBox)

Las casillas de verificación (CheckBox), son controles de verdadero o falso, es decir, generalmente se usan para indicar si se hace o no una actividad u opción. El siguiente ejemplo agrega una casilla de verificación a nuestro formulario.

```
Sub CasillaVerificacion1()
Dim oDoc As Object
Dim oHojaActiva As Object
Dim oPaginaDibujo As Object
Dim oFormularios As Object
Dim oFormulario As Object
Dim oCasilla As Object
Dim oCasillaModelo As Object

    oDoc = ThisComponent
    oHojaActiva = oDoc.getCurrentController.getActiveSheet()
    oPaginaDibujo = oHojaActiva.getDrawPage()
    oFormularios = oPaginaDibujo.getForms()
    oFormulario = oFormularios.getByName( "MiDirectorio" )

    'Creamos una nueva forma de control
    oCasilla = oDoc.createInstance("com.sun.star.drawing.ControlShape")

    'Cambiamos su tamaño y posicion
    Call CambiaTam( oCasilla, 8000, 800 )
    Call CambiaPos( oCasilla, 3000, 5000 )

    'Creamos una nueva casilla de verificación
    oCasillaModelo = oDoc.createInstance("com.sun.star.form.component.CheckBox")
    'Establecemos su nombre
    oCasillaModelo.Name = "chkMayus"
    'Y el texto que verá el usuario
    oCasillaModelo.Label = "Guardar todo en MAYUSCULAS"

    'Conectamos la forma con el nuevo control
    oCasilla.Control = oCasillaModelo

    'Lo agregamos al formulario
    oFormulario.insertByIndex(0, oCasillaModelo)

    'Agregamos la forma a la página de dibujo para que sea visible
```

```
   oPaginaDibujo.add( oCasilla )

End Sub
```

Al contar con texto para mostrar al usuario, este se puede formatear completamente como en las etiquetas (label) y los cuadros de texto (textbox).

```
Sub CasillaVerificacion2()
Dim oFormulario As Object
Dim ochkMayus As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   ochkMayus = oFormulario.getByName( "chkMayus" )

   With ochkMayus
         .Align = 0                              'Alineación horizontal
         .BackgroundColor = RGB(200,220,240)     'Color de fondo
         .TextColor = RGB(50,70,90)              'Color de la fuente
         .FontHeight = 16                        'Tamaño de la fuente
         .Printable = True                             'Si se imprime el control
         .VerticalAlign = 1                      'Alineación vertical
         .TriState = False                             'Establece el estado triple
         .VisualEffect = 2                             'Formato plano (1 = 3D)
   End With

End Sub
```

Puedes activarlo y desactivarlo.

```
Sub CasillaVerificacion3()
Dim oFormulario As Object
Dim ochkMayus As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   ochkMayus = oFormulario.getByName( "chkMayus" )
   'Alternamos la activación del control
   ochkMayus.Enabled = Not ochkMayus.Enabled

End Sub
```

Para obtener o establecer el estado del control, se usa la propiedad State, esta puede tomar los valores 0, 1 o 2, dependiendo si; esta activa o no, y de la propiedad de triple estado (TriState), que permite tener este control en, activo, desactivado o sin selección.

```
Sub CasillaVerificacion4()
Dim oFormulario As Object
Dim ochkMayus As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   ochkMayus = oFormulario.getByName( "chkMayus" )

   'Verificamos el estado de la casilla
   Select Case ochkMayus.State
         Case 0
               MsgBox "La casilla no esta seleccionada"
         Case 1
               MsgBox "La casilla esta seleccionada"
```

```
        Case 2
            MsgBox "La casilla no tiene selección"
    End Select
    'La activamos
    ochkMayus.State = 1

End Sub
```

Las casillas de verificación, también las puedes vincular con un campo de datos (DataField) o con una celda (ValueBinding), con el mismo código de los cuadros de texto.

7.5 Campo formateado (FormattedField)

Los campos formateados (FormattedField), son controles casi idénticos a los cuadros de texto (TextBox), con la diferencia de que estos aceptan todos los formatos que tenemos disponibles en el menú *Formato* | *Celdas...,* en la ficha Números, y se les puede establecer un valor máximo y un mínimo.

```
Sub CampoFormateado1()
Dim oFormulario As Object
Dim oCuadroConFormato As Object
Dim oCuadroConFormatoModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   'Creamos una nueva forma de control
   oCuadroConFormato = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   'Cambiamos su tamaño y posicion
   Call CambiaTam( oCuadroConFormato, 8000, 800 )
   Call CambiaPos( oCuadroConFormato, 3000, 5000 )

   'Creamos un nuevo campo formateado
   oCuadroConFormatoModelo = ThisComponent.createInstance(
"com.sun.star.form.component.FormattedField" )
   'Establecemos su nombre
   oCuadroConFormatoModelo.Name = "txtFVentas"

   'Conectamos la forma con el nuevo control
   oCuadroConFormato.Control = oCuadroConFormatoModelo

   'Lo agregamos al formulario
   oFormulario.insertByIndex(0, oCuadroConFormatoModelo)

   'Agregamos la forma a la página de dibujo para que sea visible
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oCuadroConFormato )

End Sub
```

Puede usar las mismas propiedades que para los cuadros de texto (TextBox), aquí vemos las propiedades particulares para este control.

```
Sub CampoFormateado2()
Dim oFormulario As Object
Dim otxtfVentas As Object
```

```
   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   otxtfVentas = oFormulario.getByName ( "txtFVentas" )

   With otxtfVentas
         .EffectiveMin = 1           'El valor mínimo
         .EffectiveMax = 100    'El valor máximo
         .Spin = True           'Si se muestra o no el campo giratorio
         .FormatKey = 4         'El formato para el contenido ( 4 = "#,##0.00" )
   End With

End Sub
```

Si estableces que se muestre el campo giratorio (Spin), la alienación del contenido siempre será a la izquierda. El valor para el formato (FormatKey), es el mismo para las celdas, puedes obtener estos valores con la macro que te muestro en; Listar formatos en un archivo de Calc. Este control también lo puedes activar y desactivar (Enabled) y establecerle un campo de datos origen (DataField), así como vincularlo con una celda (ValueBinding).

7.6 Botón de comando (CommandButton)

Los botones de comando (CommandButton) son controles generalmente usados para iniciar una acción por que usualmente tienen asociada una macro al evento “clic del ratón”, aunque soportan varios más. Veamos como agregar uno a nuestro formulario.

```
Sub BotonComando1()
Dim oFormulario As Object
Dim oBotonComando As Object
Dim oBotonComandoModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   'Creamos una nueva forma de control
   oBotonComando = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   'Cambiamos su tamaño y posicion
   Call CambiaTam( oBotonComando, 5000, 800 )
   Call CambiaPos( oBotonComando, 3000, 10000 )

   'Creamos un nuevo botón de comando
   oBotonComandoModelo = ThisComponent.createInstance("com.sun.star.form.component.CommandButton")
   'Establecemos su nombre
   oBotonComandoModelo.Name = "cmdGuardar"
   oBotonComandoModelo.Label = "~Guardar"

   'Conectamos la forma con el nuevo control
   oBotonComando.Control = oBotonComandoModelo

   'Lo agregamos al formulario
   oFormulario.insertByIndex(0, oBotonComandoModelo)

   'Agregamos la forma a la página de dibujo para que sea visible
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oBotonComando )

End Sub
```

Observa el símbolo “~” antes del título del botón de comando, este, nos permite subrayar la letra inmediata siguiente para usarse como método abreviado de teclado, ten cuidado,

si estableces una combinación ya usada en la interfaz del usuario, esta tendrá preferencia, en los formularios, este método abreviado tiene una función ligeramente diferente, cuando el “foco”, es decir el cursor, esta dentro de algún control del formulario, la combinación de teclas llevará el cursor a dicho control, pero solo eso, solo le lleva el foco. Veamos las principales propiedades particulares de este control.

```
Sub BotonComando2()
Dim oFormulario As Object
Dim ocmdGuardar As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   'Creamos una nueva forma de control
   ocmdGuardar = oFormulario.getByName ("cmdGuardar")

   With ocmdGuardar
         .Align = 1                              'Alineación horizontal ( 1 = centro )
         .VerticalAlign = 1                      'Alineación vertical ( 1 = medio )
         .BackgroundColor = RGB(220,230,240)     'Color de fondo
         .DefaultButton = True                   'Si es el botón predeterminado
         .FontName = "Linux Biolinum"            'La fuente
         .FontHeight = 18                        'El tamaño de fuente
         .TextColor = RGB(0,20,250)              'El color del texto
         .Printable = True                             'Si se imprime el control
   End With

End Sub
```

Este control también cuenta con la propiedad para activar o desactivar (Enabled), muy útil para deshabilitar momentáneamente su uso, cuando tiene una macro asociada.

7.7 Botón de opción (OptionButton)

Estos controles generalmente trabajan juntos dos o más, pues permiten seleccionar solo una opción de entre los que haya en el formulario, por ello, generalmente se agregan dos o más como en el siguiente ejemplo.

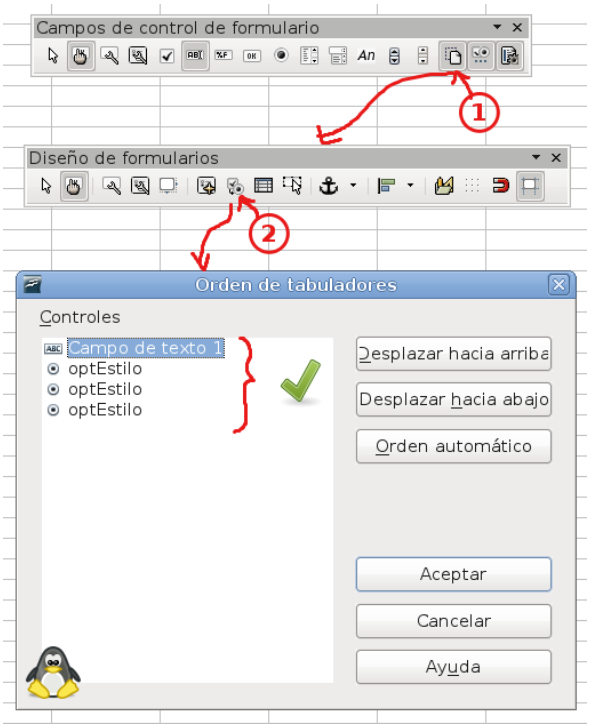
```
Sub BotonOpcion1()
Dim oFormulario As Object
Dim oBotonOpcion As Object
Dim oBotonOpcionModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   'Creamos una nueva forma de control
   oBotonOpcion = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   'Cambiamos su tamaño y posicion
   Call CambiaTam( oBotonOpcion, 5000, 800 )
   Call CambiaPos( oBotonOpcion, 3000, 11000 )

   'Creamos un nuevo botón de opción
   oBotonOpcionModelo = ThisComponent.createInstance("com.sun.star.form.component.RadioButton")
   'Establecemos su nombre
   oBotonOpcionModelo.Name = "optEstilo"
   oBotonOpcionModelo.Label = "Color"

   'Conectamos la forma con el nuevo control
```

```
    oBotonOpcion.Control = oBotonOpcionModelo

    'Lo agregamos al formulario
    oFormulario.insertByIndex(0, oBotonOpcionModelo)

    'Agregamos la forma a la página de dibujo para que sea visible
    ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oBotonOpcion )

    'Agregamos un segundo control
    oBotonOpcion = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")
    Call CambiaTam( oBotonOpcion, 5000, 800 )
    Call CambiaPos( oBotonOpcion, 3000, 12000 )
    oBotonOpcionModelo = ThisComponent.createInstance("com.sun.star.form.component.RadioButton")
    oBotonOpcionModelo.Name = "optEstilo"
    oBotonOpcionModelo.Label = "Escala de grises"
    oBotonOpcion.Control = oBotonOpcionModelo
    oFormulario.insertByIndex(0, oBotonOpcionModelo)
    ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oBotonOpcion )

    'Y un tercero
    oBotonOpcion = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")
    Call CambiaTam( oBotonOpcion, 5000, 800 )
    Call CambiaPos( oBotonOpcion, 3000, 13000 )
    oBotonOpcionModelo = ThisComponent.createInstance("com.sun.star.form.component.RadioButton")
    oBotonOpcionModelo.Name = "optEstilo"
    oBotonOpcionModelo.Label = "Blanco y negro"
    oBotonOpcion.Control = oBotonOpcionModelo
    oFormulario.insertByIndex(0, oBotonOpcionModelo)
    ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oBotonOpcion )

End Sub
```

Observa que a los tres les hemos establecido el mismo nombre, esto es importante, si quieres que se comporten como un grupo, de este modo, puedes crear tantos grupos de controles de botón de opción (OptionButton) como quieras, simplemente estableciendo el mismo nombre para cada grupo agregado. Al tener texto visible por el usuario, comparten la mayoría de las propiedades vistas hasta ahora (color de fuente y fondo, tamaño de fuente, estilo de fuente, etc), pero observa que hasta ahora, hemos usado el nombre del control para hacer referencia a el, pero a estos controles les hemos dado el mismo nombre, para distinguirlos, ahora accederemos a ellos con su índice, en vez de con su nombre, como en el siguiente ejemplo.

```
Sub BotonOpcion2()
Dim oFormulario As Object
Dim ooptEstilo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )

   ooptEstilo = oFormulario.getByIndex(0)
   ooptEstilo.BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)     'Color de fondo
   ooptEstilo.TextColor = RGB(Rnd*255,Rnd*255,Rnd*255)           'Color de la fuente

   ooptEstilo = oFormulario.getByIndex(1)
   ooptEstilo.BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)     'Color de fondo
   ooptEstilo.TextColor = RGB(Rnd*255,Rnd*255,Rnd*255)           'Color de la fuente

   ooptEstilo = oFormulario.getByIndex(2)
   ooptEstilo.BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)     'Color de fondo
   ooptEstilo.TextColor = RGB(Rnd*255,Rnd*255,Rnd*255)           'Color de la fuente

End Sub
```

Y con justa razón te preguntarás, ¿como se que índice tiene cada control?, puedes hacerlo de varias maneras, visualmente en la interfaz de Calc, en el navegador de formularios, para ver esta ventana, tienes que tener visible la barra de herramientas “Campos de control de formulario”, visible desde el menú *Ver | Barra de herramientas ->*, después solo sigue la secuencia de la imagen siguiente.



Con código, puedes acceder a los controles por el nombre del grupo, como en el ejemplo siguiente.

```
Sub BotonOpcion3()
Dim oFormulario As Object
Dim oGrupoEstilo() As Object
Dim oBoton As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )

   'Accedemos al grupo completo
   oFormulario.getGroupByName("optEstilo", oGrupoEstilo)
   'Iteramos entre los elementos del grupo
   For Each oBoton In oGrupoEstilo()
         'Cambiamos el color de fondo aleatoriamente
         oBoton.BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)
   Next

End Sub
```

Lo importante de los botones de opción (Optionbutton) es saber cual de ellos, dentro del grupo, esta seleccionado, para ello se consulta su propiedad State, que solo uno de ellos puede tener en verdadero (True).

```
Sub BotonOpcion4()
Dim oFormulario As Object
Dim oGrupoEstilo() As Object
Dim oBoton As Object
```

```
   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )

   oFormulario.getGroupByName("optEstilo", oGrupoEstilo)
   For Each oBoton In oGrupoEstilo()
         'Consultamos el estado del botón
         If oBoton.State Then
               'Mostramos el que este seleccionado
               MsgBox "Opción seleccionada: " & oBoton.Label
               Exit For
         End If
   Next

End Sub
```

Como veremos más adelante, cuando se asocia una macro al evento clic de los botones de opción, es muy sencillo saber cual es el que esta seleccionado, sin recorrer uno a uno. Estos controles también cuenta con la propiedad de activarse o desactivarse (Enabled) y la capacidad para establecerle un campo de datos origen (DataField), así como vincularlo con una celda (ValueBinding).

7.8 Cuadro de lista (ListBox)

Los cuadros de lista (ListBox), son controles usados generalmente para mostrar un listado y poder seleccionar una o varias opciones dependiendo de sus propiedades, en el siguiente ejemplo, agregamos un nuevo control de lista a nuestro formulario.

```
Sub CuadroLista1()
Dim oFormulario As Object
Dim oCuadroLista As Object
Dim oCuadroListaModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   'Creamos una nueva forma de control
   oCuadroLista = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   'Cambiamos su tamaño y posicion
   Call CambiaTam( oCuadroLista, 5000, 5000 )
   Call CambiaPos( oCuadroLista, 3000, 10000 )

   'Creamos un nuevo cuadro de lista
   oCuadroListaModelo = ThisComponent.createInstance("com.sun.star.form.component.ListBox")
   'Establecemos su nombre
   oCuadroListaModelo.Name = "lstCiudades"

   'Conectamos la forma con el nuevo control
   oCuadroLista.Control = oCuadroListaModelo

   'Lo agregamos al formulario
   oFormulario.insertByIndex(0, oCuadroListaModelo)

   'Agregamos la forma a la página de dibujo para que sea visible
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oCuadroLista )

End Sub
```

Las propiedades principales de este control son.

```
Sub CuadroLista2()
Dim oFormulario As Object
Dim olstCiudades As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )

   With olstCiudades
         .BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)    'Color de fondo
         .Border = 2                                        'Tipo de borde ( 1 = 3D )
         .BorderColor = RGB(Rnd*255,Rnd*255,Rnd*255)        'Solo si Border = 2
         .Dropdown = False                                       'Si es desplegable
         .MultiSelection = True                             'Si se permite la multiselección
   End With

End Sub
```

Si estableces que el control sea desplegable (Dropdown), te mostrará una flecha para desplegar el contenido, con lo que se comportará de forma similar al cuadro combinado (ComboBox), mi recomendación es establecer esta propiedad siempre en falso (False), si quieres un cuadro combinado, usa el control visto en el siguiente tema.

Haya varias formas de agregar elementos a un cuadro de lista, la primera y más sencilla es agregar una matriz de datos.

```
Sub CuadroLista3()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim mDatos()

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )

   mDatos = Array("Mexico", "Madrid","Buenos Aires","Bogota","Lima")
   olstCiudades.StringItemList = mDatos()

End Sub
```

Este control acepta también como origen de datos un campo de bases de datos (*DataField*). Otra forma es vincular el control con un origen de celdas, tanto para el valor seleccionado (*ValueBinding*), como para los valores mostrados (*ListEntrySource*).

```
Sub CuadroLista4()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim oDirCeldaVinculada As Object
Dim oRangoOrigen As Object
Dim mOpc(0) As New "com.sun.star.beans.NamedValue"
Dim oVinculo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )

   'Obtenemos la dirección de la celda B5
   oDirCeldaVinculada =
ThisComponent.getCurrentController.getActiveSheet.getCellByPosition(3,24).getCellAddress
```

```
    'Creamos la propiedad para vincular
    mOpc(0).Name = "BoundCell"
    mOpc(0).Value = oDirCeldaVinculada
    'Creamos la instancia de la celda a vincular createInstanceWithArguments
    oVinculo = ThisComponent.createInstanceWithArguments("com.sun.star.table.CellValueBinding", mOpc())
    'Y la vinculamos al cuadro de lista
    olstCiudades.setValueBinding( oVinculo )

    'Establecemos el rango de celdas origen
    oRangoOrigen =
ThisComponent.getCurrentController.getActiveSheet.getCellRangeByName("D27:D40").getRangeAddress
    mOpc(0).Name = "CellRange"
    mOpc(0).Value = oRangoOrigen
    'Creamos el vinculo
    oVinculo = ThisComponent.createInstanceWithArguments( "com.sun.star.table.CellRangeListSource",
mOpc())
    'Lo vinculamos al cuadro de lista
    olstCiudades.setListEntrySource( oVinculo )

End Sub
```

Otra forma es agregar (y quitar) los elementos de forma dinámica, para lograr esto, repasemos un concepto para el acceso a propiedades y métodos de los controles, hasta ahora, con hacer referencia al control a través del formulario nos era suficiente, a esta forma de referencia se le llama de acceso al “modelo”, pero algunas propiedades y métodos solo están disponibles en el modo de acceso “vista”, para acceder a este modo, usamos el siguiente código.

```
Sub CuadroLista5()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object

    oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
    olstCiudades = oFormulario.getByName( "lstCiudades" )
    'Accedemos a la vista del control a través del controlador del documento
    olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )
    'Y podemos enviar el foco, es decir, el cursor a el
    olstCiudadesVista.setFocus()

End Sub
```

En el acceso “vista”, tenemos una serie de métodos y propiedades para manipular completamente los elementos de un cuadro de lista (ListBox), veamos los más importantes. Para agregar elementos usamos el método *addItem* de la siguiente manera.

```
Sub CuadroLista6()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object
Dim sInfo As String

    oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
    olstCiudades = oFormulario.getByName( "lstCiudades" )
    olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )

    Do
          'Solicitamos un elemento
          sInfo = Trim( InputBox("Escribe la ciudad a agregar"))
          'Si esta vacio salimos
          If sInfo = "" Then Exit Do
```

```
        'Lo agregamos como primer elemento
        olstCiudadesVista.addItem( sInfo, 0)
    Loop

End Sub
```

El código anterior siempre agregará los nuevos elementos al inicio de la lista, si quieres agregarlos al final, tenemos que saber cuantos elementos ya existen en la lista, esto lo logramos con la propiedad *ItemCount*, que te devuelve el número total de elementos actuales en el cuadro de lista.

```
Sub CuadroLista7()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object
Dim sInfo As String

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )
   olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )

   Do
        sInfo = Trim( InputBox("Escribe la ciudad a agregar"))
        If sInfo = "" Then Exit Do
        'Lo agregamos como ultimo elemento
        olstCiudadesVista.addItem( sInfo, olstCiudadesVista.ItemCount)
   Loop

End Sub
```

También podemos agregar una matriz completa.

```
Sub CuadroLista8()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object
Dim sInfo()

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )
   olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )

   sInfo = Array("Barcelona","Montevideo","Arequipa")
   'Agregamos una matriz completa
   olstCiudadesVista.addItems( sInfo, 0)

End Sub
```

Observa que el método es *addItems*, en plural. Solo puedes agregar matrices de una sola dimensión. Como recomendación, cuando los datos a agregar sean muchos, lo notaras si tu código se ejecuta lento, primero llena la matriz y después usa *addItems*, veras la diferencia en comparación a agregar cada elemento individual con *addItem*.

Ahora, vamos a eliminar elementos de la lista, para ellos existe un solo método, pero es más que suficiente.

```
Sub CuadroLista9()
```

```
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object
Dim iRes As Integer

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )
   olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )

   iRes = MsgBox( "¿Estas seguro de borrar el siguiente elemento? = " & olstCiudadesVista.getItem(
0 ), 4)
   If iRes = 6 Then
         'Quitamos el primer elemento de la lista
         olstCiudadesVista.removeItems( 0,1 )
   End If

   iRes = MsgBox( "¿Estas seguro de borrar el siguiente elemento? = " & olstCiudadesVista.getItem(
olstCiudadesVista.getItemCount-1 ), 4)
   If iRes = 6 Then
         'Quitamos el ultimo elemento de la lista
         olstCiudadesVista.removeItems( olstCiudadesVista.getItemCount-1,1 )
   End If

End Sub
```

El método para eliminar elementos (removeItems), requiere de dos argumentos, el primero es la posición del primer elemento a borrar, las posiciones del cuadro de lista, como en las matrices, siempre empieza en cero, el segundo argumento es el número total de elementos a borrar. En los dos ejemplos anteriores, solo eliminamos un elemento, el primero (0) y el ultimo (getItemCount-1), es decir, contamos el total de elemento y le restamos uno para obtener el índice del ultimo elemento. Observa como le mostramos al usuario el elemento a borrar con el método *getItem*, simplemente pasandole el índice del elemento a mostrar, este método siempre te devuelve un texto (string) con el contenido del elemento indicado. Con estos argumentos en posible limpiar la lista completa, es decir, dejarla en blanco para empezar de nuevo.

```
Sub CuadroLista10()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object
Dim iRes As Integer

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )
   olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )

   iRes = MsgBox( "¿Estas seguro de borrar TODOS los elementos de la lista?", 4 )
   If iRes = 6 Then
         'Quitamos todos los elementos de la lista
         olstCiudadesVista.removeItems( 0, olstCiudadesVista.getItemCount )
   End If

End Sub
```

Recuerda que es una buena practica de programación, cuando se realizan operaciones que no se puedan deshacer, sobre todo si son de borrado, preguntar al usuario si esta seguro de realizarla. Ya vimos como devolver un elemento de la lista (*getItem*), ahora, veremos como devolver todos los elementos de la lista (*getItems*).

```
Sub CuadroLista11()
```

```
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object
Dim mDatos()
Dim co1 As Integer

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )
   olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )

   'Devolvemos todos los elementos a una matriz
   mDatos() = olstCiudadesVista.getItems()
   'Mostramos los elementos en la matriz
   For co1 = LBound(mDatos) To Ubound(mDatos)
        MsgBox mDatos(co1)
   Next co1

End Sub
```

Un punto importante es saber que elemento o elementos selecciono el usuario, para ello tenemos varios métodos, veamos.

```
Sub CuadroLista12()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object
Dim sSeleccionado As String
Dim iPos As Integer

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )
   olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )

   'Obtenemos el elemento seleccionado
   sSeleccionado = olstCiudadesVista.getSelectedItem()
   'Obtenemos la posición del elemento seleccionado
   iPos = olstCiudadesVista.getSelectedItemPos()
   'Mostramos la información
   MsgBox "Esta seleccionado: " & sSeleccionado & " en la posición " & iPos

End Sub
```

Estos métodos siempre te devolverán un solo elemento, si el cuadro de lista (ListBox) no tiene “ningún” elemento seleccionado, el método getSelectedItemPos, te devolverá un valor negativo de -1, si el cuadro de lista (*ListBox*) tiene activada la propiedad para selecciones múltiples (*MultiSelection*), tienes que usar otros métodos.

```
Sub CuadroLista13()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object
Dim mSeleccionados() As String
Dim mPos() As Integer
Dim co1 As Integer

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )
   olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )
```

```
    'Obtenemos los elementos seleccionados
    mSeleccionados() = olstCiudadesVista.getSelectedItems()
    'Obtenemos la posición del los elementos seleccionados
    mPos() = olstCiudadesVista.getSelectedItemsPos()

    For co1 = LBound(mPos) To UBound(mPos)
          'Mostramos la información
          MsgBox "Esta seleccionado: " & mSeleccionados(co1) & " en la posición " & mPos(co1)
    Next co1

End Sub
```

Observa que en los dos casos estamos obteniendo matrices de datos. Para finalizar de ver las propiedades y métodos más importantes de los cuadros de lista (*ListBox*), también es posible seleccionar elementos por código, veamos como.

```
Sub CuadroLista14()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )
   olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )

   'Seleccionamos el primer elemento de la lista
   olstCiudadesVista.selectItemPos(0, True)

   'Seleccionamos el ultimo elemento de la lista
   olstCiudadesVista.selectItemPos(olstCiudadesVista.getItemCount-1, True)

End Sub
```

En el ejemplo, seleccionamos el primero y el ultimo elemento, dependiendo de la propiedad de multiselección (*MultiSelection*), el resultado será diferente, si esta en falso (*False*), la selección reemplazará a la anterior, si esta en verdadero (*True*), la selección se sumará a lo ya seleccionado. También puedes seleccionar varios elementos al mismo tiempo, claro, si la multiselección (*MultiSelection*) esta en verdadero (*True*).

```
Sub CuadroLista15()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object
Dim mSeleccionar()

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )
   olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )

   mSeleccionar() = Array(1,3,5)
   'Seleccionamos los elementos de la matriz
   olstCiudadesVista.selectItemsPos(mSeleccionar(), True)

End Sub
```

Por ultimo, puedes seleccionar un elemento por su contenido en vez de por su posición como en el siguiente ejemplo.

```
Sub CuadroLista16()
Dim oFormulario As Object
Dim olstCiudades As Object
Dim olstCiudadesVista As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   olstCiudades = oFormulario.getByName( "lstCiudades" )
   olstCiudadesVista = ThisComponent.getCurrentController.getControl( olstCiudades )

   'Seleccionamos el elemento "Lima"
   olstCiudadesVista.selectItem("Lima", True)

End Sub
```

Si el elemento no existe, no selecciona nada, por supuesto, con los métodos aprendidos en este tema, puedes asegurarte de que si exista, ¿verdad?. Este control también acepta la activación o desactivado por código (*Enabled*).

7.9 Cuadro combinado (ComboBox)

Los cuadro combinados (*ComboBox*), son una combinación de un cuadro de lista (*ListBox*) y un cuadro de texto (*TextBox*), cuentan con una flecha en el extremo derecho para desplegar el contenido de la lista, pues esta, permanece oculta y solo es visible el elemento actualmente seleccionado, para agregar este control a nuestro formulario, usamos.

```
Sub CuadroCombinado1()
Dim oFormulario As Object
Dim oCuadroCombinado As Object
Dim oCuadroCombinadoModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   'Creamos una nueva forma de control
   oCuadroCombinado = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   'Cambiamos su tamaño y posicion
   Call CambiaTam( oCuadroCombinado, 5000, 700 )
   Call CambiaPos( oCuadroCombinado, 3000, 10000 )

   'Creamos un nuevo cuadro combinado
   oCuadroCombinadoModelo = ThisComponent.createInstance("com.sun.star.form.component.ComboBox")
   'Establecemos su nombre
   oCuadroCombinadoModelo.Name = "cboPaises"
   oCuadroCombinadoModelo.DropDown = True

   'Conectamos la forma con el nuevo control
   oCuadroCombinado.Control = oCuadroCombinadoModelo

   'Lo agregamos al formulario
   oFormulario.insertByIndex(0, oCuadroCombinadoModelo)

   'Agregamos la forma a la página de dibujo para que sea visible
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oCuadroCombinado )

End Sub
```

Al ser una combinación de los controles mencionados (*ListBox y TextBox*), comparten con estos la mayoría de sus propiedades y métodos, sobre todo con el cuadro de lista (*ListBox*). Como propiedades particulares de este, tenemos el número de líneas que muestra cuando se despliega el control.

```
Sub CuadroCombinado2()
Dim oFormulario As Object
Dim ocboPaises As Object
Dim mDatos()

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   ocboPaises = oFormulario.getByName("cboPaises")

   'Agregamos datos al cuadro combinado
   mDatos() = Array("México","Belice","El Salvador","Nicaragua")
   With ocboPaises
         .StringItemList = mDatos()
         .LineCount = 5              'El número de líneas a mostrar cuando se despliegue
         .MaxTextLen = 20            'El máximo de caracteres a aceptar
   End With

End Sub
```

Para agrega y eliminar elementos de este control, se usan exactamente los mismo métodos que en el cuadro de lista (*ListBox*), sin olvidar que hay que hacerlo desde el modo “vista” del control, excepto los relacionados con la selección de elementos (*selectItem, selectItemPos, selectItemsPos*), puesto que este control trabaja de forma un poco diferente, cuando se selecciona un elemento de la lista (no es posible hacer multiselecciones), este valor se establece como valor de su propiedad *Text*, que puedes cambiar, esto es, el control puede tener un valor diferente a los de la lista de elementos.

```
Sub CuadroCombinado3()
Dim oFormulario As Object
Dim ocboPaises As Object
Dim sDato As String

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   ocboPaises = oFormulario.getByName("cboPaises")

   'Mostramos el elemento seleccionado
   MsgBox ocboPaises.Text
   'Cambiamos el texto
   ocboPaises.Text = InputBox("Escribe un nuevo valor para el control")

End Sub
```

La posibilidad de agregar elementos diferentes a los de la lista, puede tener dos vertientes, la primera (negativa) puede ser que se capture un dato no deseado, la segunda (positiva) es que el usuario puede agregar elementos nuevos directamente. Con los métodos vistos, tu determinaras si permites esto o no, dependiendo de tus necesidades, la perspectiva de una u otra, puede cambiar. Puedes usar la activación o desactivación para este control (*Enabled*)

7.10 Botón de selección (SpinButton)

Este control nos permite aumentar o disminuir el valor de otro control (o celda), por ello generalmente trabaja en conjunto con otro, para agregar uno al formulario, usamos.

```
Sub BotonSeleccion1()
Dim oFormulario As Object
Dim oBotonSeleccion As Object
Dim oBotonSeleccionModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   'Creamos una nueva forma de control
   oBotonSeleccion = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   'Cambiamos su tamaño y posicion
   Call CambiaTam( oBotonSeleccion, 4000, 700 )
   Call CambiaPos( oBotonSeleccion, 3000, 15000 )

   'Creamos un nuevo botón de selección
   oBotonSeleccionModelo = ThisComponent.createInstance("com.sun.star.form.component.SpinButton")
   'Establecemos su nombre
   oBotonSeleccionModelo.Name = "spnEdad"

   'Conectamos la forma con el nuevo control
   oBotonSeleccion.Control = oBotonSeleccionModelo

   'Lo agregamos al formulario
   oFormulario.insertByIndex(0, oBotonSeleccionModelo)

   'Agregamos la forma a la página de dibujo para que sea visible
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oBotonSeleccion )

End Sub
```

Las principales propiedades de este control son.

```
Sub BotonSeleccion2()
Dim oFormulario As Object
Dim ospnEdad As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   ospnEdad = oFormulario.getByName("spnEdad")

   With ospnEdad
         .BackGroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)    'Color de fondo
         .Border = 2                                        'Tipo de borde ( 2 = Plano, 1 = 3D )
         .BorderColor = RGB(Rnd*255,Rnd*255,Rnd*255)        'Color del borde, solo si Border = 2
         .Orientation = 1                                   'Orientación 1 = vertical, 0 = horizontal
         .Printable = True                                        'Si se imprime el control
         .SymbolColor = RGB(Rnd*255,Rnd*255,Rnd*255)        'Color del símbolo (las flechas)
         .SpinValueMin = 0                                        'Valor mínimo
         .SpinValueMax = 20                                 'Valor máximo
         .SpinIncrement = 2                                 'Incremento
   End With

End Sub
```

De forma predeterminada, el control se agrega en forma horizontal, si cambias su orientación (*Orientation*) a vertical, tienes que cambiar su tamaño, de forma que quede más alto que ancho, si no, veras un control totalmente desajustado. Para obtener el valor actual del control, usamos el siguiente código.

```
Sub BotonSeleccion3()
Dim oFormulario As Object
Dim ospnEdad As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   ospnEdad = oFormulario.getByName("spnEdad")

   MsgBox "Valor actual del control = " & ospnEdad.spinValue

End Sub
```

Este control acepta la vinculación con una celda (*ValueBinding*), así como la activación y desactivación (*Enabled*), también el envío del foco, así como ocultarlo, recordando que estas dos ultimas propiedades, están accesibles desde el modo “vista” del control.

7.11 Barra de desplazamiento (ScrollBar)

Este control es muy similar al botón de selección (*SpinButton*), pero se usa para desplazamientos más grandes, por que tiene un área para de trabajo más grande, para agregar una a nuestro formulario, usamos el siguiente código.

```
Sub BarraDesplazamiento1()
Dim oFormulario As Object
Dim oBarraDesplazamiento As Object
Dim oBarraDesplazamientoModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   'Creamos una nueva forma de control
   oBarraDesplazamiento = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   'Cambiamos su tamaño y posición
   Call CambiaTam( oBarraDesplazamiento, 10000, 700 )
   Call CambiaPos( oBarraDesplazamiento, 3000, 15000 )

   'Creamos una nueva barra de desplazamiento
   oBarraDesplazamientoModelo = ThisComponent.createInstance("com.sun.star.form.component.ScrollBar")
   'Establecemos su nombre
   oBarraDesplazamientoModelo.Name = "sbKm"

   'Conectamos la forma con el nuevo control
   oBarraDesplazamiento.Control = oBarraDesplazamientoModelo

   'Lo agregamos al formulario
   oFormulario.insertByIndex(0, oBarraDesplazamientoModelo)

   'Agregamos la forma a la página de dibujo para que sea visible
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oBarraDesplazamiento )

End Sub
```

Sus propiedades son muy similares al botón de selección (SpinButton), veamos las propiedades particulares de este control.

```
Sub BarraDesplazamiento2()
Dim oFormulario As Object
```

```
Dim osbKm As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   osbKm = oFormulario.getByName("sbKm")

   With osbKm
         .BlockIncrement = 25              'Cambio grande
         .LineIncrement = 5                'Cambio pequeño
         .ScrollValueMin = 1               'Valor mínimo
         .ScrollValueMax = 1000            'Valor máximo
   End With

End Sub
```

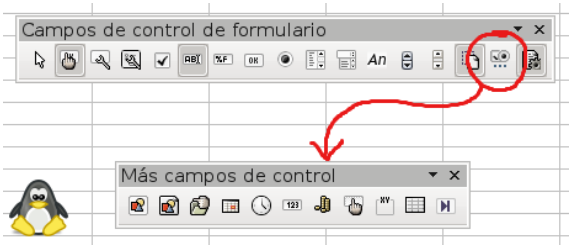
El cambio grande (BlockIncrement) se da cuando presionamos un área libre de la barra de desplazamiento y el cambio pequeño (LineIncrement) cuando presionamos las flechas de los extremos. Para conocer el valor actual de la barra de desplazamiento, usamos.

```
Sub BarraDesplazamiento3()
Dim oFormulario As Object
Dim osbKm As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   osbKm = oFormulario.getByName("sbKm")

   MsgBox "Valor actual = " & osbKm.ScrollValue

End Sub
```

Este control acepta la vinculación con una celda (*ValueBinding*), así como la activación y desactivación (*Enabled*), el envío del foco, así como ocultarlo, se tienen que establecer desde el modo “vista” del control.

7.12 Otros controles

Hasta aquí, hemos visto los controles estándar presentes en la barra “Campos de control de formularios”, pero existen otro conjunto de controles que podemos usar y a los cuales tenemos acceso desde la barra de herramientas “Campos de control de formulario”, como se muestra en la siguiente imagen.



Estos controles amplían las posibilidades para mostrar y controlar nuestros datos, veamos las principales características de cada uno.

7.12.1 Botón gráfico (ImageButton)

Este control nos permite tener una imagen como fondo de él.

```
Sub BotonGrafico1()
Dim oFormulario As Object
Dim oNuevoControl As Object
Dim oNuevoControlModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   Call CambiaTam( oNuevoControl, 5000, 5000 )
   Call CambiaPos( oNuevoControl, 2000, 17000 )

   'Agregamos un nuevo botón gráfico
   oNuevoControlModelo = ThisComponent.createInstance("com.sun.star.form.component.ImageButton")
   oNuevoControlModelo.Name = "ibFoto"
   oNuevoControl.Control = oNuevoControlModelo
   oFormulario.insertByIndex(0, oNuevoControlModelo)
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )

End Sub
```

Y sus principales propiedades.

```
Sub BotonGrafico2()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("ibFoto")

   With oControl
         .ImageURL = ConvertToURL("/home/mau/foto.jpg")      'La ruta de la imagen a mostrar
         .ScaleImage = True                                  'Si se va a escalar la imagen
         .ScaleMode = 2                    'El modo de escalar ( 2 = Ajustar al control)
   End With

End Sub
```

En su modo “vista”, podemos habilitar o deshabilitarlo, así como hacerlo invisible o enviarle el foco.

7.12.2 Control de imagen (ImageControl)

Este control es muy similar al botón gráfico (*ImageButton*), la diferencia, es que a este control le puedes asignar el campo de una fuente de datos (*DataField*), si el campo es una imagen, la mostrará en el control. Para agregar uno al formulario, usamos.

```
Sub ControlImagen1()
Dim oFormulario As Object
Dim oNuevoControl As Object
Dim oNuevoControlModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
```

```
    oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

    Call CambiaTam( oNuevoControl, 6000, 6000 )
    Call CambiaPos( oNuevoControl, 2000, 18000 )

    'Agregamos un nuevo control de imagen
    oNuevoControlModelo = ThisComponent.createInstance(
"com.sun.star.form.component.DatabaseImageControl")
    oNuevoControlModelo.Name = "icProductos"
    oNuevoControl.Control = oNuevoControlModelo
    oFormulario.insertByIndex(0, oNuevoControlModelo)
    ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )

End Sub
```

Puedes asignarle una imagen de forma dinámica.

```
Sub ControlImagen2()
Dim oFormulario As Object
Dim oControl As Object

    oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
    oControl = oFormulario.getByName("icProductos")

    With oControl
          .ImageURL = ConvertToURL("/home/mau/foto.jpg")      'La ruta de la imagen a mostrar
          .ScaleImage = True                                  'Si se va a escalar la imagen
          .ScaleMode = 2                                'El modo de escalar ( 2 = Ajustar al control)
    End With

End Sub
```

En el modo “vista”, puedes enviarle el foco, deshabilitarlo y hacerlo invisible.

7.12.3 Selección de archivo (FileSelection)

Control que nos permite mostrar el cuadro de dialogo común para seleccionar un archivo, solo te devuelve la ruta del archivo, no el archivo en si, para agregar un nuevo control, usamos.

```
Sub SeleccionArchivo1()
Dim oFormulario As Object
Dim oNuevoControl As Object
Dim oNuevoControlModelo As Object

    oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
    oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

    Call CambiaTam( oNuevoControl, 10000, 700 )
    Call CambiaPos( oNuevoControl, 2000, 18000 )

    'Agregamos un nuevo control de selección de archivo
    oNuevoControlModelo = ThisComponent.createInstance("com.sun.star.form.component.FileControl")
    oNuevoControlModelo.Name = "fcArchivo"
    oNuevoControl.Control = oNuevoControlModelo
    oFormulario.insertByIndex(0, oNuevoControlModelo)
    ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )
```

```
End Sub
```

Al contar con un área de texto, puedes usar la mayor parte de las propiedades vistas para los controles que aceptan cadenas (fuente, tamaño, color, etc), puedes establecer una ruta por default, de este modo, cuando se abra el dialogo, se abrirá en este directorio.

```
Sub SeleccionArchivo2()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("fcArchivo")

   oControl.DefaultText = "/home/mau"

End Sub
```

Para devolver el archivo seleccionado, se usa su propiedad Text.

```
Sub SeleccionArchivo3()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("fcArchivo")

   MsgBox "Archivo seleccionado: " & oControl.Text

End Sub
```

Puede recibir el foco y hacerse invisible a través del modo “vista”.

7.12.4 Campo de fecha (DateField)

Este control nos permite trabajar con fechas de una forma muy sencilla, incluso podemos mostrar un calendario para seleccionar fechas. Se agrega de la siguiente manera.

```
Sub CampoFecha1()
Dim oFormulario As Object
Dim oNuevoControl As Object
Dim oNuevoControlModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   Call CambiaTam( oNuevoControl, 5000, 700 )
   Call CambiaPos( oNuevoControl, 2000, 18000 )

   'Agregamos un nuevo campo de fecha
   oNuevoControlModelo = ThisComponent.createInstance("com.sun.star.form.component.DateField")
   oNuevoControlModelo.Name = "dfNacimiento"
   oNuevoControl.Control = oNuevoControlModelo
   oFormulario.insertByIndex(0, oNuevoControlModelo)
```

```
    ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )

End Sub
```

Al ser un campo con texto, en formato de fecha pero no deja de ser texto, cuenta con toda las propiedades para manipularlo (color, tamaño, fuente, etc), ya vistas en otros controles. Veamos las propiedades particulares de este control.

```
Sub CampoFecha2()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("dfNacimiento")

   With oControl
         .DateMin = 19000101          'Fecha mínima
         .DateMax = 20091231          'Fecha máxima
         .DateFormat = 0              'Formato mostrado
         .Dropdown = True             'Si es desplegable
         .Spin = True                 'Si muestra el campo giratorio
   End With

End Sub
```

Observa la forma “sui generis” de establecer los valores de las propiedades para el valor de la fecha mínima (*DateMin*) y máxima (*DateMax*) del control, es un número, pero primero va el año, después el mes y por ultimo el día, si las vas a establecer de forma dinámica, puedes hacerlo a través de una cadena (string), pero tienes que darle el formato exacto, si no, te dará fechas erróneas.

```
.DateMax = Format(Now,"YYYYMMDD")
```

La propiedad para mostrar la flecha desplegable (*Dropdown*), permite al usuario mostrar un calendario para seleccionar la fecha. La propiedad para establecer el formato (*DateFormat*), puedes tomar los siguientes valores enteros.

Nombre	*Valor*	*Ejemplo*
Estándar (corto)	0	20/09/09
Estándar (corto YY)	1	20/09/09
Estándar (corto YYYY)	2	20/09/2009
Estándar (largo)	3	domingo 20 de septiembre de 2009
DD/MM/YY	4	20/09/09
MM/DD/YY	5	09/20/09
YY/MM/DD	6	09/09/20
DD/MM/YYYY	7	20/09/2009
MM/DD/YYYY	8	09/20/2009
YYYY/MM/DD	9	2009/09/20
YY-MM-DD	10	09-09-20
YYYY-MM-DD	11	2009-09-20

Para obtener la fecha actual del control puedes hacerlo de dos maneras.

```
Sub CampoFecha3()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("dfNacimiento")

   'Devuelve la fecha seleccionada como número
   MsgBox oControl.Date
   'Devuelve el contenido del control como texto
   MsgBox oControl.Text

End Sub
```

Dependiendo del formato que hayas establecido, te será más conveniente usar una u otra forma, esto será en función de tus necesidades.

En su modo de “vista”, puedes activarlo o desactivarlo (*Enabled*) y hacerlo invisible. Puedes vincularlo a un campo de datos origen (*DataField*), pero no implementa la posibilidad de vincularlo a una celda, pero más adelante aprenderemos a hacerlo directamente con código.

7.12.5 Campo de hora (TimeField)

Este control es muy similar al campo de fecha (*DateField*) pero para manejar valores de hora, veamos como agregar uno nuevo en nuestro formulario.

```
Sub CampoHora1()
Dim oFormulario As Object
Dim oNuevoControl As Object
Dim oNuevoControlModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   Call CambiaTam( oNuevoControl, 5000, 700 )
   Call CambiaPos( oNuevoControl, 2000, 20000 )

   'Agregamos un nuevo campo de hora
   oNuevoControlModelo = ThisComponent.createInstance("com.sun.star.form.component.TimeField")
   oNuevoControlModelo.Name = "tfEntrada"
   oNuevoControl.Control = oNuevoControlModelo
   oFormulario.insertByIndex(0, oNuevoControlModelo)
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )

End Sub
```

Y sus propiedades particulares.

```
Sub CampoHora2()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
```

```
    oControl = oFormulario.getByName("tfEntrada")

    With oControl
          .TimeMin = 8300000          'La hora mínima 8:30
          .TimeMax = 15000000         'La hora máxima 15:00
          .TimeFormat = 1             '24 hr con segundos
          .Spin = True                'Muestra el control giratorio
    End With
  End Sub
```

El formato para la hora mínima (*TimeMin*) y máxima (*TimeMax*), tiene que ser HoraMinutosSegundos, y los segundos debe ser cuatro dígitos, el formato solo acepta cuatro valores; formato 24 horas sin segundos (0), formato 24 horas con segundos (1), formato 12 horas sin segundos (3) y formato 12 horas con segundos (4). Para recuperar el valor actual del control, tienes dos opciones.

```
  Sub CampoHora3()
  Dim oFormulario As Object
  Dim oControl As Object

     oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
  "MiDirectorio" )
     oControl = oFormulario.getByName("tfEntrada")

     'Devuelve la fecha seleccionada como número
     MsgBox oControl.Time
     'Devuelve el contenido del control como texto
     MsgBox oControl.Text

  End Sub
```

En su modo de “vista”, puedes activarlo o desactivarlo (*Enabled*) y hacerlo invisible. Puedes vincularlo a un campo de datos origen (*DataField*), pero no implementa la posibilidad de vincularlo a una celda.

7.12.6 Campo numérico (NumericField)

Este control permite manejar únicamente números, en el siguiente ejemplo agregamos un nuevo control al formulario.

```
  Sub CampoNumerico1()
  Dim oFormulario As Object
  Dim oNuevoControl As Object
  Dim oNuevoControlModelo As Object

     oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
  "MiDirectorio" )
     oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

     Call CambiaTam( oNuevoControl, 5000, 700 )
     Call CambiaPos( oNuevoControl, 2000, 20000 )

     'Agregamos un nuevo campo numérico
     oNuevoControlModelo = ThisComponent.createInstance("com.sun.star.form.component.NumericField")
     oNuevoControlModelo.Name = "nfPesos"
     oNuevoControl.Control = oNuevoControlModelo
     oFormulario.insertByIndex(0, oNuevoControlModelo)
```

```
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )

End Sub
```

Recuerda que todos los controles donde se muestre información al usuario, comparten un mínimo de propiedades (color, fuente, etc). Las propiedades particulares de este control son.

```
Sub CampoNumerico2()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("nfPesos")

   With oControl
         .ValueMin = 100                    'El valor mínimo
         .ValueMax = 500                    'El valor máximo
         .ValueStep = 2                     'El salto entre valores
         .Spin = True                       'Si muestra el control giratorio
         .ShowThousandsSeparator = True     'Si muestra el separador de miles
         .DecimalAccuracy = 2               'El número de decimales
   End With

End Sub
```

El separador de miles (*ShowThousandsSeparator*), será el mismo configurado en la interfaz del OOo.

```
Sub CampoNumerico3()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("nfPesos")

   'Devolvemos el valor del control
   MsgBox "El valor actual del control es = " & oControl.Value

End Sub
```

Este control si acepta la vinculación con una celda, así como con un campo de datos.

7.12.7 Campo moneda (CurrencyField)

Este control es casi idéntico al campo numérico (*NumericField*), con al diferencia de que puede mostrar un símbolo de moneda en su formato. Para agregar un control de este tipo a nuestro formulario, usamos el siguiente código.

```
Sub CampoMoneda1()
Dim oFormulario As Object
Dim oNuevoControl As Object
Dim oNuevoControlModelo As Object
```

```
   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   Call CambiaTam( oNuevoControl, 5000, 700 )
   Call CambiaPos( oNuevoControl, 2000, 20000 )

   'Agregamos un nuevo campo moneda
   oNuevoControlModelo = ThisComponent.createInstance("com.sun.star.form.component.CurrencyField")
   oNuevoControlModelo.Name = "cfIngresos"
   oNuevoControl.Control = oNuevoControlModelo
   oFormulario.insertByIndex(0, oNuevoControlModelo)
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )

End Sub
```

Las propiedades particulares de este control son.

```
Sub CampoMoneda2()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("cfIngresos")

   With oControl
         .CurrencySymbol = "B$"              'El símbolo de moneda
         .PrependCurrencySymbol = True       'Muestra el símbolo a la izquierda
         .ValueMin = 0                       'El valor mínimo
         .ValueMax = 1000                    'El valor máximo
         .ValueStep = 5                      'El salto entre valores
         .Spin = True                        'Si muestra el control giratorio
         .ShowThousandsSeparator = True      'Si muestra el separador de miles
         .DecimalAccuracy = 2                'El número de decimales
   End With

End Sub
```

Para recuperar el valor actual del control usamos.

```
Sub CampoMoneda3()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("cfIngresos")

   'Devolvemos el valor del control
   MsgBox "El valor actual del control es = " & oControl.Value

End Sub
```

Este control solo acepta el vinculo con un campo de datos.

7.12.8 Campo enmascarado (PatternField)

Este control permite establecer una mascara de entrada para los datos del usuario, es muy útil en entornos con usuarios con poca experiencia, pues permite limitar de forma muy estricta lo que este captura, para agregar uno a nuestro formulario, usamos.

```
Sub CampoEnmascarado1()
Dim oFormulario As Object
Dim oNuevoControl As Object
Dim oNuevoControlModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   Call CambiaTam( oNuevoControl, 5000, 700 )
   Call CambiaPos( oNuevoControl, 2000, 20000 )

   'Agregamos un nuevo campo enmascarado
   oNuevoControlModelo = ThisComponent.createInstance("com.sun.star.form.component.PatternField")
   oNuevoControlModelo.Name = "pfRFC"
   oNuevoControl.Control = oNuevoControlModelo
   oFormulario.insertByIndex(0, oNuevoControlModelo)
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )

End Sub
```

Este control tiene pocas propiedades particulares, pero son vitales para su correcto funcionamiento.

```
Sub CampoEnmascarado2()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("pfRFC")

   With oControl
         .MaxTextLen = 15                        'Longitud máxima del contenido
         .EditMask = "AAAALNNNNNNLCCC"           'Mascara de entrada
         .LiteralMask = "____-______-___"        'Mascara de caracteres
         .StrictFormat = True                    'Control de formato
   End With

End Sub
```

Estas cuatro propiedades, es esencial que trabajen todas juntas, la primera es autodescriptiva pero es muy importante que tenga la longitud exacta de la mascara de entrada (*EditMask*) y de la mascara de caracteres (*LiteralMask*), para que funcione el control de formato (*StrictFormat*) debe estar en verdadero esta propiedad. En la mascará de entrada (*EditMask*) determinamos que puede y que no puede capturar el usuario en esa exacta posición del caracter, de acuerdo a la siguiente tabla.

Caracter	*Restricción*
L	El caracter de esta posición, se muestra tal cual, no puede editarse, al llegar a esta posición, el cursor pasa al siguiente caracter editable.
a	Se permiten letras de la “A” a la “Z”, se respetan mayúsculas y minúsculas.
A	Se permiten letras de la “A” a la “Z”, todas se convierten a mayúsculas.

Caracter	*Restricción*
c	Se permiten letras de la “A” a la “Z”, y todos los dígitos, 0-9, se respetan mayúsculas y minúsculas.
C	Se permiten letras de la “A” a la “Z”, y todos los dígitos, 0-9, todas las letras pasan a mayúsculas.
N	Solo se permiten los números dígitos, 0 a 9.
X	Se permiten todos los caracteres imprimibles.

La mascara de caracteres, es lo que ve el usuario, mientras no capture nada en las respectivas posiciones.

```
Posición                123456789012345
Mascara de entrada      AAAALNNNNNNLCCC
Mascara de caracteres   ____-______-___
```

Para regresar el valor capturado, usamos.

```
Sub CampoEnmascarado3()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("pfRFC")

   MsgBox "Valor actual del control: " & oControl.Text

End Sub
```

Este control soporta la vinculación con un campo de datos (*DataField*).

7.12.9 Cuadro de grupo (GroupBox)

Este control solo es de apoyo visual para enmarcar otros grupos de controles, generalmente se usa con grupos de botones de opción (*OptionButton*) aunque los puedes usar con cualquier otro grupo de controles. Para agregar uno al formulario, usamos.

```
Sub CuadroGrupo1()
Dim oFormulario As Object
Dim oNuevoControl As Object
Dim oNuevoControlModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   Call CambiaTam( oNuevoControl, 5000, 5000 )
   Call CambiaPos( oNuevoControl, 2000, 20000 )

   'Agregamos un nuevo campo enmascarado
   oNuevoControlModelo = ThisComponent.createInstance("com.sun.star.form.component.GroupBox")
   oNuevoControlModelo.Name = "gbEstadoCivil"
   oNuevoControl.Control = oNuevoControlModelo
   oFormulario.insertByIndex(0, oNuevoControlModelo)
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )
```

```
End Sub
```

Para establecer el titulo del control usamos.

```
Sub CuadroGrupo2()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("gbEstadoCivil")

   'Establecemos el titulo del control
   oControl.Label = "Estado civil"

End Sub
```

7.12.10 Control de tablas (TableControl)

Este control nos permite mostrar una tabla, consulta o instrucción SQL para enlazar bases de datos, por ahora solo funciona en tiempo de diseño y el asistente de controles debe estar activado, el formulario previamente debe estar enlazado con los datos origen, para agregar una al formulario, usamos el siguiente código, pero, por ahora, no podrás enlazarlo por código.

```
Sub TablaControl1()
Dim oFormulario As Object
Dim oNuevoControl As Object
Dim oNuevoControlModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   Call CambiaTam( oNuevoControl, 10000, 10000 )
   Call CambiaPos( oNuevoControl, 3000, 20000 )

   'Agregamos un nuevo campo enmascarado
   oNuevoControlModelo = ThisComponent.createInstance("com.sun.star.form.component.GridControl")
   oNuevoControlModelo.Name = "gcContactos"
   oNuevoControl.Control = oNuevoControlModelo
   oFormulario.insertByIndex(0, oNuevoControlModelo)
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )

End Sub
```

7.12.11 Barra de navegación

Este control permite, cuando el formulario esta enlazado a una base de datos, controlar la navegación entre los registros, para agregarla al formulario usamos.

```
Sub BarraNavegacion1()
```

```
Dim oFormulario As Object
Dim oNuevoControl As Object
Dim oNuevoControlModelo As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oNuevoControl = ThisComponent.createInstance("com.sun.star.drawing.ControlShape")

   Call CambiaTam( oNuevoControl, 20000, 1000 )
   Call CambiaPos( oNuevoControl, 3000, 25000 )

   'Agregamos un nuevo campo enmascarado
   oNuevoControlModelo = ThisComponent.createInstance("com.sun.star.form.component.NavigationToolBar")
   oNuevoControlModelo.Name = "navBarra"
   oNuevoControl.Control = oNuevoControlModelo
   oFormulario.insertByIndex(0, oNuevoControlModelo)
   ThisComponent.getCurrentController.getActiveSheet.getDrawPage.add( oNuevoControl )

End Sub
```

Las propiedades particulares de este control son.

```
Sub BarraNavegacion2()
Dim oFormulario As Object
Dim oControl As Object

   oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"MiDirectorio" )
   oControl = oFormulario.getByName("navBarra")

   With oControl
         .BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)    'Color de fondo
         .TextColor = RGB(Rnd*255,Rnd*255,Rnd*255)          'Color de texto
         .Border = 2                                        'Borde plano
         .IconSize = 0                                      'Pequeño = 0, Grande = 1
         .ShowPosition = True                               'Muestra la información actual del cursor
         .ShowNavigation = True                             'Muestra las flechas de navegación
         .ShowRecordActions = True                          'Muestra los iconos para manipular registros
         .ShowFilterSort = True                             'Muestra los iconos para ordenar y filtrar
   End With

End Sub
```

Si estableces las ultimas cuatro propiedades en falso (*False*), solo veras un marco sin ninguna utilidad, generalmente se muestran los cuatro, el icono para manipular los registros (*ShowRecordActions*), te permite, agregar, editar y borrar registros, en ocasiones es útil deshabilitar estos y controlar estas acciones por código, esto te permite un mayor control de los datos, pero requiere, también, muchas más líneas de código, y trabajo, por supuesto, tu decides que es lo que más te conviene.

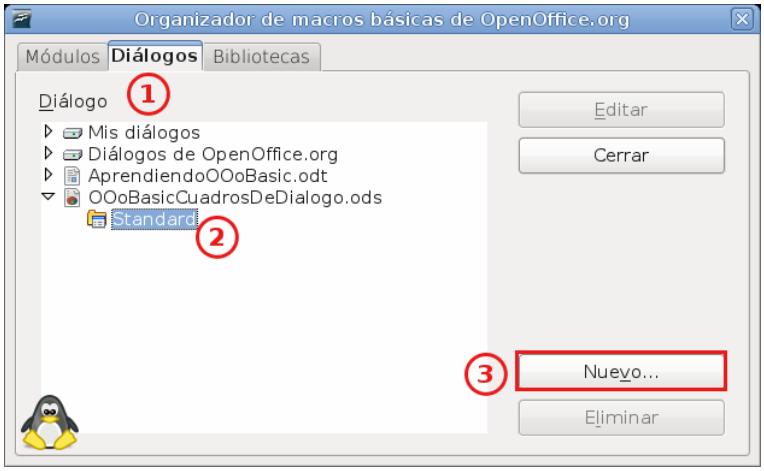
Con esto terminamos con el manejo de formularios y sus controles, hemos visto la forma de agregarlos por código y manipular las principales propiedades de cada uno, casi todo el conocimiento adquirido en este capitulo, nos será de mucha utilidad para el siguiente, en donde aprenderemos el manejo de cuadros de diálogo.

8 Trabajando con cuadros de dialogo

Los cuadros de dialogo son similares a los formularios, se diferencian en que estos se muestran en una ventana independiente y los formularios están incrustados en la hoja de calculo, también, con los formularios se puede interactuar con las celdas y los controles directamente así como vincularlos, los cuadros de dialogo, al estar dentro de su ventana, es preciso cerrarla primero para regresar a la hoja de calculo, otra diferencia importante es que los formularios están más orientados a la interacción con orígenes de datos, pues cuentan con propiedades y características especiales para ello, los cuadros de dialogo son más de propósito general aunque con un poco de ingenio también los puedes poner a trabajar como si de una interfaz de datos se tratara. Los controles disponibles para los cuadros de dialogo son en su mayoría, los mismos que ya hemos visto en los formularios por lo que solo nos centraremos en las diferencias importantes.

Para agregar un cuadro de dialogo tenemos dos formas, la primera: ve al menú *Herramientas | Macros | Organizar Macros | OpenOffice.org Basic...*, que te mostrará.

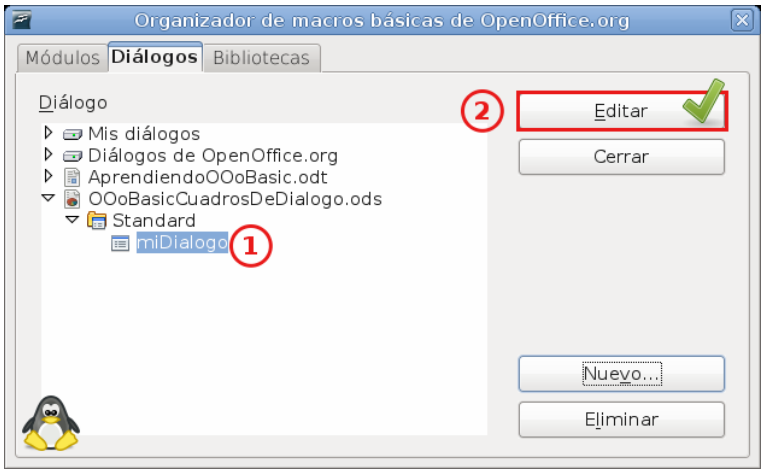
Da un clic en el botón de comando *Administrar...*, tienes que seleccionar la ficha Diálogos (1), después, el archivo y la biblioteca donde quieras agregar el nuevo cuadro de diálogo (2), por ultimo, da un clic al botón de comando *Nuevo...* (3).



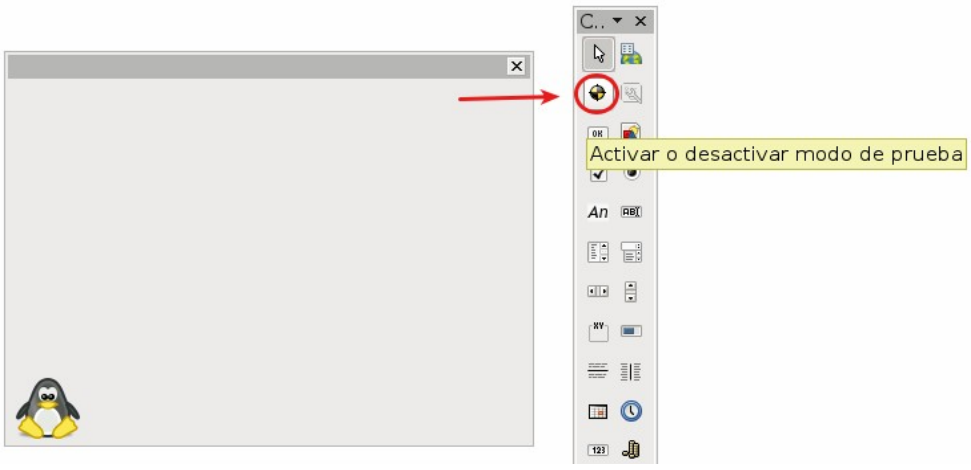
Donde puedes establecer el nombre del nuevo cuadro de dialogo, para nuestro ejemplo “miDialogo” esta bien. Generalmente el nombre del cuadro de dialogo va en relación al

trabajo para el cual lo usaremos y este nombre, sigue las mismas premisas para nombrar las variables. Debes de recordar que en tiempo de ejecución el nombre distingue mayúsculas de minúsculas. Después de establecer el nombre, da un clic en el botón de comando *Aceptar,* para regresar al cuadro de diálogo anterior, donde podrás ver ya, nuestro nuevo cuadro de diálogo.

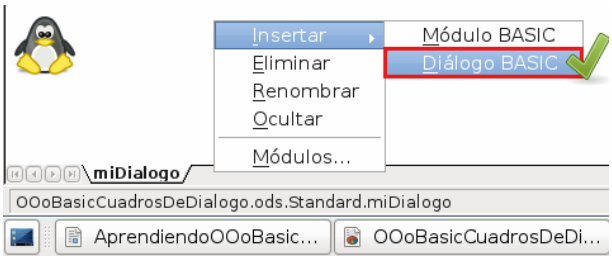
Donde podemos ver el nuevo cuadro de diálogo (1), solo nos resta darle clic en el botón de comando *Editar* (2), para que nos lleve a nuestro conocido y ya buen amigo IDE, que es donde diseñaremos el contenido y estructura de nuestro flamante cuadro de diálogo (1).



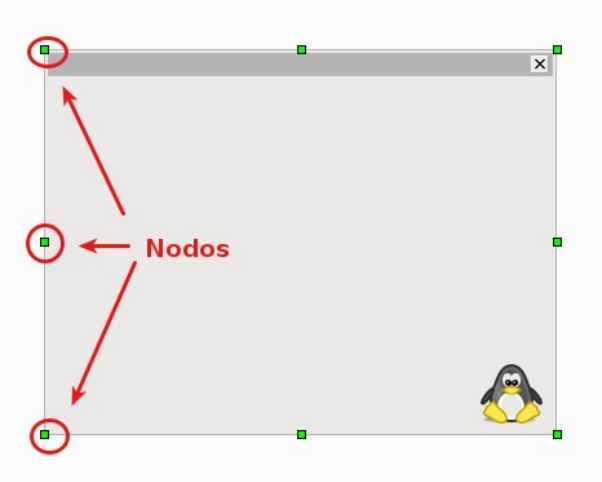
Incluso, ya lo puedes probar con el icono de *Activar o desactivar modo de prueba* (2).



Si no ves la barra de herramientas de controles, activala desde el menú *Ver | Barras de Herramientas*. La segunda forma de agregar un cuadro de diálogo, es desde el mismo IDE, en al área de pestañas de los módulos, da un clic con el botón secundario del ratón, para ver.



Cuando agregas un nuevo cuadro de diálogo, de forma predeterminada se posiciona en el centro de la ventana de trabajo, para moverlo tienes que seleccionarlo desde cualquier borde de el, ten cuidado por que este borde es un poco pequeño y si eres como yo, poco hábil con el ratón, las primeras veces es un poco difícil, no quiero contarte cuanto tiempo estuve peleándome para averiguar como se movía de posición el cuadro de diálogo pues intentaba arrastrarlo del centro. Tanto para seleccionarlo, cambiarlo de posición y de tamaño se hace desde el borde de este, te darás cuenta que ha sido seleccionado, por que veras sus nodos verdes para manipularlo, para cambiar de tamaño, arrastra cualquier de los ocho nodos verdes, para cambiar de posición, arrastra desde cualquier borde donde no haya algún nodo verde.



Los controles que agregues al cuadro de diálogo, **no están anclados a el**, es decir, si mueves el cuadro de diálogo, solo se moverá este, para mover todo el contenido, con el ratón, arrastra desde una esquina a la esquina contraria, pero cuidado, tienes que hacerlo, desde "fuera" del cuadro de diálogo, de modo que abarques todo el contenido, ahora si, puedes mover el cuadro de diálogo y sus controles. El icono de prueba solo te sirve para verlo como lo verá el usuario, pero no funcionará ningún código que este asociado a los controles.

Para mostrar un cuadro de diálogo por código, usamos.

```
Sub EjecutarMiDialogo1()
Dim oDialogo As Object

   'Cargamos la librería Standard en memoria
   DialogLibraries.LoadLibrary( "Standard" )
   'Cargamos el cuadro de diálogo en memoria
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("miDialogo") )
   'Lo ejecutamos (mostramos)
   oDialogo.execute()
   'Lo liberamos de memoria
   oDialogo.dispose()

End Sub
```

La función *CreateUnoDialog* forma parte del OOo Basic y se encarga de crear y asignar el cuadro de diálogo en memoria en tiempo de ejecución. Es importante que primer cargues en memoria (*LoadLibrary*) la librería donde esta el cuadro de diálogo, si no, te dará un error en tiempo de ejecución, sobre todo cuando muestras cuadros de diálogo al abrir archivos, como aprenderemos más adelante. La variable de objeto *DialogLibraries*, siempre apunta a la librería de diálogos desde donde se llame la macro, el nombre de la librería debe existir dentro de dicha variable, lo cual siempre puedes verificar.

```
    Msgbox DialogLibraries.hasByName("Standard")
```

El método para mostrar el cuadro de diálogo (*execute*), te puede devolver un valor, como también comprobaremos más adelante, es importante que sepas que mientras el cuadro de diálogo este abierto, la línea de ejecución “permanecerá” en la línea donde se llame a este método, como puedes comprobarlo en el siguiente ejemplo.

```
Sub EjecutarMiDialogo2()
Dim oDialogo As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("miDialogo") )
   MsgBox "Estas apunto de mostrar un cuadro de diálogo"
   oDialogo.execute()
   MsgBox "Has cerrado el cuadro de diálogo"
   oDialogo.dispose()

End Sub
```

Observa que el segundo mensaje, no se muestra hasta que cierras el cuadro de diálogo. Dentro de las macros predeterminadas que incorpora OOo, existe una función especial para cargar cuadros de diálogo, se llama *LoadDialog* y esta en la librería *Tools*, la cual, por supuesto, debes cargar primero para poder usar la función, como en el siguiente ejemplo.

```
Sub EjecutarMiDialogo3()
Dim oDialogo As Object

   'Cargamos la librería incorporada Tools de OOo
   GlobalScope.BasicLibraries.LoadLibrary("Tools")
   'Cargamos el cuadro de diálogo usando la función LoadDialog
   oDialogo = LoadDialog("Standard", "miDialogo", DialogLibraries )
   'Mostramos
   oDialogo.execute()
   'Liberamos
   oDialogo.dispose()

End Sub
```

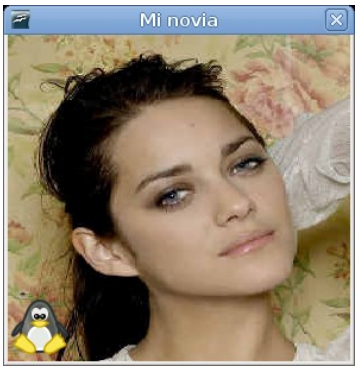
Usa el método que gustes, pero siempre asegúrate de que todos los objetos existan. Vamos a cambiar algunas propiedades del cuadro de dialogo.

```
Sub EjecutarMiDialogoo4()
Dim oDialogo As Object
Dim oDialogoModelo As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("miDialogo") )

   'Accedemos al modelo del objeto
   oDialogoModelo = oDialogo.getModel()
```

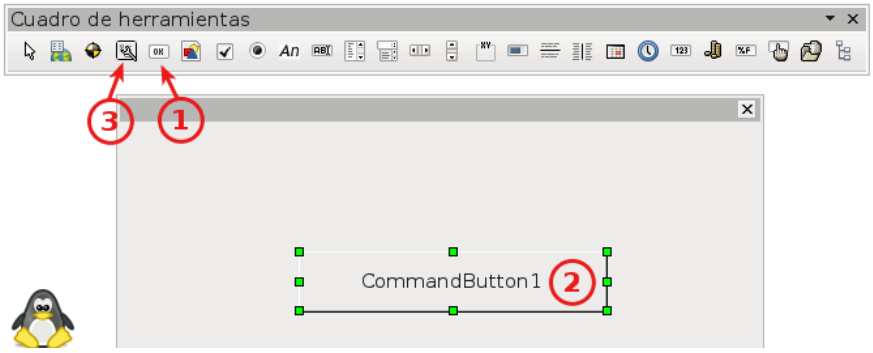
```
    With oDialogoModelo
          .Title = "Mi nuevo cuadro de diálogo"                    'El titulo del cuadro de diálogo
          .BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)          'El color de fondo
          .PositionX = 100                                         'Posición desde la izquierda
          .PositionY = 100                                         'Posición desde la parte superior
          .Width = 200                                             'El ancho
          .Height = 200                                            'El alto
    End With

    oDialogo.execute()
    oDialogo.dispose()

End Sub
```

Es importa que recuerdes que estas propiedades solo se establecen en tiempo de ejecución, es decir, “reemplazan” temporalmente a las establecidas en tiempo de diseño, pero quedan sin efecto al cerrar el cuadro de diálogo. La posición predeterminada del cuadro de diálogo al mostrarse, será la que establezcas en el IDE. La unidad para la posición en X (*PositionX*), posición en Y (*PositionY*), el ancho (*Width*) y el alto (*Height*), es pixeles, por lo que estarán determinadas por la resolución de tu pantalla, la posición del objeto, va en relación con el extremo superior izquierdo del documento que lo contiene, no de la pantalla. También puedes establecer una imagen como fondo, si lo haces, veras la imagen, pero no el color de fondo.

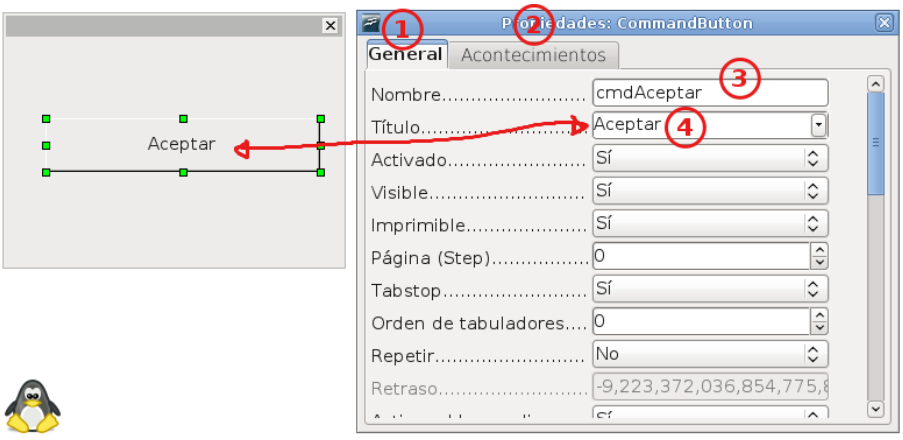
```
Sub EjecutarMiDialogo5()
Dim oDialogo As Object
Dim oDialogoModelo As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("miDialogo") )

   'Accedemos al modelo del objeto
   oDialogoModelo = oDialogo.getModel()
   With oDialogoModelo
         .Title = "Mi novia"
         .ImageURL = ConvertToUrl("/home/mau/foto.jpg")      'Imagen de fondo
   End With
   oDialogo.execute()
   oDialogo.dispose()

End Sub
```



Todos los controles que veremos aquí, los agregaremos de forma manual de la misma forma. Primero seleccionamos el control deseado (1), lo dibujamos dentro del cuadro de diálogo (2), puedes seguir agregando más controles del mismo tipo, cambiar de control o seleccionar el control recién agregado (2). Con el control seleccionado, puedes mostrar sus propiedades con el botón secundario del ratón o con el icono de propiedades (3) de la barra de herramientas.



La ventana de propiedades, nos permite hacer dos cosas, establecer la mayoría de las propiedades (1) iniciales para el control (incluyendo los cuadros de diálogo) y establecer las macros que asociaremos a sus acontecimientos (2), también llamados eventos, lo cual aprenderemos en el siguiente capítulo. Una propiedad muy importante de cualquier control, es su nombre (3) ya que este será el que usaremos para hacer referencia a el en tiempo de ejecución, otras son lo que verá el usuario (4) y otras determinarán su apariencia y comportamiento.



Para acceder a un control dentro de un formulario, usamos su nombre (3) de la siguiente forma.

```
Sub EjecutarMiDialogo6()
Dim oDialogo As Object
Dim oDialogoModelo As Object
Dim ocmdAceptar As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("miDialogo") )

   'Referencia al botón de comando cmdAceptar
   ocmdAceptar = oDialogo.getControl("cmdAceptar")
   'Cambiamos su titulo
   ocmdAceptar.Label = "Cerrar"

   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

Si el control no existe, al tratar de hacer referencia a el en una variable, no te dará ningún error, el error te lo dará hasta que trates de establecer alguna propiedad y esta no exista.

```
    'Aquí no te dará error
    ocmdAceptar = oDialogo.getControl("NoExisto")
    'Te dará el error aquí
    ocmdAceptar.Label = "Cerrar"
```

Lo común es que, al tu establecer el nombre del control al agregarlo manualmente al control de diálogo, no tengas problemas con los nombres en el código, aun así, si necesitas establecerlo desde alguna otra fuente, puedes verificar que exista el control, antes de establecerlo a la variable, como en el siguiente ejemplo.

```
Sub EjecutarMiDialogo7()
Dim oDialogo As Object
Dim ocmdAceptar As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("miDialogo") )

   'Verificamos que exista el control
   If oDialogo.getModel.hasByName("NoExisto") Then
         ocmdAceptar = oDialogo.getControl("NoExisto")
         'Cambiamos su titulo
         ocmdAceptar.Label = "Cerrar"
         oDialogo.execute()
         oDialogo.dispose()
   Else
         MsgBox "El control no existe"
   End If

End Sub
```

Observa que el método para consultar la existencia de un elemento del conjunto (*hasByName*), esta en el "modelo" del objeto, en el capítulo anterior, con la referencia directa a un objeto de un formulario, teníamos el "modelo" de este y para acceder al modo "vista", teníamos que hacerlo enlazando con el controlador del contender, es decir, del documento.

```
    otxtNombre = oFormulario.getByName("txtNombre")
    'Accedemos a la vista del control
    otxtNombreVista = oDoc.getCurrentController.getControl( otxtNombre )
```

En los cuadro de diálogo es al reves, la referencia directa al control se establece en modo "vista" y si queremos acceder al "modelo" tenemos que hacerlo de forma explicita.

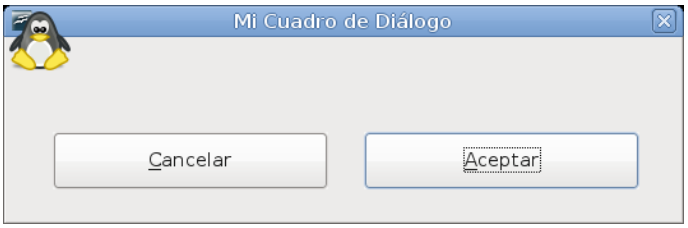
```
Sub EjecutarMiDialogo8()
Dim oDialogo As Object
Dim ocmdAceptar As Object
Dim ocmdAceptarModelo As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("miDialogo") )

   ocmdAceptar = oDialogo.getControl("cmdAceptar")
   'Accedemos al modelo del objeto
   ocmdAceptarModelo = ocmdAceptar.getModel()
   'Establecemos el título
   ocmdAceptarModelo.Label = "Cerrar"

   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

Solo que aquí es de una forma más sencilla, pues hay un método especifico para ello (getModel), puedes usar una nueva variable o acceder a través de este método directamente.

La mayoría de las propiedades y métodos de los controles vistos en el capítulo de los formularios, son aplicables a los controles de los cuadros de diálogo, la mayoría de las cuales no repetiremos aquí, solo repasaremos las más importantes, las que tengan un uso o aplicación diferente y algún nuevo control presente solo en los cuadros de diálogo, también, recuerda que cada control debes de agregarlo de forma manual a nuestro cuadro de diálogo de trabajo. La gran mayoría de las propiedades de los controles que veremos, por lo regular se establecen en tiempo de diseño, se muestran en tiempo de ejecución para comprobar que es posible manipularlas, pero lo común es que pocas de ellas cambien durante el desarrollo de un programa, pero estas pocas son muy útiles en diversas circunstantes como los demostraremos más adelante.

8.1 Botón de comando (CommandButton)

Agrega un segundo botón de comando y nombralo “cmdCancelar”, de modo que tu cuadro de diálogo te quede de la siguiente manera.

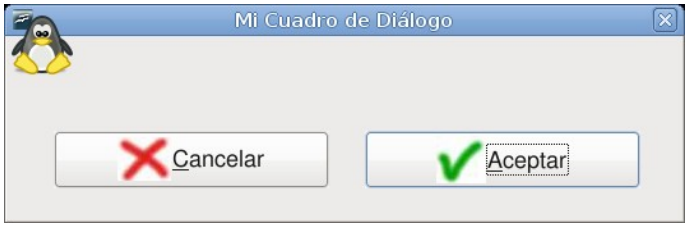


Las principales propiedades de este control son.

```
Sub BotonComando1()
Dim oDialogo As Object
Dim ocmdBoton As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("miDialogo") )

   ocmdBoton = oDialogo.getControl("cmdAceptar")
   With ocmdBoton.getModel
         .Align = 1                    'Alineación; 0 = izquierda, 1 = centro, 2 = derecha
         .FontHeight = 12              'Tamaño de fuente
         .FontName = "FreeSans"        'Nombre de la fuente
         .ImageURL = ConvertToUrl("/home/mau/bien.jpg")      'Imagen a mostrar
         .ImagePosition = 1            'Posición 1 = centro izquierda
         .PushButtonType = 1           'Tipo de botón 1 = Aceptar
   End With

   ocmdBoton = oDialogo.getControl("cmdCancelar")
   With ocmdBoton.getModel
         .Align = 1
         .FontHeight = 12
         .FontName = "FreeSans"
         .ImageURL = ConvertToUrl("/home/mau/mal.jpg")
         .ImagePosition = 1
         .PushButtonType = 2           'Tipo de botón 2 = Cancelar
   End With
   oDialogo.execute()
   oDialogo.dispose()
```

```
End Sub
```

Si estableces una imagen para el botón de comando, esta se mostrará junto con el texto para el usuario (*Label*), claro, esta propiedad puede estar vacía y solo mostrar la imagen.



Establecer el tipo de botón (*PushButtonType*), nos permite asociar de forma directa, sin código, los botones a algunas teclas, si es igual a 1, este botón se presionara con la tecla entrar (*Enter)*, si es igual a 2, el botón se ejecutará al presionar la tecla escape (*Esc*) además de que cerrara el cuadro de diálogo.

Establece esta propiedad, según corresponda su nombre para cada botón en tiempo de diseño, en la interfaz del usuario, esta propiedad se llama “Tipo de botón” y ejecuta varias veces la siguiente macro, teniendo cuidado de usar, para salir, una vez cada uno de los botones, también prueba a cerrar desde el icono de ventana.

```
Sub BotonComando2()
Dim oDialogo As Object
Dim Res As Integer

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("miDialogo") )

   'El método execute, retorna un valor
   Res = oDialogo.execute()

   Select Case Res
         Case 0 : MsgBox "El usuario selecciono Cancelar"
         Case 1 : MsgBox "El usuario selecciono Aceptar"
   End Select
   oDialogo.dispose()

End Sub
```

Observa que; no hemos asociado todavía código alguno a cada botón, aun así, podemos salir del cuadro de diálogo, por que hemos establecido la propiedad “Tipo de botón” en tiempo de diseño, nota como ahora, asociamos el método para mostrar el cuadro de diálogo (*execute*) a una variable, con lo podemos saber que botón presiono el usuario, esto, es solo una técnica, otra seria usar una variable global, como aprenderemos más adelante.

8.2 Control gráfico (ImageControl)

Este control nos permite mostrar archivos de imagen, agrega uno al cuadro de diálogo desde la barra de herramientas y llámalo “*icFoto*”. Su propiedad más importante es la que nos permite establecer la imagen a mostrar.

```
Sub ControlImagen1()
Dim oDialogo As Object
Dim oControl As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   oControl = oDialogo.getControl("icFoto")
   With oControl.getModel
         .ImageURL = ConvertToUrl("/home/mau/foto2.jpg")          'Imagen a mostrar
   End With
   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

Este control, usado en tiempo de ejecución, nos permite hacer cosas muy divertidas, como mostrar las fotos de tus contactos si estas haciendo una agenda, o la imagen de tus productos si tienes un inventario, solo por citar dos ejemplos.

8.3 Casilla de verificación (CheckBox)

Agrega un control y nombralo “chkImprimir”, generalmente, este control se usa para valores de verdadero o falso, aunque tiene una propiedad para establecer hasta tres estados.

```
Sub CasillaVerificacion1()
Dim oDialogo As Object
Dim oControl As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'El control de imagen
   oControl = oDialogo.getControl("icFoto")
   'Lo ocultamos
   oControl.Visible = False

   'La casilla de verificación
   oControl = oDialogo.getControl("chkImprimir")

   With oControl.getModel
         .TriState = True       'Estado triple
   End With
   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

Observa como hacemos referencia al control de imagen agregado en nuestro ejemplo anterior y lo ocultamos. El estado triple de una casilla de verificación, te permite establecer tres estados a este control: seleccionado, no seleccionado y sin selección. Para recuperar su valor usamos.

```
Sub CasillaVerificacion2()
Dim oDialogo As Object
Dim oControl As Object
```

```
    DialogLibraries.LoadLibrary( "Standard" )
    oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

    oControl = oDialogo.getControl("chkImprimir")
    'Recuperamos el valor del control
    Msgbox oControl.State

End Sub
```

Nota como recuperamos el valor del control sin mostrar el cuadro de diálogo, aunque lo usual es mostrar el cuadro de diálogo para que el usuario manipule los controles y entonces si, recuperar sus valores.

8.4 Cuadro de grupo (FrameControl)

Agrega un control cuadro de grupo y llámalo “fraDistribucion”. Este control nos permite agrupar a otros controles, excepto para los botones de opción donde si tienen inferencia, para los demás controles solo es estético, una ayuda visual para separar grupos de controles. Sus principales propiedades son.

```
Sub CuadroGrupo1()
Dim oDialogo As Object
Dim oControl As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'Ocultamos los controles anteriores
   oControl = oDialogo.getControl("icFoto")
   oControl.Visible = False
   oControl = oDialogo.getControl("chkImprimir")
   oControl.Visible = False

   'El cuadro de grupo
   oControl = oDialogo.getControl("fraDistribucion")

   With oControl.getModel
         .FontName = "FreeSans"            'Nombre de la fuente
         .FontHeight = 12                  'Tamaño de la fuente
         .TextColor = RGB(Rnd*255,Rnd*255,Rnd*255)    'Color de la fuente
   End With
   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

Observa como vamos ocultado los controles anteriores usados, puedes seguir ocultando los que no usemos, por ahora, o puedes hacer más grande el cuadro de diálogo para que quepan más controles, queda a tu criterio. En el capítulo anterior, formularios, para agrupar y controlar los botones de opción, era suficiente con establecerles el mismo nombre por grupo, en los cuadros de diálogo, **no es posible** usar el mismo nombre para más de un control, por ello, en este caso, es importante el uso de cuadros de grupo, como lo demostraremos a continuación.

8.5 Botón de opción (OptionButton)

Agrega tres controles de botón de opción (*OptionButton*) y nombralos “optUbuntu”, “optFedora” y “optArch”. Establece el segundo como seleccionado. Es muy importante que los agregues “dentro” del cuadro de grupo (*FrameControl*) creado en el tema anterior, de modo que tengas algo así.

Para acceder al botón de opción que este seleccionado, usamos.

```
Sub BotonOpcion1()
Dim oDialogo As Object
Dim oControl As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'Ocultamos los controles anteriores
   oControl = oDialogo.getControl("icFoto") : oControl.Visible = False
   oControl = oDialogo.getControl("chkImprimir") : oControl.Visible = False

   'El primero botón de opción
   oControl = oDialogo.getControl("optUbuntu")
   MsgBox oControl.State

   'El segundo botón de opción
   oControl = oDialogo.getControl("optFedora")
   MsgBox oControl.State

   'El tercer botón de opción
   oControl = oDialogo.getControl("optArch")
   MsgBox oControl.State

   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

Por supuesto, ya lo notaste, si mostramos el valor (*State*) de los botones de opción, antes de mostrar el cuadro de diálogo, siempre tendremos el mismo botón seleccionado, el que hayas establecido como predeterminado, para este ejemplo, por ahora, solo recuerda como obtenemos el estado (*State*), en el siguiente capítulo, aprenderemos una forma muy elegante de como consultar el estado de grupos de botones de opción, sin tener que acceder uno por uno. Cada grupo de botones de opción que agregues, formara un grupo al cual puedes acceder por su índice, como en.

```
Sub BotonOpcion2()
Dim oDialogo As Object
Dim oGrupo() As Object
Dim oBoton As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'Accedemos al primer grupo de controles
   oDialogo.getModel.getGroup( 0, oGrupo, "" )
   'Iteramos en cada botón
   For Each oBoton In oGrupo()
         MsgBox oBoton.Name
   Next

End Sub
```

Observa como devolvemos el grupo deseado, en este caso el cero, en la variable (*oGrupo*) pasada como argumento al método (*getGroup*), esta forma regresar un valor en uno de los argumentos de la función, es muy común en la programación en C, C++ y otros lenguajes, en los derivados de Basic no es muy común, pero como vez, se presenta algunas veces.

8.6 Etiqueta (Label)

Las etiquetas, a pesar de tener la capacidad de asociarles macros, son comúnmente usadas como controles estáticos, agrega una nueva etiqueta y llámala “lblTitulo”. Entre otras de sus propiedades, puedes cambiar en tiempo de ejecución, las siguientes.

```
Sub Etiqueta1()
Dim oDialogo As Object
Dim oControl As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'La etiqueta
   oControl = oDialogo.getControl("lblTitulo")
   With oControl.getModel
         .Label = "Bienvenido al programa"            'El titulo
         .FontHeight = 18
         .TextColor = RGB(Rnd*255,Rnd*255,Rnd*255)
   End With
   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

En no pocas ocasiones, puedes usar una etiqueta para mostrar ayuda contextual al usuario de forma dinámica.

8.7 Campo de texto (TextField)

Agrega un control de cuadro de texto y llámalo “txtNombre”. Estos controles son los más comúnmente usados para que el usuario introduzca información a procesar.

```
Sub CuadroTexto1()
Dim oDialogo As Object
Dim oControl As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'El cuadro de texto
   oControl = oDialogo.getControl("txtNombre")
   'Mostramos el cuadro de dialogo
   oDialogo.execute()
   'Al cerrar mostramos el contenido del control
   MsgBox oControl.Text
   oDialogo.dispose()
```

```
End Sub
```

Observa en el ejemplo que consultamos el valor del control, “después” de que se cierra el cuadro de diálogo, recuerda que esto puede ser correcto, siempre y cuando la información ya este validada, esto es muy importante. Este control comparte la mayoría de las propiedades vistas en los formularios, excepto la capacidad de mostrar texto enriquecido.

8.8 Cuadro de lista (ListBox)

Agrega un control de cuadro de lista y nombralo “lstPaises”.

```
Sub CuadroLista1()
Dim oDialogo As Object
Dim oControl As Object
Dim mDatos()

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   mDatos() = Array("México","España","Argentina","Colombia","Panama")
   'El cuadro de lista
   oControl = oDialogo.getControl("lstPaises")
   'Agregamos elementos
   oControl.getModel.StringItemList = mDatos()
   oDialogo.execute()
   'Mostramos el elemento seleccionado
   MsgBox oControl.SelectedItem
   oDialogo.dispose()

End Sub
```

Todos los métodos vistos en el capitulo de formularios, con este control, aplican para su uso dentro de cuadros de diálogo.

8.9 Cuadro combinado (ComboBox)

Agrega un control de cuadro combinado y llámalo “cboColor”.

```
Sub CuadroCombinado1()
Dim oDialogo As Object
Dim oControl As Object
Dim mDatos()

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   mDatos() = Array("Gris","Rojo","Amarillo","Verde","Azul")
   'El cuadro combinado
   oControl = oDialogo.getControl("cboColor")
   'Agregamos elementos
   oControl.getModel.StringItemList = mDatos()
   oDialogo.execute()
```

```
    'Mostramos el elemento seleccionado
    MsgBox oControl.Text
    oDialogo.dispose()

End Sub
```

Puedes usar las mismas propiedades y métodos vistos en el capítulo de formularios, recordando que este control comparte la mayoría de estas con el control cuadro de lista.

8.10 Barra de desplazamiento (ScrollBar)

Agrega un control de barra de desplazamiento y llámalo “sbMover”.

```
Sub BarraDesplazamiento1()
Dim oDialogo As Object
Dim oControl As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'La barra de desplazamiento
   oControl = oDialogo.getControl("sbMover")
   oDialogo.execute()

   'Mostramos el valor seleccionado
   MsgBox oControl.Value
   oDialogo.dispose()

End Sub
```

En la barra de herramientas de controles, veras dos controles de este tipo, uno horizontal y otro vertical, en realidad es el mismo control, solo que su propiedad de orientación (*Orientation*) esta establecida en uno y otro valor, si cambias esta propiedad, debes de intercambiar, también, los valores de ancho y alto del control, esto es por que no se redimensiona automáticamente.

```
Sub BarraDesplazamiento2()
Dim oDialogo As Object
Dim oControl As Object
Dim Ancho As Long

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'La barra de desplazamiento
   oControl = oDialogo.getControl("sbMover")
   'Cambiamos la horientacion a vertical
   oControl.Orientation = 1
   'Guardamos el ancho
   Ancho = oControl.getModel.Width
   'Asignamos el nuevo ancho
   oControl.getModel.Width = oControl.getModel.Height
   'Establecemos el nuevo alto
   oControl.getModel.Height = Ancho

   oDialogo.execute()
   'Mostramos el valor seleccionado
```

```
    MsgBox oControl.Value
    oDialogo.dispose()

End Sub
```

Las demás propiedades, son las mismas vistas en los formularios.

8.11 Barra de progreso (ProgressBar)

Agrega un control barra de progreso en nuestro cuadro de diálogo y llámalo “pbAvance”. Este control no esta presente en los formularios, por lo que veremos sus principales propiedades aquí.

```
Sub BarraProgreso1()
Dim oDialogo As Object
Dim oControl As Object

    DialogLibraries.LoadLibrary( "Standard" )
    oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

    'La barra de desplazamiento
    oControl = oDialogo.getControl("pbAvance")

    With oControl.getModel()
          .BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)    'Color de fondo
          .FillColor = RGB(Rnd*255,Rnd*255,Rnd*255)          'Color de relleno
          .ProgressValueMax = 100                            'Valor máximo
          .ProgressValueMin = 1                              'Valor mínimo
    End With
    oDialogo.execute()
    oDialogo.dispose()

End Sub
```

Este control esta pensado para mostrar el avance (progreso) de una tarea al usuario, por ejemplo, las líneas importadas de un archivo, la exportación de información o cualquier otro proceso que tenga un inicio y un fin, es decir, que generalmente sepamos cuando empieza y cuando termina para poder mostrar el avance como en el siguiente ejemplo.

```
Sub BarraProgreso2()
Dim oDialogo As Object
Dim oControl As Object
Dim co1 As Integer

    DialogLibraries.LoadLibrary( "Standard" )
    oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

    'La barra de desplazamiento
    oControl = oDialogo.getControl("pbAvance")

    With oControl.getModel()
          For co1 = .ProgressValueMin To .ProgressValueMax
               'Asignamos el valor a la barra de progreso
               .ProgressValue = co1
               'Esperamos 10 milisegundos
              Wait 10
```

```
        Next col
    End With
    oDialogo.execute()
    oDialogo.dispose()

End Sub
```

Antes de que te enojes conmigo con justa razón, te explico, “aparentemente”, no veras ningún efecto, salvo, si eres observador, que al mostrar el cuadro de diálogo, la barra de progreso se mostrará “llena”, solo quise mostrarte el uso común en un rutina para la barra de progreso, generalmente, los valores máximos y mínimos se ajustan en tiempo de ejecución para adaptarse al proceso que estamos procesando, y el valor actual de la barra (*ProgressValue*), es el que va cambiando con el tiempo, en el siguiente capitulo veremos más ejemplos de este control en usos más cotidianos. También, es frecuente que este control se muestre al usuario, solo, mientras muestra el avance, después, se oculta, claro, esto, es a tu gusto y criterio, aunque, como buen programador, antes del gusto, satisface las necesidades del usuario.

8.12 Línea (FixedLine)

Este control es solo cosmético, cierto, le puedes asignar macros a los eventos que soporta, pero no es usual, generalmente solo se agrega para separar controles o secciones dentro del cuadro de diálogo, agrega un control de línea y llámala “linSepara”. Este control tiene pocas propiedades.

```
Sub Linea1()
Dim oDialogo As Object
Dim oControl As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'La línea
   oControl = oDialogo.getControl("linSepara")
   With oControl.getModel()
         .Orientation = 0                                    'Orientación horizontal = 0
         .Label = "Opciones"                                 'Solo en líneas horizontales
         .BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)     'Color de fondo
         .TextColor = RGB(Rnd*255,Rnd*255,Rnd*255)           'Color de texto
   End With
   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

Al igual que con las barras de desplazamiento, las líneas horizontales y verticales son el mismo control, solo cambia la propiedad orientación (*Orientation*), si cambias la orientación de la línea a vertical (*Orientation = 1*), el texto (*Label*) no se mostrará y tienes que intercambiar el ancho por alto para mantener la proporción del control.

```
Sub Linea2()
Dim oDialogo As Object
Dim oControl As Object
Dim Ancho As Long

   DialogLibraries.LoadLibrary( "Standard" )
```

```
    oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

    'La línea
    oControl = oDialogo.getControl("linSepara")
    'Cambiamos la horientacion a vertical
    oControl.getModel.Orientation = 1
    'Guardamos el ancho
    Ancho = oControl.getModel.Width
    'Asignamos el nuevo ancho
    oControl.getModel.Width = oControl.getModel.Height
    'Establecemos el nuevo alto
    oControl.getModel.Height = Ancho
    oDialogo.execute()
    oDialogo.dispose()

End Sub
```

Al contar con texto para mostrar al usuario, este control cuenta con todas las propiedades propias de los controles con texto (fuente, tamaño, color, estilo, etc), por lo que puedes establecerlas también.

8.13 Control de archivo (Filecontrol)

Este control nos permite seleccionar la ruta y nombre de cualquier archivo, solo nos devolverá la ruta seleccionada por el usuario, teniendo que agregar cualquier código necesario para manipular dicho archivo. Agrega un control de archivo y llamalo “fcArchivo”

```
Sub ControlArchivo1()
Dim oDialogo As Object
Dim oControl As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'El control de archivo
   oControl = oDialogo.getControl("fcArchivo")
   With oControl.getModel()
         .BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)    'Color de fondo
         .TextColor = RGB(Rnd*255,Rnd*255,Rnd*255)          'Color del texto
         .FontName = "Liberation Sans"                      'Nombre de la fuente
         .FontHeight = 14                                   'Tamaño de la fuente
         .Text = "/home/mau/Desktop"                        'Ruta predeterminada
   End With
   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

La ruta que establezcas en la propiedad texto (*Text*), será la ruta donde se abrirá el cuadro de diálogo de selección de archivo cuando el usuario presionar el botón Examinar... de este control, esta misma propiedad te sirve para recuperar la ruta de archivo seleccionado por el usuario, no obstante, siempre valida que el archivo exista.

8.14 Control de árbol (TreeControl)

Este control solo esta presente en los cuadros de diálogo, sirve para mostrar elementos jerarquizados, con la posibilidad de expandirse y contraerse. Agrega un nuevo control de árbol y llámalo “tcDirectorios”. Sus principales propiedades son.

```
Sub ControlArbol1()
Dim oDialogo As Object
Dim oControl As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'El control de árbol
   oControl = oDialogo.getControl("tcDirectorios")
   With oControl.getModel()
         .BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)    'Color de fondo
         .RootDisplayed = True                              'Si mostramos la raíz
         .Border = 2                                        'Tipo de borde 2 = plano
         .Height = 170                                      'Alto
         .Width = 125                                       'Ancho
         .PositionX = 10                                    'Posición X (desde izquierda)
         .PositionY = 120                                   'Posición Y (desde arriba)
               'Si se imprime el control
         .SelectionType = 1                                 'Tipo de selección
   End With
   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

El tipo de selección (*SelectionType*), determinará la forma de selección con el ratón o teclado, sus valores son: ninguno (0), sencillo (1), solo se selecciona un elemento, múltiple (2), se pueden seleccionar varios elementos, incluso alternados con apoyo de la tecla CTRL y rango (3) donde podemos arrastrar y seleccionar varios elementos.

Para agregar datos a este control, es necesario apoyarnos en otro servicio.

```
Sub ControlArbol2()
Dim oDialogo As Object
Dim oControl As Object
Dim oTDM As Object
Dim oRaiz As Object
Dim oPadre As Object
Dim oHijo As Object
Dim co1 As Integer

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   'El control de árbol
   oControl = oDialogo.getControl("tcDirectorios")
   'Servicio para controlar el contenido del control
   oTDM = createUnoService("com.sun.star.awt.tree.MutableTreeDataModel")
   'Creamos el nodo raíz
   oRaiz = oTDM.createNode( "Raíz", True )
   'Establecemos el nodo raíz
   oTDM.setRoot( oRaiz )

   For co1 = 1 To 5
         'Creamos un nodo padre
         oPadre = oTDM.createNode( "Padre " & co1, True )
```

```
        'Lo asignamos a la raíz
        oRaiz.appendChild(oPadre)
        'Creamos un nodo hijo
        oHijo = oTDM.createNode("Hijo " & co1, True )
        'Lo asignamos al nodo padre
        oPadre.appendChild( oHijo )
    Next co1

    'Asginamos los datos al modelo del control
    oControl.getModel.DataModel = oTDM
    'Altura de la fila de datos
    oControl.getModel.RowHeight = 25
    'Mostramos el cuadro de diálogo
    oDialogo.execute()
    'Mostramos el nodo seleccionado
    MsgBox oControl.getSelection.getDisplayValue

    oDialogo.dispose()

End Sub
```

Al cerrar el cuadro de diálogo, te tiene que mostrar el nodo seleccionado, si no hay ningún nodo seleccionado, te dará un error, para evitarlo, primero consulta el número de selecciones que hay.

```
    If oControl.getSelectionCount > 0 Then
        'Mostramos el nodo seleccionado
        MsgBox oControl.getSelection.getDisplayValue
    End If
```

En siguiente ejemplo, a partir de la ruta especificada, buscaremos todos los directorios contenidos y los agregaremos como nodos.

```
Sub ControlArbol3()
Dim oDialogo As Object
Dim oControl As Object
Dim oTDM As Object
Dim oRaiz As Object
Dim oPadre As Object
Dim sRuta As String

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   oControl = oDialogo.getControl("tcDirectorios")
   'Servicio para controlar el contenido del control
   oTDM = createUnoService("com.sun.star.awt.tree.MutableTreeDataModel")

   'La ruta para devolver directorios
   sRuta = "/home/mau/"
   If Dir(sRuta) <> "" Then
        'Creamos el nodo raíz
        oRaiz = oTDM.createNode( sRuta, True )
        'Establecemos el nodo raíz
        oTDM.setRoot( oRaiz )
        'Buscamos el primer directorio (16)
        sRuta = Dir(sRuta,16)
        'Cuando no haya más directorios sRuta estará vacía
        Do While sRuta <> ""
            Select Case Left(sRuta,1)
                'Nos saltamos los ocultos
                Case ".", ".."
```

```
                    Case Else
                          oPadre = oTDM.createNode( sRuta, True )
                          oRaiz.appendChild(oPadre)
                End Select
                'Siguiente directorio
                sRuta = Dir
          Loop
          oControl.getModel.DataModel = oTDM
    End If
    oDialogo.execute()

    If oControl.getSelectionCount > 0 Then
          'Mostramos el nodo seleccionado
          MsgBox oControl.getSelection.getDisplayValue
    End If
    oDialogo.dispose()

End Sub
```

Por supuesto, puedes hacer que se muestren tantos subdirectorios como quieras, en el siguiente capítulo resolveremos esta cuestión, pero puedes ir tratando de encontrar el ¿como?. A los nodos, también es posible agregarles una imagen, como en el siguiente ejemplo.

```
Sub ControlArbol4()
Dim oDialogo As Object
Dim oControl As Object
Dim oTDM As Object
Dim oRaiz As Object
Dim oPadre As Object
Dim oHijo As Object
Dim co1 As Integer
Dim mRutas(1) As String

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("MiDialogo") )

   oControl = oDialogo.getControl("tcDirectorios")
   oTDM = createUnoService("com.sun.star.awt.tree.MutableTreeDataModel")
   oRaiz = oTDM.createNode( "Raíz", True )
   oTDM.setRoot( oRaiz )
   'Matriz con las rutas de las imagenes de ejemplo
   mRutas(0) = ConvertToUrl("/home/mau/bien.jpg")
   mRutas(1) = ConvertToUrl("/home/mau/mal.jpg")
   For co1 = 1 To 2
         oPadre = oTDM.createNode( "Padre " & co1, True )
         oRaiz.appendChild(oPadre)
         oHijo = oTDM.createNode("Hijo " & co1, True )
         'Asignamos la ruta de la imagen
         oHijo.setNodeGraphicURL( mRutas(co1-1) )
         oPadre.appendChild( oHijo )
   Next co1

   oControl.getModel.DataModel = oTDM
   oControl.getModel.RowHeight = 25
   oDialogo.execute()

   If oControl.getSelectionCount > 0 Then
         MsgBox oControl.getSelection.getDisplayValue
   End If

   oDialogo.dispose()

End Sub
```

Este control cuenta con varios métodos más que nos permiten controlarlo completamente, pero estos métodos, los aprenderemos en el siguiente capítulo, donde aplicaremos todos los conocimientos vistos a los largo de este libro.

8.15 Otros controles

Todos los siguientes controles, son muy similares, son cuadros de texto, con alguna característica especial para algún tipo de dato, sus propiedades son las mismas vistas en el capítulo de formularios, y la forma de acceder a ellos, es por su nombre, igual que con cualquier otro control dentro del cuadro de diálogo, como lo hemos venido practicando en este tema.

Estos controles son:

- Campo de fecha (DateField)
- Campo de hora (TimeField)
- Campo numérico (NumericField)
- Campo de moneda (CurrencyField)
- Campo formateado (FormattedFiel)
- Campo enmascarado (PatternField)

9 Trabajando con eventos

Hemos llegado a un tema medular en la programación con OOo, la que responde a la importante pregunta, ¿cuando?, cuando es que queremos que se ejecuten nuestras macros, en este capítulo trataremos de dar respuesta a esta pregunta, para ello, sabe que en estos dos últimos temas, pondremos en practica la mayoría de los temas y conocimientos vistos hasta ahora, los repasaremos y confirmaremos, por eso, te recomiendo fehacientemente que, si tienes alguna duda en algún tema, lo repases con calma y si aun hay dudas, usa las listas de correo o el foro para resolverlas, pero es importante, que no tengas dudas hasta aquí.

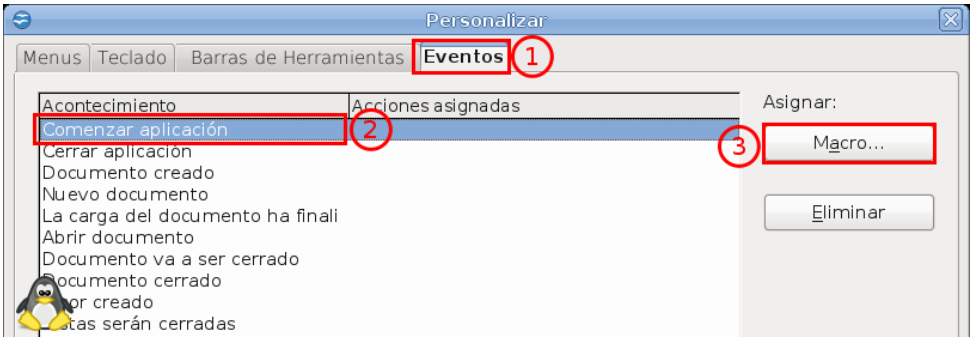
A los eventos, también se les conoce como "*sucesos*", nosotros usaremos el nombre de evento, por ser el que se usa en OOo. Los primeros eventos que aprenderemos, son los relacionados con la aplicación, es decir, con el programa completo.

9.1 Eventos de la aplicación

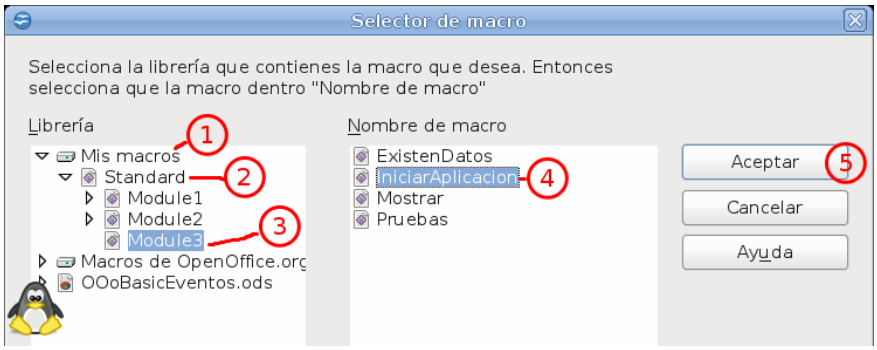
Agrega la siguiente macro al archivo especial "Mis Macros", es importante que quede en este archivo para que trabaje correctamente.

```
Sub IniciarAplicacion()

   MsgBox "Hola, bienvenido a OOo"

End Sub
```

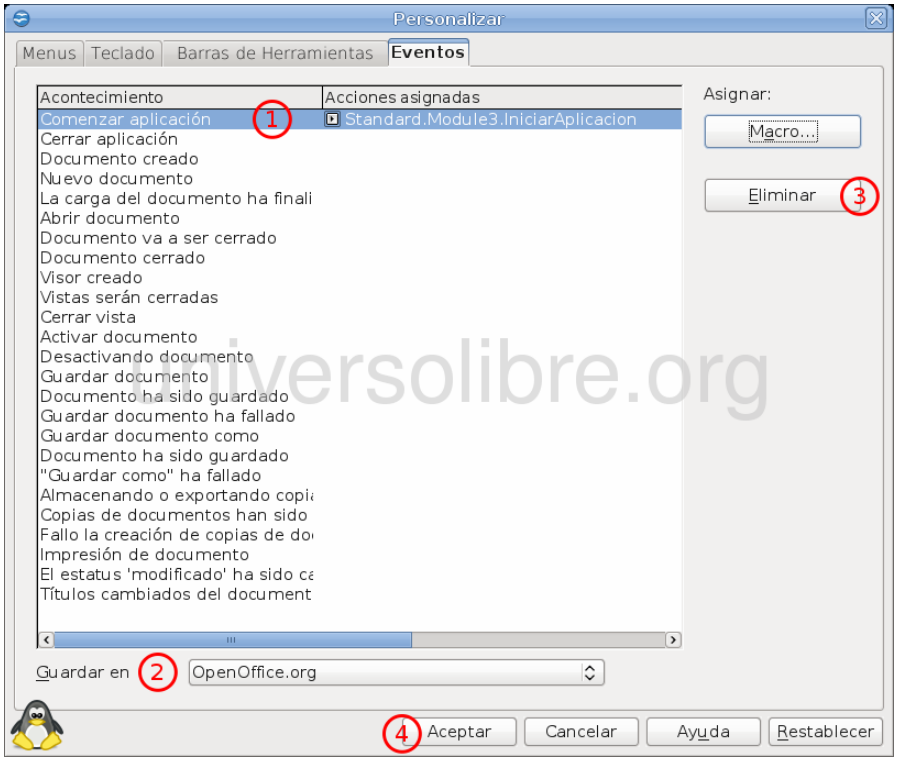
Ahora, ve al menú Ver | Barra de herramientas | Personalizar... En el cuadro de diálogo "Personalizar", selecciona la ficha "Eventos" (1), selecciona el evento "Comenzar Aplicación" (2), da un clic en el botón de comando "Macro..." (3).



En el cuadro de diálogo "Selector de macros", selecciona "Mis Macros" (1), después la librería (2) donde hayas creado el modulo (3) con la macro (4), selecciona la macro y por ultimo da un clic en el botón de comando "*Aceptar*" (5).



El cuadro de diálogo “Selector de macros” se cerrará y veras de nuevo, el de “Personalizar”, en donde puedes ver la como hemos asignado la macro seleccionada al evento deseado (1). Asegurate de que, esta asignación, quede guardada en OpenOffice.org (2). En este cuadro de diálogo, también puedes eliminar cualquier asignación establecida, con el botón de comando “*Eliminar*” (3). Para finalizar, da clic en el botón de comando “Aceptar” (4).



Guarda y cierra todos los archivos abiertos que tengas de OpenOffice.org, incluyendo el entorno de desarrollo (*IDE*) si es que lo tienes abierto, inmediatamente después, abre cualquier aplicación del mismo. Tienes que ver.

Nota que el mensaje solo aparece una vez cuando abres la primer aplicación, esto es, por que OpenOffice.org, es un “contenedor” de las demás aplicaciones. Cuando trabajes con eventos, es muy importante que tengas muy claro **que puedes y que no puedes hacer**, para ir aclarando este concepto tan importante, modifica la macro para que quede de la siguiente manera.

```
Sub IniciarAplicacion()

   MsgBox "Hola, bienvenido a OOo"

   MsgBox ThisComponent.dbg_properties

End Sub
```

De nuevo, cierra “todo” (recuerda que todo incluye al IDE), y vuelve a abrir Calc o cualquier aplicación de OpenOffice.org. Después de ver el primer mensaje de bienvenida, tienes que ver el siguiente mensaje.

Como ya hemos aprendido en estas notas, “ThisComponent” apunta al archivo desde donde se llama a la macro, pero en este caso, nos dice “*desconocido*” (*Unknown*), por que aun no tenemos ningún archivo en memoria. Prueba a ejecutar la macro, ya estando abierta cualquier aplicación o archivo para que veas la diferencia de mensaje que te muestra.

Vuelve a modificar la macro para que quede de la siguiente manera.

```
Sub IniciarAplicacion()

   MsgBox "Hola, bienvenido a OOo"

   MsgBox ThisComponent.dbg_properties

   MsgBox StarDesktop.dbg_properties

End Sub
```

Una vez más, cierra todo y abre tu aplicación favorita. Ahora, en el tercer mensaje, tienes que ver las propiedades del objeto StarDesktop. ¿Y para que podría servir ejecutar una macro al inicio de la aplicación?, esta pregunta podría tener muchas respuestas, todo dependerá de tus necesidades, como por ejemplo, abrir algún archivo (o archivos) necesarios para tu trabajo.

```
Sub IniciarAplicacion()
Dim sRuta As String
Dim oDoc As Object
Dim mArg()

   'Reemplaza esta ruta por la de tu archivo
   sRuta = ConvertToUrl( "/home/mau/Utilerias.ods" )
   oDoc = StarDesktop.loadComponentFromURL( sRuta, "_blank", 0, mArg() )

End Sub
```

O llevar un registro de uso en un simple archivo de texto.

```
Sub IniciarAplicacion()
Dim sRuta As String
Dim sInfo As String
Dim iLibre As Integer
```

```
    sInfo = "OpenOffice.org se ejecutó el día " & Format(Date, "ddd, dd-mmm-yy") & " a las " &
Format(Now, "HH:mm:ss")
    sRuta = ConvertToUrl("/home/mau/registro.log")
    'Si el archivo no existe se crea
    iLibre = FreeFile
    Open sRuta For Append As #iLibre
    Print #iLibre, sInfo
    Close #iLibre

End Sub
```

Recuperar datos de tu intranet o de Internet, asegurarte de que existan archivos indispensables, pueden ser otras útiles formas de asignar una macro al inicio de la aplicación, de nuevo, tus necesidades serán tus rectoras al respecto.

Ahora, veamos el segundo evento de la lista; “Cerrar aplicación”, para ello, asigna la siguiente macro a este vento, asegurándote de guardar esta asignación en OpenOffice.org.

```
Sub CerrarAplicacion()

    MsgBox "Cerrando aplicación"

End Sub
```

Para ver el mensaje, cierra “todo”, incluyendo el IDE, al cerrar el ultimo archivo o aplicación abierta, tienes que ver dicho mensaje. Este evento lo puedes usar para cerrar conexiones a recursos externos, o para complementar el registro de actividad, solo como ejemplos de muchas posibilidades.

```
Sub CerrarAplicacion()
Dim sRuta As String
Dim sInfo As String
Dim iLibre As Integer

    sInfo = "OpenOffice.org se cerró el día " & Format(Date, "ddd, dd-mmm-yy") & " a las " &
Format(Now, "HH:mm:ss")
    sRuta = ConvertToUrl("/home/mau/registro.log")
    'Si el archivo no existe sé crea
    iLibre = FreeFile
    Open sRuta For Append As #iLibre
    Print #iLibre, sInfo
    Close #iLibre

End Sub
```

Al siguiente evento “Crear documento”, asignale la siguiente macro, de nuevo, cuidando de que quede guardada la asignación en OpenOffice.org.

```
Sub CrearDocumento()

    MsgBox "Se ha creado un nuevo documento"

End Sub
```

Prueba a crear nuevos documentos de la aplicación que quieras, en todos tienes que ver el mensaje de la macro, incluso, en los nuevos documentos que se crean por código. Ahora, prueba la siguiente variante de la macro.

```
Sub CrearDocumento()

   'MsgBox "Se ha creado un nuevo documento"
   MsgBox ThisComponent.getTitle

End Sub
```

Nota que hemos deshabilitado la primer línea y agregado otra. Esta vez, tienes que ver correctamente el nombre del nuevo documento creado. Con lo que estamos seguros de que la variable *ThisComponent*, realmente apunta al nuevo documento creado, con lo cual, podemos controlar lo que queramos hacer con él.

Nuestro siguiente evento se llama “Abrir documento”, aquí la macro para asociar.

```
Sub AbrirDocumento()

   MsgBox "Se ha abierto un documento"

End Sub
```

Con estos dos últimos eventos vistos, ya no es necesario cerrar todo, así que es fácil probarlos. Creo que la diferencia entre ellos es clara, uno solo se “desencadena”, cuando se crean documentos nuevos y el otro, solo al abrir documentos existentes, incluso, cuando se crean o abren por código. Modifica la macro anterior para que quede así.

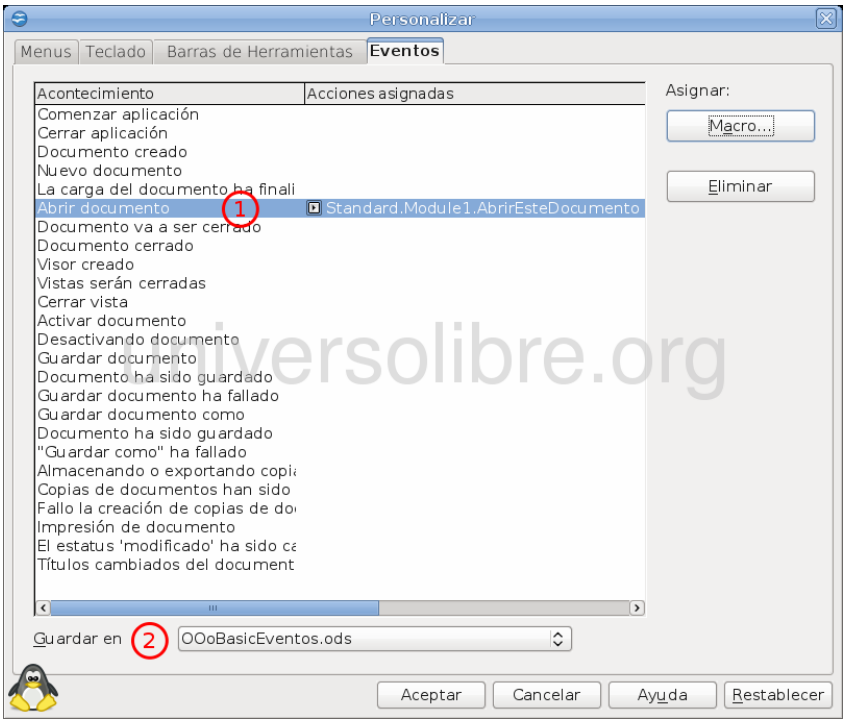
```
Sub AbrirDocumento()

   'MsgBox "Se ha abierto un documento"
   MsgBox ThisComponent.getTitle

End Sub
```

Al abrir documentos, veras el nombre de este, con lo cual, comprobamos que en este evento, también podemos tener control sobre los archivos abiertos.

Pero este evento puede tener una variante interesante, copia la siguiente macro.

```
Sub AbrirEsteDocumento()

   MsgBox "Esta macro solo se ejecuta, cuando se abre este archivo: " & ThisComponent.getTitle

End Sub
```

Y asociala con el evento “Abrir documento” (1), pero, esta vez **no guardes** esta asignación en OpenOffice.org, si no en algún documento existente, en mi caso, la guardo en el archivo “OooBasicEventos.ods” (2), como te muestro a continuación.



Cierra y abre el archivo donde hayas hecho la asignación, tienes que ver dos mensajes en pantalla, el primero, es de la macro asociada al evento “Abrir documento” con la asignación guardada en OpenOffice.org.

El segundo, es el mensaje asociado al mismo evento; “Abrir documento”, pero de la asignación guardada en dicho archivo.

Abre más archivos para que notes la diferencia. Reescribe la macro asociada al documento para que quede de la siguiente manera.

```
Sub AbrirEsteDocumento()
Dim oCelda As Object

   'MsgBox "Esta macro solo se ejecuta, cuando se abre este archivo: " & ThisComponent.getTitle
   'Referencia a la celda A1 de la primer hoja del documento
   oCelda = ThisComponent.getSheets.getByIndex(0).getCellByPosition(0,0)
   'Aumentamos en uno el valor de dicha celda
   oCelda.setValue( oCelda.getValue() + 1 )

End Sub
```

El ejemplo anterior, es una muestra sencilla de lo que podemos hacer asignando macros a los eventos. El criterio para guardar la asignación, en OpenOffice.org o dentro de un documento, estará, casi siempre, determinada por tus necesidades y dominio del lenguaje, pero sobre todo, por tu criterio, el cual, te recomiendo, “debe” estar enfocado, sustentado, guiado, por las necesidades de los usuarios finales de tu aplicación, no, de las del programador, así como por las posibilidades (y limitantes) del lenguaje y entorno de desarrollo. Si guardas la asignación de la macro en un documento, también puedes tener la macro en dicho documento, pero, dependiendo de tus necesidades, a veces será conveniente y a veces no, lo cual tienes que evaluar y determinar en su momento.

Regresa a la macro del evento “Abrir documento”, la que su asignación la hemos guardado en OpenOffice.org y reescribela de la siguiente forma.

```
Sub AbrirDocumento( Event )

   'MsgBox "Se ha abierto un documento"
   'MsgBox ThisComponent.getTitle
   MsgBox Event.Source.getTitle

End Sub
```

Pruebala y nota que obtenemos el mismo resultado de la línea inmediata anterior (comentada en esta versión de la macro), con la diferencia, de que hemos accedido al objeto de una nueva forma. Todas las macros que se asignan a eventos, cuentan con un argumento (*Event*), que, puede o no ir, pero si existe, a través este argumento puedes acceder al "**objeto**" (*Source*) que "**desencadeno**" el evento, esto, es una forma muy versátil de trabajar con macros y eventos, como lo iremos demostrando en las siguientes líneas. El nombre de este argumento, realmente puede ser cualquier palabra válida como nombre de variable, pero es un estándar de "facto", usar el nombre de "Event", aunque nada te impide llamarlo "Evento" si así es tu gusto.

Veamos el siguiente evento que se llama "Guardar documento como". Asigna la siguiente macro a este evento y guarda esta asignación en OpenOffice.org.

```
Sub GuardarDocumentoComo( Evento )

   MsgBox "Se desencadenó el evento Guardar Documento Como en el archivo: " & Evento.Source.getTitle

End Sub
```

Observa como hemos usado la palabra "Evento" para el argumento y este sigue funcionando correctamente. El mensaje que tienes que ver, incluirá el nombre del archivo original "**antes**" de guardarse. Veamos inmediatamente el siguiente evento, que es muy ilustrativo del orden de los eventos. Asigna la siguiente macro al evento; "El documento se ha guardado como" y guarda la asignación en OpenOffice.org.

```
Sub ElDocumentoSeHaGuardadoComo( Evento )

   MsgBox "Se desencadenó el evento El Documento Se Ha Guardado Como, el archivo se guardó con el
nombre: " & Evento.Source.getTitle

End Sub
```

Guarda cualquier archivo nuevo o existente, usando el menú *Archivo | Guardar como...*, las macros anteriores asignadas a sus respectivos eventos, te tienen que mostrar el nombre del archivo original y el nombre del archivo nuevo. El primer evento se desencadena "antes" de guardar definitivamente el archivo, y el segundo evento, cuando ya el archivo fue guardado correctamente. Si el usuario, en el cuadro de diálogo para seleccionar el nuevo nombre del archivo, selecciona cancelar la operación, ninguno de estos dos eventos se ejecutara. Estos dos eventos, también los puedes asignar y guardar en un documento, puedes incluso, asignar y guardar tanto en OpenOffice.org como en el documento, pero toma en cuenta que si haces esto, se ejecutarán dos veces cada evento, lo cual, puede o no ser correcto, de nuevo, dependiendo de tus necesidades.

Los dos eventos siguientes, son similares a estos últimos, con la diferencia, de que se desencadenan simplemente al guardar un archivo ya existente. Asigna la siguiente macro al evento; "Guardar documento" y guarda su asignación donde quieras.

```
Sub GuardarDocumento( Evento )

   MsgBox "El documento " & Evento.Source.getTitle & " se va a guardar"
```

```
End Sub
```

Y la siguiente macro, asignala al evento; “El documento se ha guardado”

```
Sub ElDocumentoSeHaGuardado( Evento )

   MsgBox "El documento " & Evento.Source.getTitle & " se guardo correctamente"

End Sub
```

Si guardaste la asignación en OpenOffice.org, veras el nombre de cada archivo que guardes, y si la guardaste en un documento, siempre verás el nombre del mismo archivo. Nota que un evento se desencadena “antes” de guardar y el otro “después” de guardar.

Los dos siguiente eventos, se desencadenan al cerrar un documento. El evento “Cerrar documento” se ejecuta primero.

```
Sub CerrarDocumento( Evento )

   MsgBox "El documento " & Evento.Source.getTitle & " se va a cerrar"

End Sub
```

Y después el evento “El documento se esta cerrando”.

```
Sub ElDocumentoSeEstaCerrando( Evento )

   MsgBox "El documento " & Evento.Source.getTitle & " se cerro correctamente"

End Sub
```

Aunque por los nombres, parecería que se ejecuta primero el segundo, no es así. En el evento “El documento se esta cerrando”, no es recomendable hacer manipulaciones del documento, pues este, ya estará cerrado.

El siguiente evento se llama “Activar documento”, cuya macro de prueba para ver su funcionamiento es la siguiente, asignala a este evento y guardala en OpenOffice.org.

```
Sub ActivarDocumento( Evento )

   MsgBox "Se activo el documento: " & Evento.Source.getTitle

End Sub
```

Cambia entre documentos abiertos y te mostrará el mensaje, pero, si creas nuevos documentos o abres archivos existentes y no has desactivado las macros de los eventos “Abrir documento” o “Crear documento”, notarás que, aparentemente, ya no se ejecuta el evento, pues no te muestra el mensaje de estos, si no solo el del evento “Activar documento”, esto no significa que los demás eventos se desactiven, para probarlo, modificalos de modo que queden así.

Para el evento “Crear documento”

```
Sub CrearDocumento( Evento )
```

```
   Call GuardarInfo ( Evento )

End Sub
```

Para el evento “Abrir documento”

```
Sub AbrirDocumento( Evento )

   Call GuardarInfo ( Evento )

End Sub
```

Para el evento “Activar documento”

```
Sub ActivarDocumento( Evento )

   Call GuardarInfo ( Evento )

End Sub
```

Y la macro para guardar la información.

```
Sub GuardarInfo( Evento )
Dim sRuta As String
Dim sInfo As String
Dim iLibre As Integer

   sInfo = "Documento: " & Evento.Source.getTitle & ", Evento: " & Evento.EventName & ", " &
Format(Date, "ddd, dd-mmm-yy") & " - " & Format(Now, "HH:mm:ss")
   sRuta = ConvertToUrl("/home/mau/registro.log")
   'Si el archivo no existe se crea
   iLibre = FreeFile
   Open sRuta For Append As #iLibre
   Print #iLibre, sInfo
   Close #iLibre
End Sub
```

Crea un nuevo documento y abre un nuevo documento, después, abre el archivo de registro, donde tienes que ver algo similar a:

```
Documento: Sin título 3, Evento: OnFocus, mar, 13-oct-09 - 11:58:17
Documento: Sin título 3, Evento: OnNew, mar, 13-oct-09 - 11:58:17
Documento: Origen.ods, Evento: OnFocus, mar, 13-oct-09 - 11:58:24
Documento: Origen.ods, Evento: OnLoad, mar, 13-oct-09 - 11:58:24
```

Nota como primero se llama al evento “Activar documento” (*OnFocus*) y después al evento “Crear documento” (*OnNew*) para el caso de documentos nuevos y para documentos existentes, también se llama primero al evento “Activar documento” (*OnFocus*) y después al evento “Abrir documento” (*OnLoad*), una diferencia entre estos eventos, es que los eventos “Crear documento” (*OnNew*) y “Abrir documento” (*OnLoad*), solo se ejecutan una vez y el evento “Activar documento” (*OnFocus*), se ejecuta cada vez que el usuario cambia de archivo.

Nuestro siguiente evento, nos servirá, también, para ilustrar el orden en que se llaman a los eventos, asigna la siguiente macro al evento “Desactivar documento” guardando esta asociación en OpenOffice.org.

```
Sub DesActivarDocumento( Evento )

   Call GuardarInfo ( Evento )

End Sub
```

Y modifica también los siguiente eventos; “Cerrar documento” y “El documento se esta cerrando”, de modo que queden así.

```
Sub CerrarDocumento( Evento )

   Call GuardarInfo ( Evento )

End Sub

Sub ElDocumentoSeEstaCerrando( Evento )

   Call GuardarInfo ( Evento )

End Sub
```

Dependiendo del orden en que actives o desactives tus archivos, en tu registro se verá algo así.

```
Documento: Sin título 3, Evento: OnUnfocus, mar, 13-oct-09 - 12:38:03
Documento: Sin título 2, Evento: OnFocus, mar, 13-oct-09 - 12:38:03
Documento: Sin título 2, Evento: OnUnfocus, mar, 13-oct-09 - 12:38:04
Documento: Sin título 3, Evento: OnFocus, mar, 13-oct-09 – 12:38:04
```

Observa como se alterna entre los eventos “Activar documento” (*OnFocus*) y “Desactivar documento” (*OnUnfocus*) de los archivos, para el caso de cerrar el archivo.

```
Documento: Origen.ods, Evento: OnPrepareUnload, mar, 13-oct-09 - 12:38:18
Documento: Origen.ods, Evento: OnUnload, mar, 13-oct-09 – 12:38:18
Documento: Sin título 2, Evento: OnFocus, mar, 13-oct-09 - 12:38:18
```

Observa que primero se ejecuta el evento “Cerrar documento” (*OnPrepareUnload*) y después el evento “El documento se esta cerrando” (*OnUnload*), aquí los nombres confunden un poco. Observa como, **no** se ejecuta el evento “Desactivar documento” (*OnUnfocus*) del archivo que se esta cerrando, si no que pasa directamente al evento activar (*OnFocus*) del siguiente archivo abierto.

Nuestro siguiente evento es “Imprimir documento”, este evento se ejecuta “antes” de enviar la impresión. Asigna la siguiente macro a este evento.

```
Sub ImprimirDocumento( Evento )

   Call GuardarInfo( Evento )

End Sub
```

Que mostrará en nuestro registro, así que no te podrán decir que no imprimieron el documento, ¿verdad?.

```
Documento: Sin título 2, Evento: OnPrint, mar, 13-oct-09 – 13:24:24
```

Por ultimo, el evento “Se ha cambiado el estado 'Modificado'”, cuya macro de prueba es la siguiente, guardala en OpenOffice.org.

```
Sub DocumentoModificado( Evento )

   Call GuardarInfo( Evento )

End Sub
```

Antes de ver el registro, al documento activo, hazle algunos cambios, puedes incluso cambiar de documento, vuelve de nuevo al documento y guardalo, debes de ver algo similar en tu registro.

```
Documento: OOoBasicEventos.ods, Evento: OnModifyChanged, mar, 13-oct-09 - 14:02:19
Documento: OOoBasicEventos.ods, Evento: OnUnfocus, mar, 13-oct-09 - 14:02:48
Documento: OOoBasicEventos.ods, Evento: OnFocus, mar, 13-oct-09 - 14:02:49
Documento: OOoBasicEventos.ods, Evento: OnModifyChanged, mar, 13-oct-09 - 14:03:02
```

Este evento (*OnModifyChanged*) solo se ejecuta cuando modificamos un documento, pero ojo, solo con la primer modificación, y vuelve a llamarse, cuando guardamos el documento.

Usar un archivo de registro, aparte de saber que fue lo que hizo el usuario, es muy útil para depurar una macro, cuando trabajes con eventos, es mejor usar un archivo de registro para saber que valores toman las variables en las macros, que usar el observador del IDE, o usar el método MsgBox.

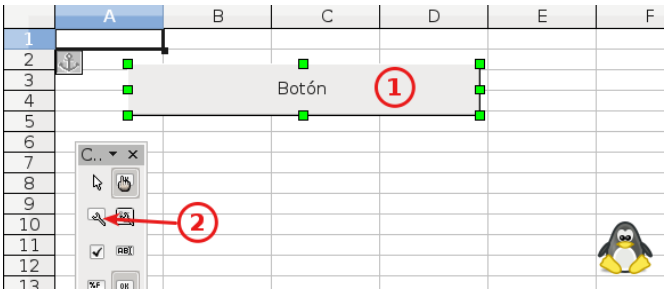
Nota como tenemos una macro para casi todo los eventos, esto realmente no es muy usual, pero no descartes que un día lo necesites, también, considera que si guardas las asignaciones en OpenOffice.org, los eventos responderán a “todos” los documentos de “todas” las aplicaciones presentes en OpenOffice.org.

9.2 Asignando eventos en controles

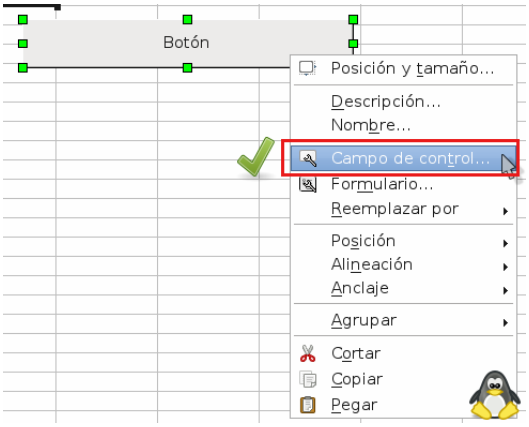
Los formularios y sus controles, los puedes asociar a bases de datos, y automatizar muchas tareas sin una línea de código, pero para ilustrar el uso de eventos y macros asociadas a estos, por ahora, omitiremos esta capacidad de enlazarse con bases de datos y trataremos de asignar código para todas las tareas. Los controles tienen muchos eventos en común, compartidos, la mayoría, con los controles de cuadros de diálogo, veremos los más usuales de ellos, tratando de ir ilustrando su uso en unos y en otros, que reitero, es muy similar.

Lo primero que aprenderemos, es donde asociar los eventos de los controles a nuestras macros, ya que todos se asignan de la misma manera, solo veremos el primero, tanto en los formularios como en los cuadros de diálogo, después, solo indicaremos que macro va en cada evento.

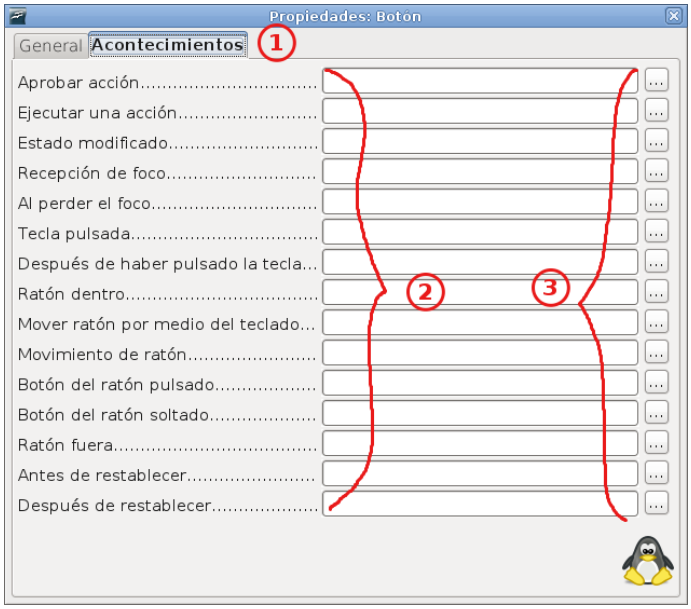
Agrega un control de formulario a tu hoja de calculo, el que quieras. Para nuestro ejemplo, hemos agregado un botón de comando (*CommandButton*) (1), asegurate de que este seleccionado (solo en tiempo de diseño) y da un clic en el icono “Control” (2) de la barra de herramientas “Campos de control de formulario”, como lo ilustramos en la siguiente imagen.



Otra forma de lograr lo mismo, es a través del menú contextual del control.

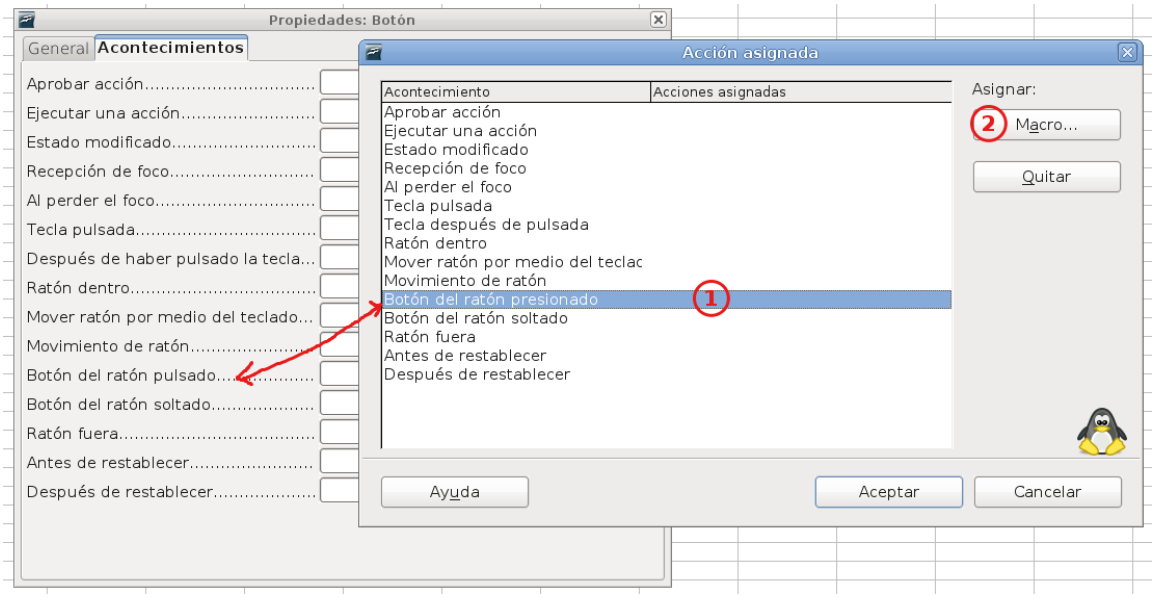


Cualquiera de las dos formas anteriores, te mostrará el cuadro de diálogo “Propiedades” del control, que ya hemos usado, pero ahora, tienes que seleccionar la ficha “Acontecimientos” (1), que te mostrará los eventos (2) que soporta el control seleccionado y el icono de asignación de macro (3) para cada uno.

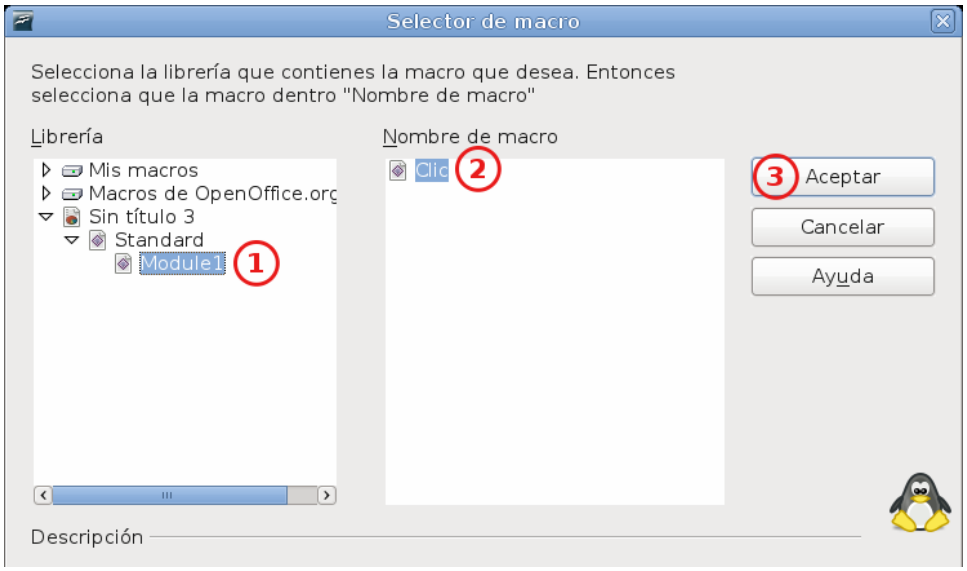


Selecciona el icono de asignación de macro, que te mostrará el siguiente cuadro de diálogo. De forma predeterminada, estará seleccionado el evento que hayas seleccionado en el cuadro de diálogo anterior, pero aquí puedes cambiar a otro (1) si quieres, pero ojo, compara esta lista de eventos con la del cuadro de diálogo anterior, observa que los nombres son “similares”, no iguales, pero están en el mismo orden, trataré de mostrarte los dos nombres cuando haga

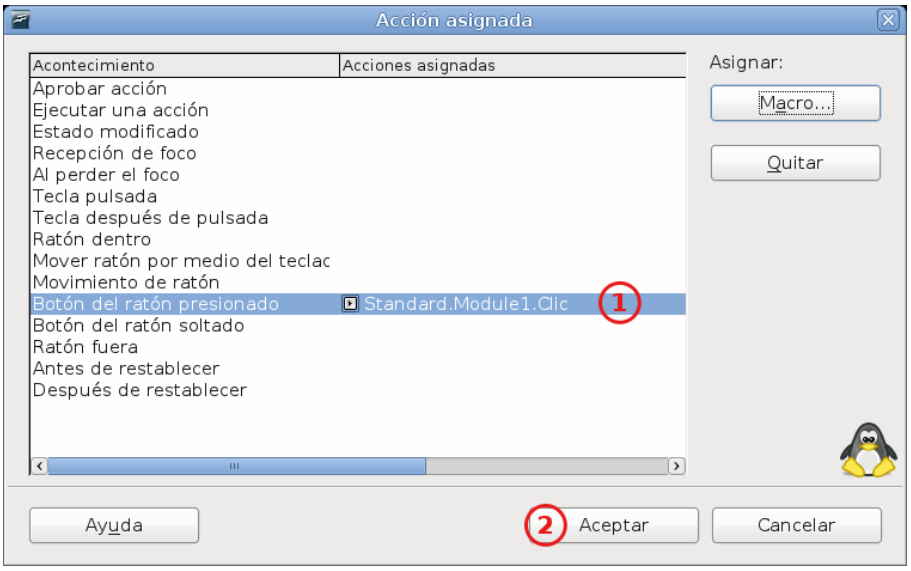
referencia a ellos. Selecciona el evento que quieras y da un clic en el botón de comando "*Macro*" (2) como se ve en la siguiente imagen.



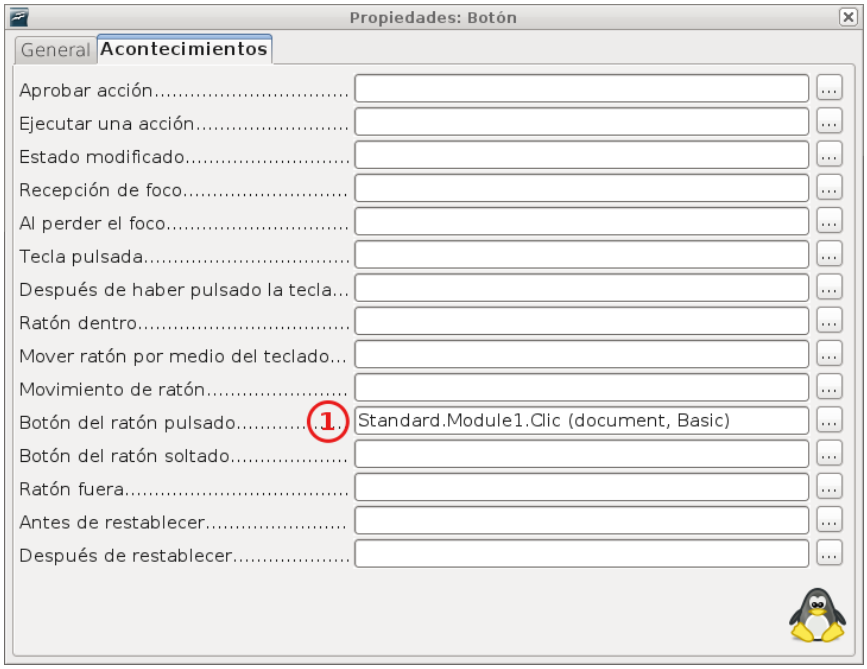
Tienes que ver el cuadro de diálogo "Selector de macros" que ya conoces. Tienes que seleccionar el archivo con el modulo (1), donde este la macro (2) que asignaremos al evento seleccionado, da un clic en el botón de comando "Aceptar" (3) para hacer la asignación.



La acción anterior te regresará al cuadro de diálogo "Asignar acción", pero ahora tienes que ver la asociación respectiva de evento y macro (1). Solo te resta, dar clic en el botón de comando "Aceptar" (2).

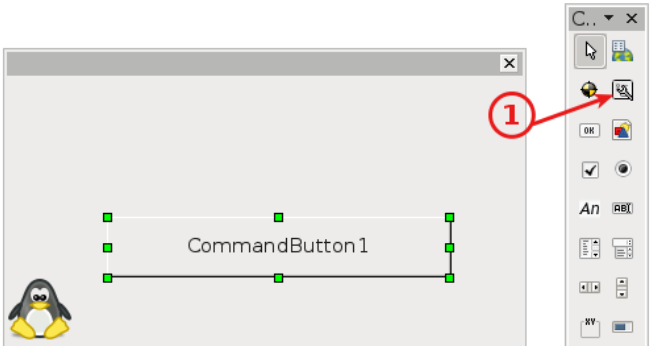


Regresando al nuestro primero cuadro de diálogo, donde de nuevo, tienes que ver la asignación de la macro correcta en el evento correcto (1). Para terminar, solo cierra el cuadro de diálogo.



Recuerda que para que los controles respondan a las macros asignadas a sus eventos, tienes que pasar a modo ejecución, como aprendimos en el capítulo anterior.

Ahora veamos como se asigna macros a los eventos de los controles en cuadros de diálogo. Para ello, agrega un cuadro de diálogo a tu archivo desde el IDE, puedes agregar un control o seleccionar directamente el cuadro de diálogo. Después, selecciona el icono “Propiedades” (1) del cuadro de herramientas.



Si lo prefieres, puedes usar el menú contextual sobre el control seleccionado, donde te mostrará como primera opción ***Propiedades***.

Esta acción te mostrará el cuadro de diálogo "Propiedades", el mismo visto unas líneas atrás, donde también tienes que seleccionar la ficha "Acontecimientos" (eventos). Todos los subsiguientes pasos, son los mismos aprendidos en los formularios.

9.3 Principales eventos en controles

La mayoría de los controles implementan los mismos eventos. Lo importante es que sepas "cuando" sucede un evento, cuando es llamado para interceptarlo y codificar lo necesario para cumplir el propósito de tu aplicación. También, aunque muchos eventos son soportados por un control, no es común escribir código en ellos, por ejemplo, las etiquetas (*Label*) implementan el evento "*Botón del ratón pulsado – Clic de ratón*", pero no es común usarlo, casi siempre, este control es usado como un control estático o solo para mostrar información, pero claro, puedes usarlo, si así lo consideras. Otro ejemplo de este caso, son los controles cuadros de texto (*TextBox*), este control también implementa el evento "*Botón del ratón pulsado – Clic de ratón*", pero no es muy común establecer código en el evento de este control, de nuevo, tu criterio y experiencia decidirá. Vamos a trabajar sobre formularios y cuadros de diálogo, puedes usar el archivo que quieras, pero pon atención en los controles que vamos agregando, pues los vamos a ir usando conforme conozcamos los diferentes eventos.

9.3.1 Evento "Botón del ratón pulsado" – Clic de ratón

Este evento sucede cuando el usuario pulsa un botón del ratón, el botón que sea. Creo que es uno de los eventos más usados. Como curiosidad; ¿te imaginas cuantos clics, se han dado desde que se invento el primero ratón?. Agrega un etiqueta (*Label*) al formulario y asigna la siguiente macro a este evento.

```
Sub Evento_Clic( Evento )
Dim oControl As Object

   'Accedemos al modelo del control
   oControl = Evento.Source.getModel
   'Establecemos la fecha y hora actual, en la propiedad Titulo
   oControl.Label = Now
```

```
End Sub
```

Nota que no necesitamos saber el “nombre” del control, pues accedemos a el, desde el mismo evento (*Evento.Source*). Una posibilidad muy interesante de la asignación de eventos en controles, es que podemos asignar la misma macro, a “todos” los controles que queramos, prueba a asignar esta misma macro a, por ejemplo, un botón de opción (*OptionButton*) y veras que también funciona, esto es posible, por que este control, también cuenta con la propiedad “Titulo” (*Label*), así que esta macro, funcionara con cualquier control que cuente con esta propiedad.

Para nuestro siguiente ejemplo, es necesario que agregues un cuadro de diálogo a tu archivo y un botón de comando (*CommandButton*) al formulario, a este ultimo, asigna la siguiente macro al evento que estamos estudiando.

```
Sub MostrarDialogo()
Dim oDialogo As Object

   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("Dialogo") )
   oDialogo.execute()
   oDialogo.dispose()

End Sub
```

Ahora, agrega una etiqueta (*Label*) al cuadro de dialogo y asigna la misma macro asignada a nuestra etiqueta (*Label*) en el formulario. Pruebala y verifica que funcione como se espera, es decir, que te muestre la fecha y hora actual. Si todo va bien hasta ahora, debes de tener una etiqueta y un botón de comando en el formulario, con el que mostramos el cuadro de diálogo con su respectiva etiqueta.

Al principio, mencionamos que este evento es llamado cuando se presiona un botón del ratón, el que sea, pero podemos saber que botón presiono el usuario de la siguiente manera. Para este ejemplo, necesitamos un nuevo botón de comando (*CommandButton*) y un control de imagen (*ImageControl*) en el formulario, así como tres imágenes que te gusten, localizadas en una ruta accesible. Al botón de comando, asigna la siguiente macro al evento que estamos estudiando.

```
Sub MostrarImagen( Evento )
Dim sRuta(2) As String
dim oFormulario As Object
Dim oImagen As Object

   'Las rutas de ubicación de las imágenes
   sRuta(0) = ConvertToUrl("/home/mau/imagen1.jpg")
   sRuta(1) = ConvertToUrl("/home/mau/imagen2.jpg")
   sRuta(2) = ConvertToUrl("/home/mau/imagen3.jpg")

   'Accedemos al formulario
   oFormulario = Evento.Source.getModel.getParent()
   'El control de imagen
   oImagen = oFormulario.getByName("imgImagen")

   'Evaluamos que botón fue pulsado
   Select Case Evento.Buttons
         Case 1           'Botón izquierdo
               oImagen.ImageURL = sRuta(0)
         Case 2           'Botón derecho
               oImagen.ImageURL = sRuta(1)
         Case 4           'Botón central
               oImagen.ImageURL = sRuta(2)
   End Select
```

```
End Sub
```

Observa como hemos hecho referencia al formulario, a través del control que llama al evento (*Evento.Source.getModel.getParent*), también observa que el nombre de la macro no tiene por que llevar la palabra “evento”, puedes llamarla como quieras, siempre que cumpla los requisitos del lenguaje. La forma de hacer referencia al formulario, es válida, si el control al que queremos hacer referencia esta en el mismo formulario, si no esta en el mismo formulario, tienes que usar la forma larga aprendida en capítulos anteriores.

```
oFormulario = ThisComponent.getCurrentController.getActiveSheet.getDrawPage.getForms.getByName(
"Standard" )
```

Si quieres hacer lo mismo en controles de cuadros de diálogo, la forma de acceder a estos cambia, como se ve en el siguiente código.

```
Option Explicit
Dim oDialogo As Object

Sub MostrarImagen( Evento )
Dim sRuta(2) As String
Dim oImagen As Object

   'Las rutas de ubicación de las imágenes
   sRuta(0) = ConvertToUrl("/home/mau/imagen1.jpg")
   sRuta(1) = ConvertToUrl("/home/mau/imagen2.jpg")
   sRuta(2) = ConvertToUrl("/home/mau/imagen3.jpg")

   'El control de imagen
   oImagen = oDialogo.getControl("imgImagen").getModel()

   'Evaluamos que botón fue pulsado
   Select Case Evento.Buttons
         Case 1           'Botón izquierdo
               oImagen.ImageURL = sRuta(0)
         Case 2           'Botón derecho
               oImagen.ImageURL = sRuta(1)
         Case 4           'Botón central
               oImagen.ImageURL = sRuta(2)
   End Select

End Sub

Sub MostrarDialogo()
   DialogLibraries.LoadLibrary( "Standard" )
   oDialogo = CreateUnoDialog( DialogLibraries.Standard.getByName("Dialogo") )
   oDialogo.execute()
   oDialogo.dispose()
End Sub
```

Observa que la variable para hacer referencia al cuadro de diálogo (*oDialogo*), la hemos quitado de la macro “MostrarDialogo” y la hemos puesto a nivel modulo, esto nos permite hacer referencia a esta variable, desde otras macros, como ya lo hemos aprendido. Cuidado, la macro “MostrarImagen” asignada al formulario y al cuadro de diálogo, ya no son la misma, las he escrito en módulos diferentes, por ello pueden tener el mismo nombre.

9.3.2 Evento “Botón del ratón soltado”

Este evento es casi igual al anterior, excepto por que es llamado, cuando el usuario suelta el botón del ratón. Para este ejemplo usaremos el control de imagen que ya tenemos y dos de las misma imágenes usadas en el ejemplo anterior. Asigna al evento “Botón del ratón pulsado” (*Clic de ratón*) la macro “Imagen_Clic” y al evento “Botón del ratón soltado” (*Al soltar el botón del ratón*) la macro “Imagen_Soltar_Clic”, que te muestro aquí.

```
Sub Imagen_Clic( Evento )
Dim sRuta(0) As String
dim oFormulario As Object
Dim oImagen As Object

   'Las rutas de ubicación de las imágenes
   sRuta(0) = ConvertToUrl("/home/mau/imagen2.jpg")

   'Accedemos al formulario
   oFormulario = Evento.Source.getModel.getParent()
   'El control de imagen
   oImagen = oFormulario.getByName("imgImagen")
   'Cambiamos la imagen
   oImagen.ImageURL = sRuta(0)

End Sub

Sub Imagen_Soltar_Clic( Evento )
Dim sRuta(0) As String
dim oFormulario As Object
Dim oImagen As Object

   'Las rutas de ubicación de las imágenes
   sRuta(0) = ConvertToUrl("/home/mau/imagen1.jpg")

   'Accedemos al formulario
   oFormulario = Evento.Source.getModel.getParent()
   'El control de imagen
   oImagen = oFormulario.getByName("imgImagen")
   'Cambiamos la imagen
   oImagen.ImageURL = sRuta(0)

End Sub
```

Presiona cualquier botón del ratón sobre el control de imagen, la imagen tiene que cambiar, pero ojo, manten presionado el botón unos segundos, después sueltalo, la imagen debe de cambiar. Trata de hacer esto mismo con controles de cuadro de diálogo.

En este evento, también puedes saber que botón presionó el usuario, usando la misma técnica del ejemplo anterior.

OOo Basic no implementa el evento “Doble_Clic”, pero puedes emularlo con una propiedad de este evento, de la siguiente manera.

```
Sub Emular_Doble_Clic( Evento )

   'Evaluamos el número de clics
   If Evento.ClickCount = 2 Then
        MsgBox "Doble clic"
   End If

End Sub
```

Esta técnica la puedes usar en cualquiera de los dos eventos vistos hasta ahora. Es importante, a la hora de asignar una macro a un evento, que consideres las propiedades de este, es decir, si asocias esta macro a otro evento, por ejemplo, alguno del teclado, los cuales no implementa esta propiedad, el código te producirá un error.

El siguiente ejemplo combina el uso del clic y del doble clic, para esto, agrega un nuevo botón de comando (*CommandButton*), dos cuadros de lista (*ListBox*) y usaremos la etiqueta ya agregada. La asociación de eventos y macros en los controles es el siguiente.

Control	Nombre	Eventos	Macros
Botón de comando	cmdDatos	Botón del ratón pulsado	cmdDatos_Clic
Cuadro de lista	lstCuadroLista1	Botón del ratón pulsado Botón del ratón soltado	Lista_Clic Lista_Doble_Clic
Cuadro de lista	lstCuadroLista2	Botón del ratón pulsado Botón del ratón soltado	Lista_Clic Lista_Doble_Clic

Y las macros respectivas.

```
Sub cmdDatos_Clic( Evento )
Dim oFormulario As Object
Dim oLista1 As Object
Dim oLista2 As Object
Dim mDatos()

   oFormulario = Evento.Source.getModel.getParent()
   oLista1 = oFormulario.getByName("lstCuadroLista1")
   oLista2 = oFormulario.getByName("lstCuadroLista2")
   'Vaciamos el segundo cuadro de lista
   oLista2.StringItemList = mDatos()
   mDatos = Array("Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto",
"Septiembre", "Octubre", "Noviembre", "Diciembre")
   'Llenamos con datos el primero
   oLista1.StringItemList = mDatos()

End Sub


Sub Lista_Clic( Evento )
Dim oFormulario As Object
Dim oEtiqueta As Object

   oFormulario = Evento.Source.getModel.getParent()
   oEtiqueta = oFormulario.getByName("lblInfo")
   'Mostramos en la etiqueta el elemento seleccionado
   oEtiqueta.Label = Evento.Source.getSelectedItem

End Sub


Sub Lista_Doble_Clic( Evento )
Dim oFormulario As Object
Dim oDestino As Object
Dim oDestinoVista As Object

   'Si hay dos clic y hay elemento seleccionado
   If Evento.ClickCount = 2 And ( Evento.Source.getSelectedItemPos > -1 ) Then
         oFormulario = Evento.Source.getModel.getParent()
         'El destino será el control que NO este llamando al evento
         If Evento.Source.getModel.Name = "lstCuadroLista1" Then
               oDestino = oFormulario.getByName("lstCuadroLista2")
         Else
```

```
            oDestino = oFormulario.getByName("lstCuadroLista1")
        End If
        'Accedemos a la vista del control destino
        oDestinoVista = ThisComponent.getCurrentController.getControl( oDestino )
        'Agregamos el elemento seleccionado del origen al destino
        oDestinoVista.addItem( Evento.Source.getSelectedItem, oDestinoVista.getItemCount )
        'Quitamos el elemento seleccionado del origen
        Evento.Source.removeItems( Evento.Source.getSelectedItemPos, 1 )
    End If

End Sub
```

Observa que en la macro "Lista_Doble_Clic", tenemos que acceder a la "vista" del control destino a través del controlador (*getCurrentController*) del documento, pero para el control que llamo al evento no es necesario, pues de forma predeterminada, esta en su modo vista. En la primer macro (*cmdDatos_Clic*), solo iniciamos los valores de los cuadros de lista, el primero lo llenamos con los meses del año y el segundo solo lo vaciamos. En la segunda macro (*Lista_Clic*), cada vez que el usuario selecciona un elemento de cualquiera de los dos cuadros de lista, el elemento seleccionado se muestra en la etiqueta (*Label*) "lblInfo", y la tercer macro (*Lista_Doble_Clic*), mueve el valor seleccionado de un cuadro de lista al otro. Tus controles tienes que verse más o menos así.

La propiedad que cuenta los clics (*ClickCount*) del ratón en este evento, puede ir aumentando a mucho más que dos, mientras otro evento no lo detenga o hasta que se te canse el dedo de dar clic. Para lograr lo mismo en un cuadro de diálogo, solo tienes que considerar el modo diferente en que se hace referencia a los controles, que es a través de la variable a nivel de modulo que apunta al cuadro de diálogo, las propiedades y el algoritmo deben de funcionar igual.

9.3.3 Evento "Ratón dentro" – Puntero encima

Este evento se ejecuta, cuando el ratón "entra", se posiciona "dentro" del cualquier área del control que lo llama. Asigna la siguiente macro a este evento de una nueva etiqueta.

```
Sub Raton_Entra( Evento )

    'Cambiamos de color el fondo del control
    Evento.Source.getModel.BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)
    'Quitamos la leyenda del control
    Evento.Source.getModel.Label = ""

End Sub
```

Este evento solo se ejecuta una vez al entrar el ratón en el control.

9.3.4 Evento "Ratón fuera" – Puntero fuera

Este evento es el contrario del anterior, se ejecuta al salir el ratón del control. Asigna la siguiente macro a la misma etiqueta usada en el evento anterior.

```
Sub Raton_Sale( Evento )

    'Cambiamos el color y el texto de la etiqueta
    Evento.Source.getModel.BackgroundColor = RGB(255,255,255)
```

```
    Evento.Source.getModel.Label = "Mueve el ratón sobre mi"

End Sub
```

Este evento también se ejecuta solo una vez.

9.3.5 Evento “Movimiento del ratón”

Este evento es llamado mientras el cursor del ratón este dentro del control y este en movimiento. Puedes usar la misma etiqueta o agregar una nueva para la siguiente macro, que tienes que asignar a este evento. También usaremos la primer etiqueta que agregamos (lblInfo) para mostrar información de este evento.

```
Sub Raton_En_Movimiento( Evento )
Dim oFormulario As Object
Dim oEtiqueta As Object

   'Cambiamos de color el fondo del control
   Evento.Source.getModel.BackgroundColor = RGB(Rnd*255,Rnd*255,Rnd*255)
   oFormulario = Evento.Source.getModel.getParent()
   oEtiqueta = oFormulario.getByName("lblInfo")
   'Mostramos la posición del ratón dentro del control
   oEtiqueta.Label = "X = " & Evento.X & " : " & "Y = " & Evento.Y

End Sub
```

Mira que interesante (y divertido), podemos mostrar la ubicación del cursor del ratón a través de sus coordenadas X-Y, dentro del control que llama a este evento.

El siguiente ejemplo, toma el ancho, el alto y la diagonal del control y de acuerdo a estos valores, establece proporcionalmente el valor para el rojo, el verde y el azul del color de fondo del control.

```
Sub Color_RGB( Evento )
Dim oFormulario As Object
Dim oEtiqueta As Object
Dim Ancho As Integer
Dim Alto As Integer
Dim Rojo As Integer
Dim Verde As Integer
Dim Azul As Integer

   'Obtenemos el ancho y alto del control
   Ancho = Evento.Source.getSize.Width
   Alto = Evento.Source.getSize.Height

   'Obtenemos la proporción de cada color
   Rojo = Evento.X / (Ancho / 255)
   Verde = Evento.Y / (Alto / 255)
   Azul = Sqr( Evento.Y ^ 2 + Evento.X ^ 2 ) / ( Sqr( Alto ^ 2 + Ancho ^ 2 ) / 255)

   'Establecemos los colores
   Evento.Source.getModel.BackgroundColor = RGB( Rojo, Verde, Azul)
   oFormulario = Evento.Source.getModel.getParent()
   oEtiqueta = oFormulario.getByName("lblInfo")

   'Mostramos la información
```

```
    oEtiqueta.Label = "Rojo = " & Rojo & " : Verde = " & Verde & " : Azul = " & Azul

End Sub
```

Con esta macro, logramos cambiar el color de fondo del control, de acuerdo a la posición del cursor del ratón dentro de él. Para el valor del color azul, proporcional a la diagonal del control, viene en nuestra ayuda Don Pitagoras y su famoso teorema.

Prueba a realizar este ejercicio, pero usando botones de selección, también puedes usar barras de desplazamiento o un control numérico para establecer el valor de los colores.

9.3.6 Evento “Mover ratón por medio del teclado” - Movimiento de ratón con tecla pulsada

Este evento es similar al anterior, pero solo es llamado cuando alguna de las siguientes teclas esta presionada: SHIFT (MAYUS), CTRL o ALT. Agrega una nueva etiqueta y asigna la siguiente macro a este evento.

```
Sub Mover_Con_TeclaPulsada( Evento )
Dim oFormulario As Object
Dim oEtiqueta As Object
Dim Ancho As Integer
Dim Color As Long
Dim Rojo As Integer
Dim Verde As Integer
Dim Azul As Integer

   'Obtenemos el ancho y alto del control
   Ancho = Evento.Source.getSize.Width

   'Obtenemos la proporción actual de cada color
   Color = Evento.Source.getModel.BackgroundColor
   Rojo = Int( Color / 65536 ) And 255
   Verde = Int( Color / 256 ) And 255
   Azul = Color And 255

   'Establecemos los colores, solo modificamos el color de acuerdo a la tecla especial seleccionada
   Select Case Evento.Modifiers
         Case 1
               Rojo = Evento.X / (Ancho / 255)
               Evento.Source.getModel.BackgroundColor = RGB( Rojo, Verde, Azul)
         Case 2
               Verde = Evento.X / (Ancho / 255)
               Evento.Source.getModel.BackgroundColor = RGB( Rojo, Verde, Azul)
         Case 4
               Azul = Evento.X / (Ancho / 255)
               Evento.Source.getModel.BackgroundColor = RGB( Rojo, Verde, Azul)
   End Select
   oFormulario = Evento.Source.getModel.getParent()
   oEtiqueta = oFormulario.getByName("lblInfo")
   oEtiqueta.Label = "Rojo = " & Rojo & " : Verde = " & Verde & " : Azul = " & Azul

End Sub
```

Tomamos el ancho del control como unidad de medida, que va desde 0 en el extremo izquierdo a 255 al extremo derecho. Tomamos el valor actual de cada color y solo cambiamos el color de acuerdo a la tecla pulsada, SHITF para el rojo, CTRL para el verde y ALT

para el azul. Los valores de las teclas (*Modifiers*) están determinados por las siguientes constantes.

com.sun.star.awt.KeyModifier	*Valor*	*Valor en Interfaz*
com.sun.star.awt.KeyModifier.SHIFT	1	Tecla SHIFT
com.sun.star.awt.KeyModifier.MOD1	2	Tecla CTRL
com.sun.star.awt.KeyModifier.MOD2	4	Tecla ALT

Puedes sumar estos valores para saber si hay más de una tecla pulsada.

9.3.7 Evento “Recepción de foco” - Al activar área

Este evento es llamado cuando el control recibe el “foco”, es decir, cuando el cursor pasa a el por medio de teclado o ratón o incluso por medio de código. Para nuestro ejemplo agrega un nuevo control de cuadro de texto (*TextBox*) y establece un texto predeterminado en el, después, asigna la siguiente macro a este evento.

```
Sub PonerColor( Evento )

   'Cambiamos el color de fondo y de fuente
   With Evento.Source.getModel
         .BackgroundColor = RGB( 255, 255, 153)
         .TextColor = RGB( 0, 0, 200)
   End With

End Sub
```

Cuando el cursor entre en este control, el color de fondo y fuente cambiara, esto es útil para indicarle visualmente al usuario en que control esta el foco, por supuesto, hay que cambiar estos colores cuando el foco cambie de control, como lo demostramos en el siguiente evento.

9.3.8 Evento “Al perder el foco” - Al desactivar área

Este evento es el inverso del anterior, es llamado cuando el cursor sale del control. Al mismo control cuadro de texto (*TextBox)* agregado en nuestro evento anterior, asigna la siguiente macro.

```
Sub QuitarColor( Evento )

   'Cambiamos el color de fondo y de fuente
   With Evento.Source.getModel
         .BackgroundColor = RGB( 255, 255, 255)
         .TextColor = RGB( 0, 0, 0)
   End With

End Sub
```

Cuando el foco sale del control, establecemos los colores predeterminados, blanco para el fondo del control y negro para la fuente, estas macros las puedes asignar a los eventos de cualquier control que pueda recibir el foco y que implemente estas propiedades. Agrega unos cuantos cuadro de texto (*TextBox*) y asigna las mismas macros para que veas su uso.

Ten cuidado cuando manejes eventos de foco, la primer precaución es usar las propiedades correctas cuando estés codificando, un error te puede provocar un ciclo infinito, por que el IDE trata de mostrarte el mensaje de error, pero al mostrarlo, el control sigue respondiendo a estos eventos lo que provoca que no puedas corregirlo sin forzar la salida de OpenOffice.org. Un error similar lo puede provocar una lógica errónea cuando mueves el foco por código.

El control que tiene el foco, resalta de los demás. Te aseguro que más de un usuario te lo agradecerá. Algunos programadores usan este evento para hacer la validación del contenido, bien estructurado, no deberías tener problemas, por ejemplo.

```
    'Evaluamos si es un número
    If Not IsNumeric( Evento.Source.getModel.Text ) Then
          MsgBox "El contenido no es un número"
          'Si no es, regresamos el foco al control
          Evento.Source.setFocus()
    End If
```

Solo debes de tener precaución de no cruzar eventos que provoquen ciclos infinitos, esto suele pasar mucho cuando uno es novel y se agregan demasiados controladores de eventos a un mismo control sin una técnica depurada.

Otros programadores prefieren hacer una sola validación global de todos los datos introducidos por el usuario, no importa que técnica uses, lo que si es muy importante, es que nunca, reitero, nunca dejes de validar los datos que introduce el usuario y notificarle de un modo suficientemente visible que es lo que esta mal y por que, he visto, no pocos “grandes” proyectos, obviar tan elemental tarea.

9.3.9 Evento “Tecla pulsada”

Este evento es llamado cuando el usuario pulsa una tecla del teclado. A un control cuadro de texto (*TextBox*), asigna la siguiente macro a este evento.

```
Sub Tecla_Pulsada( Evento )
Dim oFormulario As Object
Dim oEtiqueta As Object

   oFormulario = Evento.Source.getModel.getParent()
   oEtiqueta = oFormulario.getByName("lblInfo")
   'Mostramos el ultimo caracter introducido y su código
   oEtiqueta.Label = Evento.KeyChar & " = " & Evento.KeyCode

End Sub
```

Si algún caracter, requiere una combinación de teclas, este evento solo es llamado una vez, por ejemplo, para las letras acentuadas, si mantienes presionada una tecla, este evento será llamado una y otra vez hasta que sueltes la tecla y de acuerdo a la configuración de velocidad de repetición de tu teclado. La primero propiedad (KeyChar) te muestra el caracter tal cual se ve en la interfaz del usuario, si vez solo un símbolo como este: □ es que se presionó una

tecla que no tiene representación visual, la segunda propiedad (KeyCode), es el código de la tecla pulsada, de acuerdo a las siguientes constantes.

com.sun.star.awt.Key	*Valor*	*Tecla*
com.sun.star.awt.Key.NUM0	256	0
com.sun.star.awt.Key.NUM1	257	1
com.sun.star.awt.Key.NUM2	258	2
com.sun.star.awt.Key.NUM3	259	3
com.sun.star.awt.Key.NUM4	260	4
com.sun.star.awt.Key.NUM5	261	5
com.sun.star.awt.Key.NUM6	262	6
com.sun.star.awt.Key.NUM7	263	7
com.sun.star.awt.Key.NUM8	264	8
com.sun.star.awt.Key.NUM9	265	9
com.sun.star.awt.Key.A	512	A
com.sun.star.awt.Key.B	513	B
com.sun.star.awt.Key.C	514	C
com.sun.star.awt.Key.D	515	D
com.sun.star.awt.Key.E	516	E
com.sun.star.awt.Key.F	517	F
com.sun.star.awt.Key.G	518	G
com.sun.star.awt.Key.H	519	H
com.sun.star.awt.Key.I	520	I
com.sun.star.awt.Key.J	521	J
com.sun.star.awt.Key.K	522	K
com.sun.star.awt.Key.L	523	L
com.sun.star.awt.Key.M	524	M
com.sun.star.awt.Key.N	525	N
com.sun.star.awt.Key.O	526	O
com.sun.star.awt.Key.P	527	P
com.sun.star.awt.Key.Q	528	Q
com.sun.star.awt.Key.R	529	R
com.sun.star.awt.Key.S	530	S
com.sun.star.awt.Key.T	531	T
com.sun.star.awt.Key.U	532	U
com.sun.star.awt.Key.V	533	V
com.sun.star.awt.Key.W	534	W
com.sun.star.awt.Key.X	535	X
com.sun.star.awt.Key.Y	536	Y
com.sun.star.awt.Key.Z	537	Z
com.sun.star.awt.Key.F1	768	F1
com.sun.star.awt.Key.F2	769	F2
com.sun.star.awt.Key.F3	770	F3

com.sun.star.awt.Key.F4	771	F4
com.sun.star.awt.Key.F5	772	F5
com.sun.star.awt.Key.F6	773	F6
com.sun.star.awt.Key.F7	774	F7
com.sun.star.awt.Key.F8	775	F8
com.sun.star.awt.Key.F9	776	F9
com.sun.star.awt.Key.F10	777	F10
com.sun.star.awt.Key.F11	778	F11
com.sun.star.awt.Key.F12	779	F12
com.sun.star.awt.Key.F13	780	
com.sun.star.awt.Key.F14	781	
com.sun.star.awt.Key.F15	782	
com.sun.star.awt.Key.F16	783	
com.sun.star.awt.Key.F17	784	
com.sun.star.awt.Key.F18	785	
com.sun.star.awt.Key.F19	786	
com.sun.star.awt.Key.F20	787	
com.sun.star.awt.Key.F21	788	
com.sun.star.awt.Key.F22	789	
com.sun.star.awt.Key.F23	790	
com.sun.star.awt.Key.F24	791	
com.sun.star.awt.Key.F25	792	
com.sun.star.awt.Key.F26	793	
com.sun.star.awt.Key.DOWN	1024	Flecha abajo
com.sun.star.awt.Key.UP	1025	Flecha arriba
com.sun.star.awt.Key.LEFT	1026	Flecha izquierda
com.sun.star.awt.Key.RIGHT	1027	Flecha derecha
com.sun.star.awt.Key.HOME	1028	Inicio
com.sun.star.awt.Key.END	1029	Fin
com.sun.star.awt.Key.PAGEUP	1030	RePag
com.sun.star.awt.Key.PAGEDOWN	1031	AvPag
com.sun.star.awt.Key.RETURN	1280	Enter
com.sun.star.awt.Key.ESCAPE	1281	Esc
com.sun.star.awt.Key.TAB	1282	Tabulador
com.sun.star.awt.Key.BACKSPACE	1283	Retroceso
com.sun.star.awt.Key.SPACE	1284	Barra espaciadora
com.sun.star.awt.Key.INSERT	1285	Insert
com.sun.star.awt.Key.DELETE	1286	Suprimir
com.sun.star.awt.Key.ADD	1287	+ (Suma)
com.sun.star.awt.Key.SUBTRACT	1288	- (Resta)
com.sun.star.awt.Key.MULTIPLY	1289	* (Multiplicación)
com.sun.star.awt.Key.DIVIDE	1290	/ (División)

com.sun.star.awt.Key.POINT	1291	. (Punto)
com.sun.star.awt.Key.COMMA	1292	, (Coma)
com.sun.star.awt.Key.LESS	1293	< (Menor que)
com.sun.star.awt.Key.GREATER	1294	> (Mayor que)
com.sun.star.awt.Key.EQUAL	1295	= (Igual)
com.sun.star.awt.Key.OPEN	1296	
com.sun.star.awt.Key.CUT	1297	
com.sun.star.awt.Key.COPY	1298	
com.sun.star.awt.Key.PASTE	1299	
com.sun.star.awt.Key.UNDO	1300	
com.sun.star.awt.Key.REPEAT	1301	
com.sun.star.awt.Key.FIND	1302	
com.sun.star.awt.Key.PROPERTIES	1303	
com.sun.star.awt.Key.FRONT	1304	
com.sun.star.awt.Key.CONTEXTMENU	1305	
com.sun.star.awt.Key.HELP	1306	
com.sun.star.awt.Key.MENU	1307	
com.sun.star.awt.Key.HANGUL_HANJA	1308	
com.sun.star.awt.Key.DECIMAL	1309	. (Punto decimal)
com.sun.star.awt.Key.TILDE	1310	
com.sun.star.awt.Key.QUOTELEFT	1311	
com.sun.star.awt.Key.DELETE_TO_BEGIN_OF_LINE	1536	
com.sun.star.awt.Key.DELETE_TO_END_OF_LINE	1537	
com.sun.star.awt.Key.DELETE_TO_BEGIN_OF_PARAGRAPH	1538	
com.sun.star.awt.Key.DELETE_TO_END_OF_PARAGRAPH	1539	
com.sun.star.awt.Key.DELETE_WORD_BACKWARD	1540	
com.sun.star.awt.Key.DELETE_WORD_FORWARD	1541	
com.sun.star.awt.Key.INSERT_LINEBREAK	1542	
com.sun.star.awt.Key.INSERT_PARAGRAPH	1543	
com.sun.star.awt.Key.MOVE_WORD_BACKWARD	1544	
com.sun.star.awt.Key.MOVE_WORD_FORWARD	1545	
com.sun.star.awt.Key.MOVE_TO_BEGIN_OF_LINE	1546	
com.sun.star.awt.Key.MOVE_TO_END_OF_LINE	1547	
com.sun.star.awt.Key.MOVE_TO_BEGIN_OF_PARAGRAPH	1548	
com.sun.star.awt.Key.MOVE_TO_END_OF_PARAGRAPH	1549	
com.sun.star.awt.Key.SELECT_BACKWARD	1550	
com.sun.star.awt.Key.SELECT_FORWARD	1551	
com.sun.star.awt.Key.SELECT_WORD_BACKWARD	1552	
com.sun.star.awt.Key.SELECT_WORD_FORWARD	1553	
com.sun.star.awt.Key.SELECT_WORD	1554	
com.sun.star.awt.Key.SELECT_LINE	1555	
com.sun.star.awt.Key.SELECT_PARAGRAPH	1556	

com.sun.star.awt.Key.SELECT_ALL	1557	
com.sun.star.awt.Key.SELECT_TO_BEGIN_OF_LINE	1558	
com.sun.star.awt.Key.SELECT_TO_END_OF_LINE	1559	
com.sun.star.awt.Key.MOVE_TO_BEGIN_OF_DOCUMENT	1560	
com.sun.star.awt.Key.MOVE_TO_END_OF_DOCUMENT	1561	
com.sun.star.awt.Key.SELECT_TO_BEGIN_OF_DOCUMENT	1562	
com.sun.star.awt.Key.SELECT_TO_END_OF_DOCUMENT	1563	
com.sun.star.awt.Key.SELECT_TO_BEGIN_OF_PARAGRAPH	1564	
com.sun.star.awt.Key.SELECT_TO_END_OF_PARAGRAPH	1565	

Una tecla tendrá el mismo código (*KeyCode*), no importa si esta en minúsculas, mayúsculas o acentuadas pues el código se refiere a la tecla física del teclado, pero en la propiedad *KeyChar*, tendrás el caracter tal cual en pantalla. Si la tecla no es reconocida, *KeyCode* siempre será cero. Si presionas una tecla o una combinación de teclas que use OpenOffice.org, generalmente este responderá a ella, por ejemplo F1, que te mostrará la ayuda. Puedes saber si el usuario presiono una combinación de teclas, consultando la propiedad *Modifiers,* como lo demostramos en el siguiente evento en donde se comparten las mismas propiedades.

9.3.10 Evento “Después de haber pulsado la tecla” - Tecla soltada

Este evento es llamado cuando el usuario suelta la tecla, pero cuidado, podríamos pensar que este evento se ejecuta solo una vez al soltar la tecla, pero no es así, si mantienes presionada la tecla, este evento será llamado, también, una y otra vez, esto es, por que OOo Basic, reconoce cada caracter que se introduce con la repetición como si fuera una pulsación de tecla independiente. Por ello, cuando mantienes presionada una tecla, estos eventos son llamados alternativamente, consideralo al escribir tu código. Asigna la siguiente macro a este evento de un nuevo cuadro de texto.

```
Sub Tecla_Soltada( Evento )
Dim oFormulario As Object
Dim oEtiqueta As Object
Dim sInfo As String

   oFormulario = Evento.Source.getModel.getParent()
   oEtiqueta = oFormulario.getByName("lblInfo")

   Select Case Evento.KeyCode
         'Teclas sin representación visual
         Case 768 : sinfo = "F1"
         Case 769 : sinfo = "F2"
         Case 770 : sinfo = "F3"
         Case 771 : sinfo = "F4"
         Case 772 : sinfo = "F5"
         Case 773 : sinfo = "F6"
         Case 774 : sinfo = "F7"
         Case 775 : sinfo = "F8"
         Case 776 : sinfo = "F9"
         Case 777 : sinfo = "F10"
         Case 778 : sinfo = "F11"
         Case 779 : sinfo = "F12"
         Case 1024 : sinfo = "Flecha abajo"
         Case 1025 : sinfo = "Flecha arriba"
         Case 1026 : sinfo = "Flecha izquierda"
```

```
        Case 1027 : sinfo = "Flecha derecha"
        Case 1028 : sinfo = "Inicio"
        Case 1029 : sinfo = "Fin"
        Case 1030 : sinfo = "RePag"
        Case 1031 : sinfo = "AvPag"
        Case 1280 : sinfo = "Enter"
        Case 1281 : sinfo = "Esc"
        Case 1282 : sinfo = "Tab"
        Case 1283 : sinfo = "Retroceso"
        Case 1284 : sinfo = "Espacio"
        Case 1285 : sinfo = "Insertar"
        Case 1286 : sinfo = "Suprimir"
        'Todas las demás
        Case Else : sInfo = Evento.KeyChar
    End Select

    'Si presiono alguna tecla especial
    Select Case Evento.Modifiers
        Case 1 : sInfo = "Shift + " & sInfo
        Case 2 : sInfo = "Ctrl + " & sInfo
        Case 4 : sInfo = "Alt + " & sInfo
    End Select

    sInfo = sInfo & " : " & Evento.KeyCode
    oEtiqueta.Label = sInfo

End Sub
```

Recuerda que las teclas especiales (*Modifiers*), son susceptibles de sumarse, es decir, el usuario puede pulsar más de una y esta propiedad nos lo informará, modifica la macro anterior para considerar esto.

9.3.11 Otros eventos

Todos los eventos vistos hasta ahora, son compartidos por todos los controles, pero hay algunos que solo están disponibles para algunos controles.

El evento **“Al ejecutar” - Al iniciar**, responde de manera similar al evento “Botón de ratón pulsado”, pero tiene la particularidad de que no cuenta con propiedades para saber si se presionó alguna tecla (*Modifiers*) o botón del ratón (*Buttons*), por lo que solo es llamado con el botón primario del ratón. Si no usaras estos argumentos, este evento puede ser una buena opción para los botones de comando.

El evento **“Modificado”**, es llamado cuando el contenido de un control cambia, dependiendo del control se puede desencadenar de diferentes formas, por ejemplo, si es un cuadro de lista, con solo cambiar de selección entre su contenido, este evento es llamado, pero si es un cuadro de texto, este evento solo es llamado al perder el foco el control y solo si el contenido de este cambio.

El evento **“Estado modificado” - Estado de elemento modificado**, es llamado cuando en un control cuadro de lista, cambia el elemento seleccionado.

El evento **“Texto modificado”**, responde a cualquier control que tenga área de edición de texto, con cualquier cambio en su contenido y cada vez que se cambie, este evento es llamado.

Hay que tener cuidado con el uso de estos tres últimos eventos. Un error frecuente es modificar el contenido del control en estos eventos, con lo que corremos el riesgo de caer en un ciclo infinito, usalos con precaución y moderación.

10 Un proyecto paso a paso

Ya estamos en nuestro último capitulo, en este, pondremos en practica la mayoría de los conocimientos adquiridos. El proyecto que vamos a desarrollar, es un sistema básico pero funcional de facturación que se llamará, por supuesto; “Factura Libre”.

Dentro del desarrollo de software, existe un tema primario e importante que se llama “Análisis y diseño de sistemas”, que son las herramientas y conocimientos necesarios para desarrollar, lo más “óptimamente” posible, un programa. Aun y cuando seas un programador novel o autodidacta (como yo), no lo tomes como un tema menor, en tu buscador favorito encontraras amplia documentación al respecto, como este tema sale fuera del ámbito (y alcance) de estas notas, para los fines de nuestro proyecto, usaremos mucho más, al que dicen llamar, el menos común de los sentidos, el “sentido común”.

Tratemos de responder las tres preguntas siguientes: ¿qué?, ¿para qué? y ¿para quién?. ¿Qué?, un sistema de facturación. ¿Para qué?, para llevar el control completo del proceso de facturación. ¿Para quién?, usuario básico de computadora. El “qué” y “para que” pueden convertirse sin problema en el “objetivo” general de tu sistema, que no debes de perder de vista nunca, a lo largo del desarrollo de tu programa. Aun y cuando tu vayas a ser el usuario del sistema, es importante que tengas presente y consideres que un programa lo usa, siempre, un usuario final, reitero, aun y cuando tú seas la misma persona, imagínate que no es así. Parte de la idea de facilitarle la vida (informática) al usuario (no de complicársela), en la medida de lo posible, el programador tiene que “cubrir las necesidades” del usuario no al revés, no lo olvides, he visto no pocos sistemas, de muchos tamaños, donde el usuario tiene que cambiar sus procesos de trabajo, por responsabilidad (por decirlo de forma suave) de los programadores. Tu estas al servicio del usuario, no el al tuyo, reitero, no lo olvides.

Ya tenemos el objetivo de nuestro sistema, ahora, tratemos de ver el ¿como?. Para responder esta pregunta, es relevante saber con que recursos contamos. Primero los más importantes, los recursos humanos. En el desarrollo de un sistema, dependiendo de su tamaño y complejidad, pueden participar muchas personas; analistas de sistemas, diseñadores de interfaces, normalizadores de bases de datos, programadores, documentadores, implantadores, entre otros muchos especialistas. Para nuestro sistema, trataremos de asumir algunos de estos roles, por ahora, estamos como analistas del sistema y hemos determinado que en recursos humanos contamos con dos personas, tu y yo. Un recurso humano de suma importancia para el éxito de un sistema, es el usuario final, su conocimiento de **la necesidad a cubrir por el sistema**, puede ser crucial para su buen termino, puede llegar a ser tu recurso más valioso.

Ahora, pensemos en los recursos materiales a nuestra disposición, principalmente, el hardware donde se implementará el sistema, no pienses en el hardware donde se desarrollará, si no donde se utilizará, puede que tu máquina sea moderna y rápida, la del usuario tal vez no. En los recursos materiales, considera, también, el sistema o sistemas operativos donde se usará el sistema así como las versiones de estos, para nuestro caso, puede ser importante que consideres la versión de OpenOffice.org con que cuenta el usuario final.

El siguiente punto a analizar, serían los objetivos particulares del sistema, es decir, los alcances de este, que hará y que no, por ejemplo:

- Control de clientes
- Control de facturas
- Control de pagos

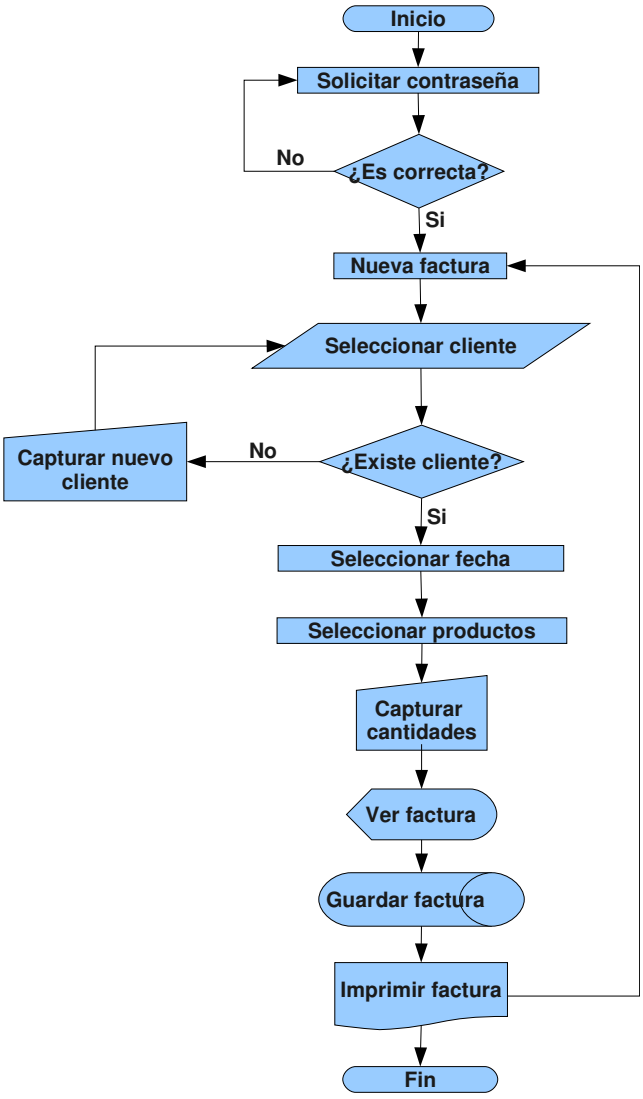
Tienes que establecer la prioridad de estos alcances, en primer instancia, aquellos sin los cuales, el sistema deja de cumplir su objetivo general, por ejemplo, clientes y facturas, el control de usuarios, de pagos e inventario, pueden ser primarios o secundarios, según, las necesidades de tu usuario, el análisis del sistema y por supuesto, el presupuesto del cliente. Toma

en cuenta que los recursos humanos y materiales pueden determinar algunos alcances del sistema, pero a su vez, los alcances pueden influir en las necesidades “mínimas” de estos recursos, es decir, tienes que tener una visión global de todos estos aspectos, pues unos infieren en los otros.

Nuestra siguiente tarea, es hacer un seudo-código en lenguaje natural, por ejemplo:

- Abrir programa.
- Solicitar contraseña.
- Verificar contraseña.
- Si es correcta dar acceso.
- Si no es correcta solicitarla otra vez.
- Crear una nueva factura.
- Seleccionar cliente.
- Si no existe, dar de alta al cliente.
- Seleccionar fecha.
- Seleccionar productos a facturar.
- Establecer cantidades a facturar.
- Guardar factura.
- Imprimir factura.
- Reiniciar el proceso.
- Salir del programa.

Estas instrucciones pueden ser tan genéricas o detalladas como lo consideres, su utilidad es que te van dando la pauta para el “flujo” de los datos, su relación entre ellos y la jerarquía de los mismos. En siguiente paso es diseñar un diagrama de flujo que refleje lo más certero el seudo-código del paso anterior. Un diagrama de flujo es la representación gráfica de un proceso o un conjunto de procesos, para la solución de un problema, puede ser tan sencillo como el que te muestro de ejemplo, o tan complejo para representar el algoritmo completo de un sistema. Generalmente, hay un conjunto mínimo de símbolos estándar usados para representar cualquier diagrama de flujo, en tu buscador favorito encontraras suficiente documentación al respecto.



El siguiente paso es determinar que datos son los que guardará el programa, como primera aproximación tenemos la siguiente lista genérica.

- Los datos de los clientes
- Los datos de las facturas
- El detalle de las facturas

Ahora, hagamosla más detallada de modo que sean “casi” definitivos los datos a guardar. En este punto es recomendable una conversación clara y extensa con el usuario final, de hecho, es recomendable una excelente comunicación con el usuario final a lo largo de todo el desarrollo de tu sistema, veras que será mucho mejor para los dos.

¿Que datos de los clientes y de las facturas guardaremos?, esto dependerá del alcance de tu sistema, previo acuerdo con el usuario y de las necesidades contables, que sabemos, varían de región en región. Nos avocaremos a las solicitadas en México, aunque espero sean lo suficientemente ilustrativas para sea posible adaptarlo a cualquier otra latitud o a tus necesidades particulares. Para nuestro caso, los datos mínimos que requerimos guardar para poder elaborar una factura valida son:

- El nombre del cliente, también llamado Razón Social.
- Su dirección fiscal completa, es decir, la declarada frente a la autoridad.

- Su RFC, que en México es una secuencia especial alfanumérica que identifica de forma univoca a cada contribuyente.
- La fecha y lugar de elaboración de la factura.
- Detalle de los conceptos facturados, cantidad, descripción, importe, subtotal.
- Los impuestos respectivos de acuerdo a su estatus fiscal y área de servicios.

Ahora, desglosaremos cada uno de estos puntos en unidades de información más pequeñas, lo más pequeñas posibles:

- Nombre (Razón Social del cliente)
- Calle
- Número
- Colonia
- Delegación o Municipio
- CP (Código Postal)
- Ciudad o Estado
- RFC
- Número de factura
- Cliente al que se le factura
- Fecha de la factura
- Cantidad
- Descripción
- PU (Precio Unitario)
- Importe
- Subtotal
- Impuestos
- Total

Observa que algunos de estos datos (Importe, subtotal, impuestos, total) son susceptibles de calcularse, máxime si estamos en una hoja de calculo, en un sistema pequeñito como el que estamos enfrentando, no veras mucha diferencia de desempeño entre guardar todos los datos ya calculados o calcularlos cada vez que son mostrados al usuario, en sistemas más grandes puede ser crucial esta diferencia. Por ahora podemos darnos el lujo de usar uno u otro método, pero no olvides esta importante diferencia.

Ahora, tratemos de agrupar los datos en grupos lógicos, es decir, los que tengan relación **directa**, unos con otros, primero los datos de los clientes:

Clientes
Nombre
Calle
Número
Colonia
Delegación o Municipio
Código Postal
Ciudad o Estado
RFC

Ahora, los datos de las facturas:

Facturas
Número
Cliente

Facturas
Fecha
Subtotal
Impuesto
Total
Estado

Observa que hemos agregado un dato, Estado, este nos servirá como mínimo, para saber si una factura ya fue pagada o no, aunque puede tener muchos otros estatus, tantos como tu criterio te dicte.

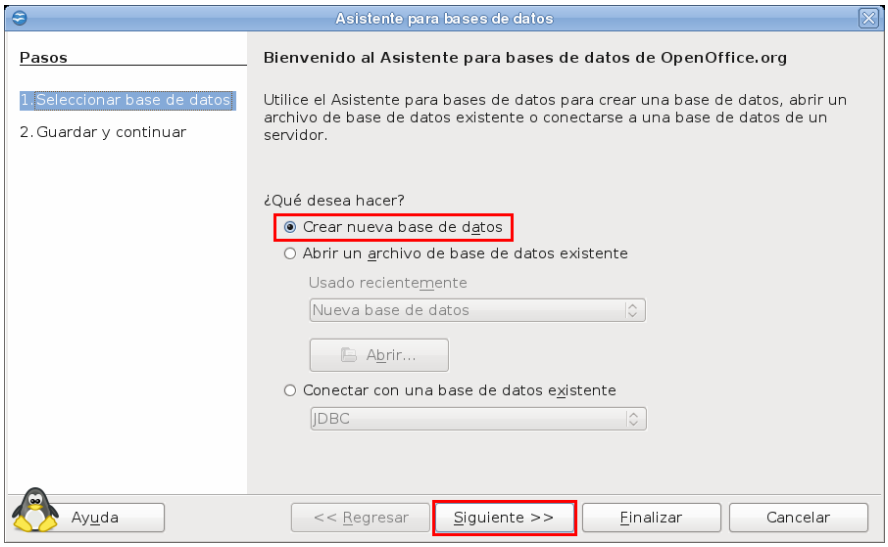
Ahora, el detalle de la factura:

Detalle Factura
Número de Factura
Cantidad
Descripción
Precio
Importe

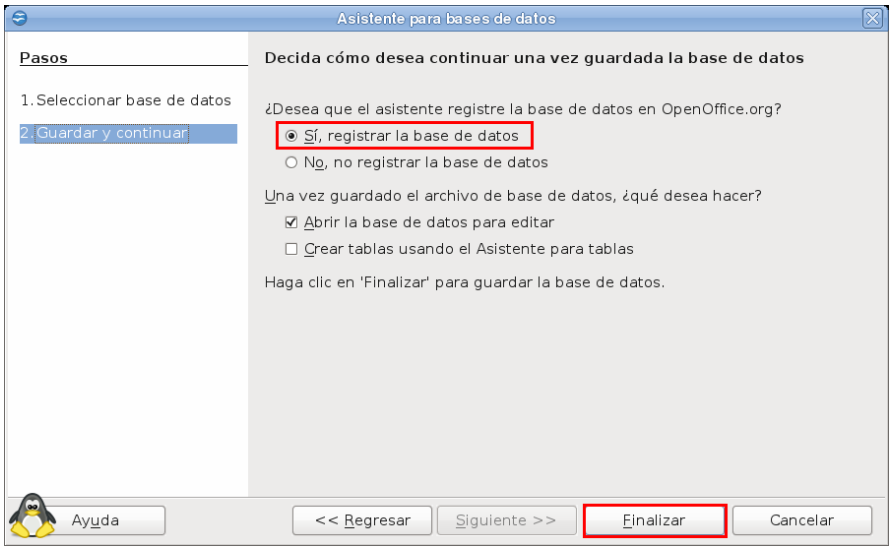
A cada grupo puedes agregarlos los datos que tu necesites, por ejemplo; a los clientes puedes agregarle datos como teléfonos o personas de contacto, a las facturas algunos clientes te piden que agregues números de pedido, remisión o algún otro dato, en el detalle tal vez necesites un dato para la unidad. Quita o agrega según te dicte tu experiencia y necesidad.

A cada uno de estos grupos le llamaremos "**tabla**", al conjunto de todas nuestras tablas le llamaremos "**base de datos**". A cada dato de cada tabla le llamaremos "**campo**" y será el encabezado de cada columna, cada fila de datos que guardemos con todos sus campos, le llamaremos "**registro**".

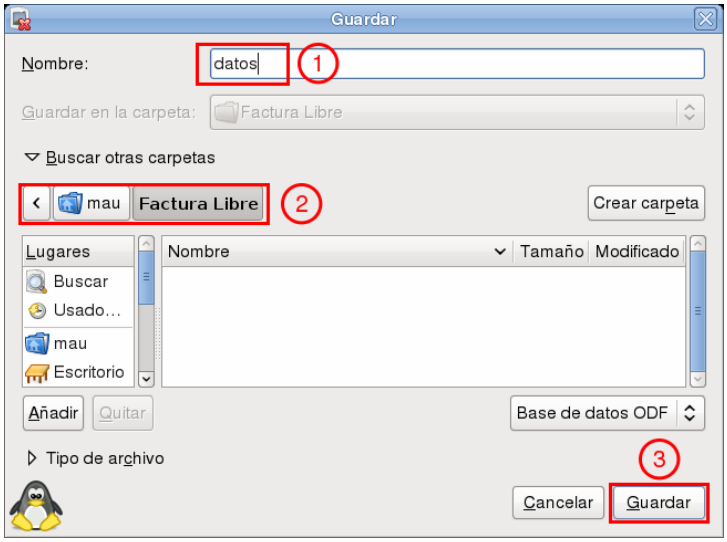
Abre Base para que te muestre el Asistente para base de datos donde seleccionaremos **Crear nueva base de datos** para dar clic en el botón de comando **Siguiente**.



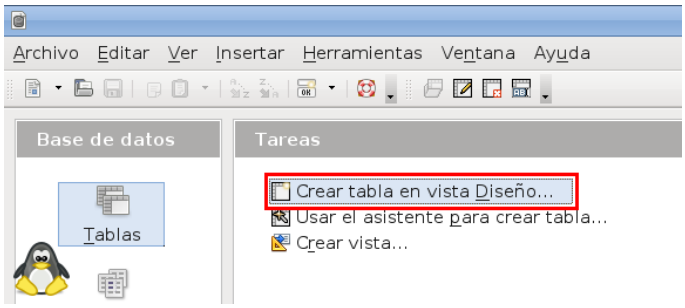
El siguiente paso le indicaremos que si queremos registrar la base de datos para después dar clic en el botón de comando Finalizar.



En el ultimo paso, selecciona el nombres de tu nueva base de datos (1), asegurate de ir guardando todo en un solo directorio (2), para finalizar da clic en **Guardar** (3).



Crearemos una nueva tabla en vista diseño.

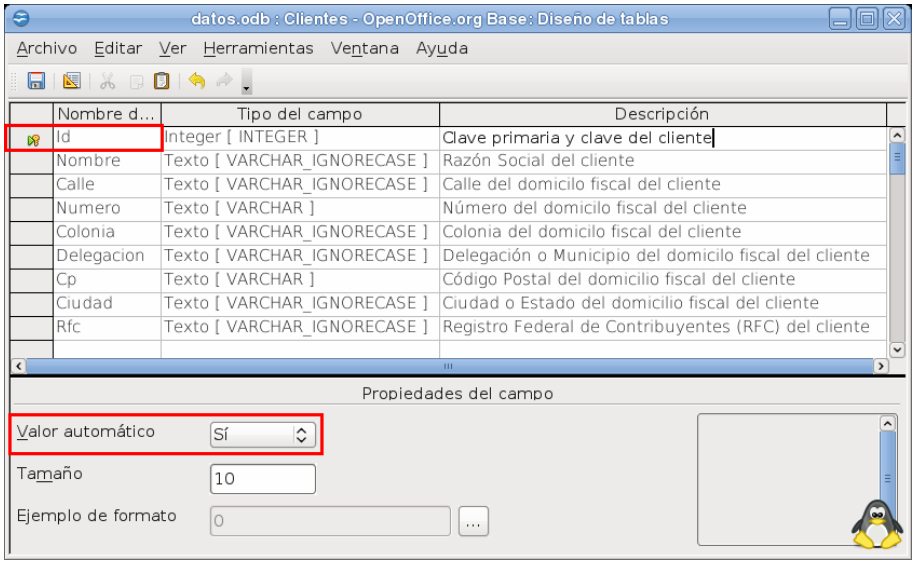


Agrega los siguiente campos con las siguientes características para la tabla Clientes.

Campo	*Tipo*	*Tamaño*
Id	Entero (INTEGER)	

Campo	Tipo	Tamaño
Nombre	Texto (VARCHAR_IGNORECASE)	100
Calle	Texto (VARCHAR_IGNORECASE)	100
Numero	Texto (VARCHAR)	50
Colonia	Texto (VARCHAR_IGNORECASE)	50
Delegacion	Texto (VARCHAR_IGNORECASE)	50
Cp	Texto (VARCHAR)	5
Ciudad	Texto (VARCHAR_IGNORECASE)	50
Rfc	Texto (VARCHAR_IGNORECASE)	13

Nota que a esta tabla le hemos agregado un campo llamado “**Id**”, puedes nombrarlo también “Clave” o algún otro que consideres. La utilidad de este campo es identificar de forma unívoca a cada registro de nuestra tabla, en bases de datos a este campo suele llamarsele “**clave primaria**”, que en las tablas de Base distingues por el icono de la llave a la izquierda del nombre del campo, esta propiedad se establece de forma automática en cada campo que establezcas como entero (Integer) y establezcas que sea Valor automático. Procura agregar al menos un campo que sea clave primaria en todas tus tablas.



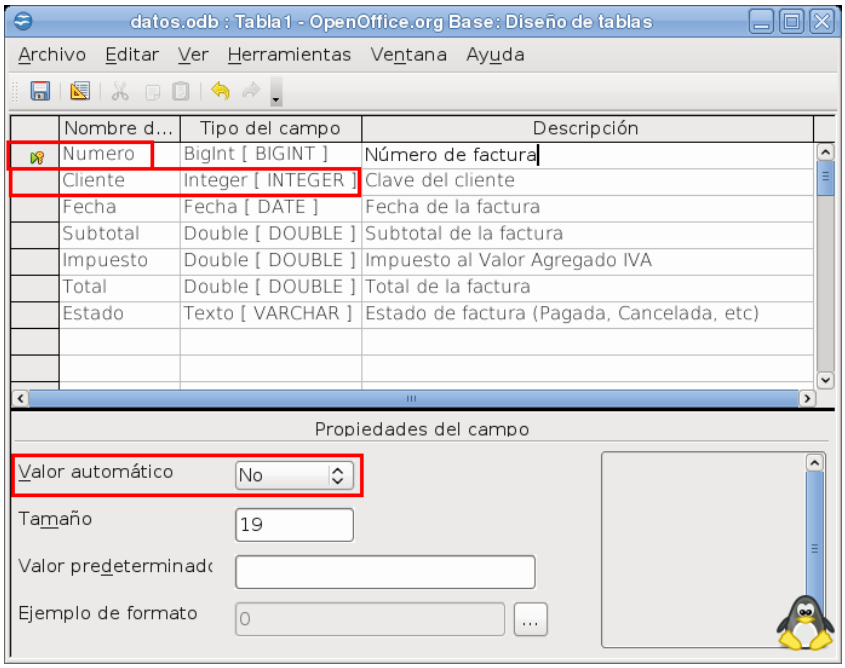
Como siguiente paso guardar la nueva tabla con el nombre “Clientes”.

En el caso de la tabla “Facturas”, el número de factura es único, por lo que este puede fungir como clave primaria, hay casos (en México) en los que las facturas llevan una letra adicional cuando el emisor cuenta con sucursales, en este caso tienes que ingeniártelas para solventar esto. Los campos para esta tabla son:

Campo	Tipo	Tamaño
Numero	Entero grande (BIGINT)	
Cliente	Entero (INTEGER)	

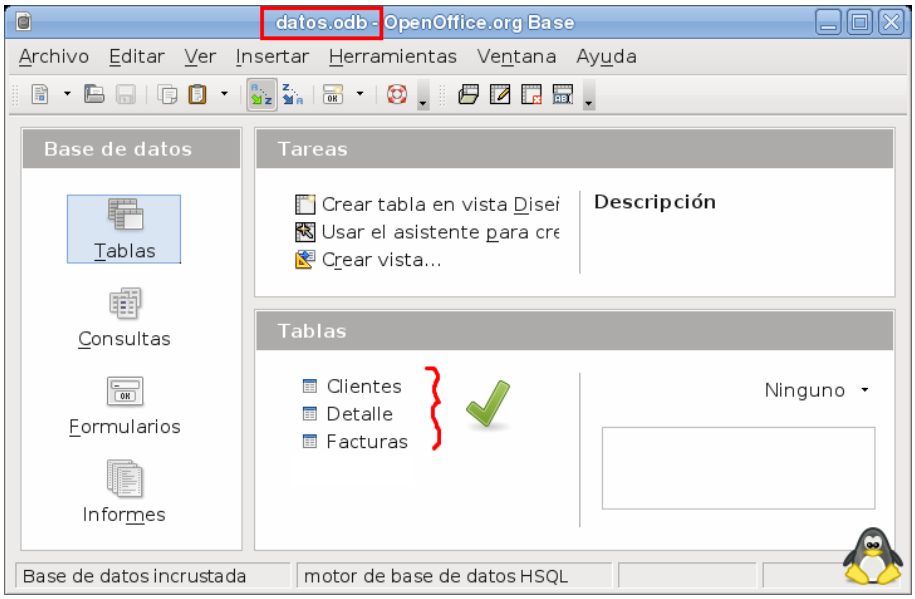
Campo	Tipo	Tamaño
Fecha	Fecha (DATE)	
Subtotal	Doble (DOUBLE)	
Impuesto	Doble (DOUBLE)	
Total	Doble (DOUBLE)	
Estado	Texto (VARCHAR)	20

Es importa que no establezcas el campo **Numero** como valor automático porqué, salvo en contadas ocasiones, este número no empezará en cero, también observa que el campo **Cliente**, lo hemos establecido como entero (integer), esto es importante pues con este campo relacionaremos esta tabla con la tabla “Clientes”. Si tu régimen fiscal te obliga a aplicar más u otros impuestos, establecelos en esta tabla. Así mismo, si tu cliente te pide agregar algún dato como “Pedido”, también hazlo en esta tabla.



Guarda la tabla con el nombre “Facturas”.

Te queda de tarea agregar los campos para la tercer tabla llamada “Detalles”, cuidado con esta tabla, puede tener muchas variantes, usa la más sencilla.



Ahora crea y guarda un nuevo archivo de Calc que nos servirá como interfaz para el usuario, en el, agregaremos los controles necesarios para manipular los datos. La separación de los datos y la interfaz, es una buena practica de programación. En este punto tenemos muchas posibilidades, usaremos una combinación de formularios y cuadros de diálogo para resolverlo. Si bien los formularios tienen incorporadas propiedades simples y útiles para enlazarse y manipular bases de datos, para fines didácticos, procuráremos hacer uso de cuadros de diálogo que nos obligarán a hacer uso de más código pero en compensación repasaremos muchos de los conceptos tratados en este libro. La mayor parte del código esta suficientemente comentado, nos detendremos solo en aquellos que requieran algún comentario en especial. Cada cuadro de diálogo tendrá asociado un modulo de código, además incluye uno que se llame Utilidades para agregar las macros y funciones genéricas que nos usaremos en otros modulos.

Lo primero que tenemos que garantizar, es que los datos existan, para ello, al abrir nuestro archivo verificamos que estén registrados con la siguiente macro que tienes que asociar al evento ***Abrir documento*** de este archivo.

```
Option Explicit

Sub Inicio()
Dim oDatos As Object

   'Verificamos si los datos existen
   oDatos = ExistenDatos()
   If IsNull( oDatos ) Then
         MsgBox "No se encontró la base de datos, el programa se cerrara", 16,"Error grave"
         'Cerramos este archivo
         ThisComponent.Close( True )
   End If

End Sub
```

La función ***ExistenDatos*** tiene el siguiente código.

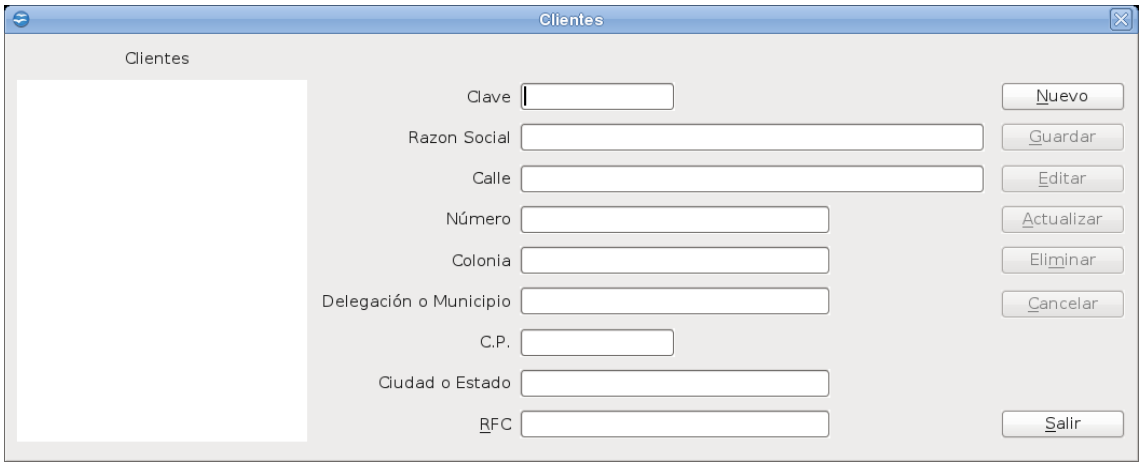
```
'Función para verificar que existan los datos, si existen devolvemos la referencia
Function ExistenDatos() As Object
Dim oDBC As Object
Dim oBD As Object
```

```
    'El servicio de bases de datos
    oDBC = createUnoService("com.sun.star.sdb.DatabaseContext")
    'Nos aseguramos de que esten registrados
    If oDBC.hasByName( DATOS ) Then
          'Accedemos a ellos
          oBD = oDBC.getByName( DATOS )
          'Los regresamos
          ExistenDatos = oBD
    End If

End Function
```

Observa que en esta función estamos usando una constante; **DATOS**, te sugiero agregar un nuevo modulo para ir declarando, tanto las variables como las constantes globales que vayamos necesitando. El establecer una constante para el nombre registrado de la base de datos, nos permite cambiar en un solo lugar este nombre y usarlo a todo lo largo de nuestro programa.

```
Option Explicit

'Cambia aquí el nombre de tu conexión
Global Const DATOS As String = "datos"
```

El siguiente paso es crear el cuadro de diálogo para manipular los datos de la tabla ***Clientes***, agrega un nuevo cuadro de diálogo, agrega un control cuadro de lista (ListBox), controles cuadro de texto (TextBox), uno por cada campo de nuestra tabla, botones de comando (CommandButton) y etiquetas (Label) de modo que te quede lo más parecido a la siguiente imagen.



Todos los cuadros de texto (TextBox) tiene su propiedad **Solo lectura = Si**, los botones de comando (CommandButton), están desactivados, excepto Nuevo y Salir, el botón de comando Salir tiene la propiedad **Tipo de botón = Cancelar**.

Renombra la primer hoja del archivo a **Inicio**, agrega un botón de comando y asociale la siguiente macro a su evento ***Ejecutar una acción***.

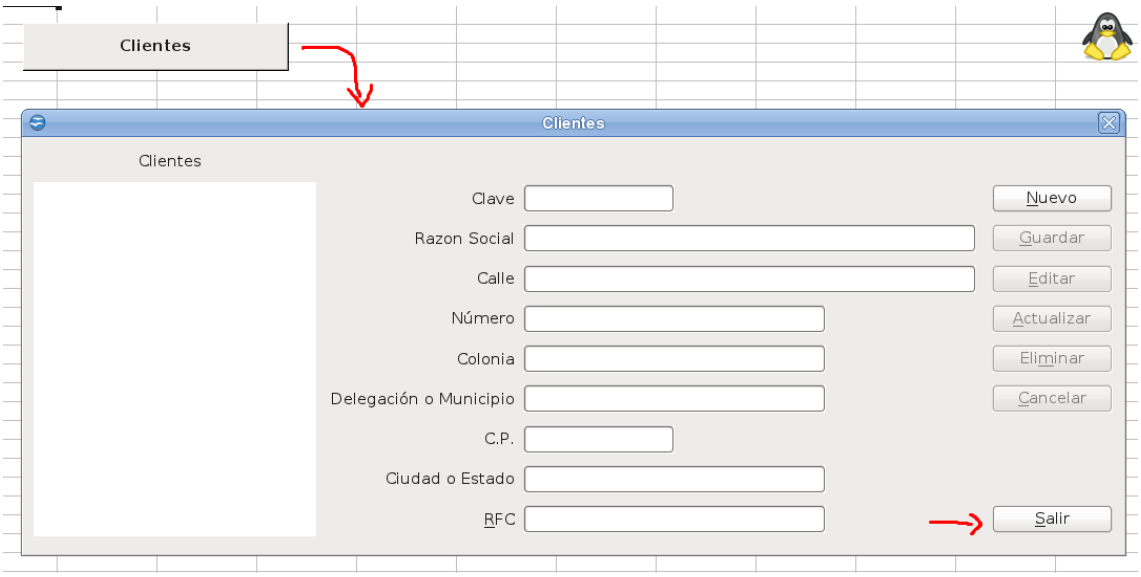
```
Option Explicit

Sub Clientes()
Dim oDlg As Object

   'Cargamos el cuadro de diálogo
   oDlg = CargarDialogo( "Standard","dlgClientes" )
```

```
    'Nos aseguramos de que sea válido
    If Not IsNull(oDlg) Then
          'Lo mostramos
          oDlg.execute()
          'Lo liberamos
          oDlg.dispose()
    Else
          MsgBox "No se pudo mostrar el cuadro de diálogo de Clientes"
    End If

End Sub
```

El código de la función ***CargarDialogo*** es el siguiente.

```
'Función para cargar un cuadro de dialogo en memoria y regresar el cuadro de dialogo
Function CargarDialogo(Libreria As String, Nombre As String) As Object
Dim oLibreria as Object

   'Verificamos que la libreria exista
   If DialogLibraries.hasByName( Libreria ) Then
         'Cargamos la libreria en memoria
         DialogLibraries.LoadLibrary( Libreria )
         'Referencia a la libreria
         oLibreria = DialogLibraries.getByName( Libreria )
         'Verificamos que exista el cuadro de diálogo
         If oLibreria.hasByName( Nombre ) Then
               'Creamos el cuadro de diálogo
               CargarDialogo = CreateUnoDialog( oLibreria.getByName( Nombre ) )
         End If
   End If

End Function
```

En este punto, desde la interfaz de usuario que estamos construyendo, ya deberías poder mostrar y cerrar el cuadro de diálogo.



El botón de comando ***Salir***, debe funcionar aun sin código pues establecimos su propiedad ***Tipo de botón*** en ***Cancelar***, con lo cual podemos cerrar el cuadro de diálogo.

Si bien verificamos que existieran los datos al abrir el documento, no es mala idea volver a verificarlo antes de cargar el cuadro de diálogo. Usaremos una variable a nivel de modulo

para guardar la conexión a la base de datos, también, cada control que se vaya a manipular, debe de referenciarse en una variable, por ello agregaremos las variables necesarias en el mismo nivel del modulo, de modo que sean visibles para todos los eventos y todas las macros que desarrollemos en este modulo.

```
Option Explicit
Dim oDatos As Object
Dim lstClientes As Object
Dim txtClave As Object
Dim txtNombre As Object
Dim txtCalle As Object
Dim txtNumero As Object
Dim txtColonia As Object
Dim txtDelegacion As Object
Dim txtCp As Object
Dim txtCiudad As Object
Dim txtRfc As Object
Dim cmdNuevo As Object
Dim cmdGuardar As Object
Dim cmdEditar As Object
Dim cmdActualizar As Object
Dim cmdEliminar As Object
Dim cmdCancelar As Object
Dim cmdSalir As Object
```

Todos los controles los cargaremos en una sola macro que llamaremos ***CargarControles*** y cuyo código es el siguiente.

```
Sub CargarControles( Dialogo As Object)
   With Dialogo
         lstClientes = .getControl("lstClientes")
         txtClave = .getControl("txtClave")
         txtNombre = .getControl("txtNombre")
         txtCalle = .getControl("txtCalle")
         txtNumero = .getControl("txtNumero")
         txtColonia = .getControl("txtColonia")
         txtDelegacion = .getControl("txtDelegacion")
         txtCp = .getControl("txtCp")
         txtCiudad = .getControl("txtCiudad")
         txtRfc = .getControl("txtRfc")
         cmdNuevo = .getControl( "cmdNuevo" )
         cmdGuardar = .getControl( "cmdGuardar" )
         cmdEditar = .getControl( "cmdEditar" )
         cmdActualizar = .getControl( "cmdActualizar" )
         cmdEliminar = .getControl( "cmdEliminar" )
         cmdCancelar = .getControl( "cmdCancelar" )
         cmdSalir = .getControl( "cmdSalir" )
   End With
End Sub
```

La macro ***Clientes***, ya con la verificación de los datos y la carga de controles, queda de la siguiente manera.

```
Sub Clientes()
Dim oDlg As Object

   'Cargamos el cuadro de diálogo
   oDlg = CargarDialogo( "Standard","dlgClientes" )
   'Nos aseguramos de que sea válido
   If Not IsNull(oDlg) Then
         'Nos aseguramos de que existan los datos
         oDatos = ExistenDatos()
         If Not ISNull( oDatos ) Then
```

```
            'Cargamos los controles en memoria
            Call CargarControles( oDlg )
            'Lo mostramos
            oDlg.execute()
            'Lo liberamos
            oDlg.dispose()
        Else
            MsgBox "No se encontró la base de datos", 16,"Error grave"
        End If
    Else
        MsgBox "No se pudo mostrar el cuadro de diálogo de Clientes"
    End If

End Sub
```

De aquí en adelante, cuando modifiquemos una macro, solo te indicare las líneas necesarias donde hacer el cambio, de este modo no repetiremos tanto código.

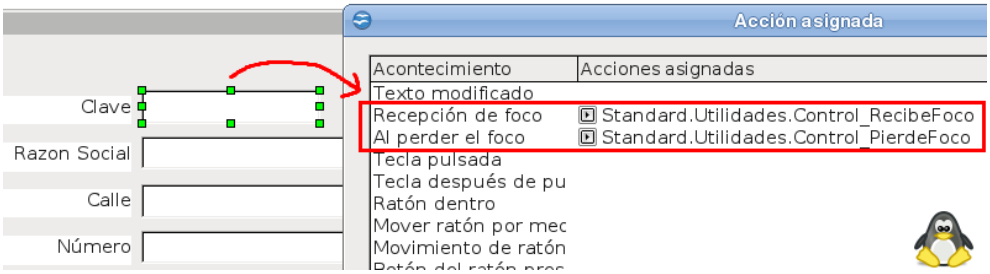
Lo siguiente que haremos no es indispensable, pero a parte de que se ve bonito, nos muestra la versatilidad de OOo Basic, haremos que el color de fondo de los cuadros de texto (TextBox), cambie al color que quieras cuando recibe el foco y regrese a blanco cuando sale de el, de este modo, le indicamos visualmente al usuario en que campo esta actualmente, las macros que logran esto son las siguientes.

```
'Cambia el color de fondo al recibir el foco
Sub Control_RecibeFoco(Evento As Object)
   Evento.Source.Model.Border = 0
   Call CambiaColorFF( Evento.Source, AMARILLO_PASTEL, 0 )
End Sub

'Cambia el color de fondo al perder el foco
Sub Control_PierdeFoco(Evento As Object)
   Evento.Source.Model.Border = 1
   Call CambiaColorFF( Evento.Source, BLANCO, 0 )
End Sub

'Procedimiento que cambia el color de fondo y fuente de un control
Sub CambiaColorFF(Control As Object, Fondo As Long, Fuente As Long)
   Control.Model.BackgroundColor = Fondo
   Control.Model.TextColor = Fuente
End Sub
```

Observa que en la macro ***CambiaColorFF*** podemos cambiar tanto el color de fondo como la fuente, de este modo puedes jugar con la combinación de colores que más te guste, yo siempre dejo la fuente en negro porque tengo poca imaginación para eso de los colores. Observa que al recibir el foco uso la constante ***AMARILLO_PASTEL*** y al perder el foco uso ***BLANCO***, estas constantes están declaradas a nivel global y tienen los siguiente valores.

```
Global Const AMARILLO_PASTEL As Long = 16777164
Global Const BLANCO As Long = 16777215
```

Ahora, para verlas trabajando, solo te resta asociar las macros respectivas en los eventos correspondientes de los cuadros de texto.



Asocia las mismas macros a **todos** los cuadros de texto (TextBox), si, las mismas a todos los controles, de hecho, mientras un control, no importa cual, soporte estas propiedades, puedes asociarle estas macros, esto es una característica muy poderosa de OOo Basic.

Mencionamos que los cuadros de texto (*TextBox*) son de solo lectura, esto es para que controlemos la adición y edición de los datos, también, los botones de comando (*CommandButton*) están desactivados de forma predeterminada y solo activamos los necesarios en el momento indicado, por ejemplo, si la tabla de clientes esta vacía, solo debe estar activado el botón de ***Nuevo***, pero si ya hay datos, podemos activar además: ***Editar*** y ***Eliminar***. De modo que los botones de comando se activarán y desactivaran de acuerdo a la acción que seleccione el usuario, la siguiente tabla nos ayudará a controlar el estado de cada botón.

	Nuevo	Guardar	Editar	Actualizar	Eliminar	Cancelar
Nuevo		✓		✓	✓	✓
Guardar	✓					
Editar		✓		✓	✓?	✓?
Actualizar			✓			
Eliminar		✓		✓	✓?	✓?
Cancelar	✓		✓			
Salir		✓		✓	✓	✓

El signo de interrogación nos indica que estos botones pueden o no pueden estar activados, ¿de que depende? Dependerá de la existencia o no de más registros.

Cuando abrimos nuestro cuadro de diálogo, tenemos que traer los clientes existentes, si los hay y mostrarlos en el cuadro de lista (*ListBox*), para ello, agregamos la siguiente línea a la macro ***Clientes***, justo después de cargar los controles.

```
'Cargamos los controles en memoria
Call CargarControles( oDlg )
'Traemos los nombres de los clientes si lo hay
Call TraerClientes( oDatos )
```

La macro ***TraerClientes*** tiene el siguiente código.

```
Sub TraerClientes( oDatos )
Dim oConsulta As Object
Dim sSQL As String
Dim mClientes()
Dim lTotalRegistros As Long

   'Obtenemos el total de clientes
   lTotalRegistros = TotalRegistros( oDatos, "Clientes", "Id" )
   'Si no hay registros
   If lTotalRegistros = 0 Then
         MsgBox "El catalogo de Clientes esta vacio, usa el botón Nuevo para agregar uno",
64,"Clientes"
   Else
         'Si hay registros construimos la consulta para regresar los clientes ordenados por Nombre
         sSQL = "SELECT Nombre FROM Clientes ORDER BY Nombre"
         'Hacemos la consulta
         oConsulta = HacerConsulta( oDatos, sSQL )
         'Movemos los datos a una matriz
         mClientes = CampoAMatriz( oConsulta, lTotalRegistros, 1 )
         'Y los agregamos al cuadro de lista
         Call MatrizACuadroLista( mClientes(), lstClientes )
         'Como si hay datos, activamos Editar y Eliminar
         cmdEditar.setEnable( True )
         cmdEliminar.setEnable( True )
         'Seleccionamos el primer cliente
         lstClientes.selectItemPos( 0, True )
   End If

End Sub
```

Tenemos varias funciones subrutinas nuevas; TotalRegistros, HacerConsulta, CampoAMatriz y MatrizACuadroLista, cuyo código es el siguiente.

```
'Función para enviar una consulta SQL a una base de datos
Function HacerConsulta( BaseDatos As Object, SQL As String) As Object
Dim oConexion As Object
Dim oDeclaracion As Object
Dim oResultado As Object

   'Creamos una conexion a la base de datos
   oConexion = BaseDatos.getConnection( "","" )
   'Creamos un objeto para las instrucciones SQL
   oDeclaracion = oConexion.createStatement()
   'Ejecutamos la consulta
   oResultado = oDeclaracion.executeQuery( SQL )
   'Si hay resultados
   If Not IsNull( oResultado ) Then
         oResultado.next()
         HacerConsulta = oResultado
   End If

End Function


'Función que devuelve el total de registros de un determinado campo y tabla
Function TotalRegistros( BaseDatos As Object, Tabla As String, Campo As String) As Long
Dim sSQL As String
Dim oConsulta As Object

   'Construimos la consulta
   sSQL = "SELECT COUNT( " & Campo & " ) FROM " & Tabla
   'Hacemos la consulta
```

```
   oConsulta = HacerConsulta( BaseDatos, sSQL )
   'Obtenemos y regresamos el valor
   TotalRegistros = oConsulta.getLong(1)

End Function


'Función para mover los registros que regresa una consulta a una matriz
Function CampoAMatriz( Consulta As Object, NumReg As Long, Tipo As Integer )
Dim mDatos()
Dim co1 As Long

   'Pasandole el número de registros hacemos un ciclo
   'determinado en vez de uno indeterminado
   NumReg = NumReg - 1
   'Redimencionamos la matriz
   Redim mDatos( NumReg )
   For co1 = 0 To NumReg
         'El tipo de campo es importante para usar el método correcto
         Select Case Tipo
               Case 1 : mDatos( co1 ) = Consulta.getString(1)
               Case 2 : mDatos( co1 ) = Consulta.getInt(1)
               'Aquí iremos agregando los tipos necesarios
         End Select
         'Nos movemos al siguiente registro
         Consulta.next()
   Next co1
   CampoAMatriz = mDatos()

End Function


'Macro para agregar una matriz a un cuadro de lista
'el método addItem es muy lento, por ello es mejor
'agregar la matriz completa
Sub MatrizACuadroLista( mDatos(), CuadroLista As Object )

   CuadroLista.addItems( mDatos(), 0 )

End Sub
```

Empecemos con el código de nuestro botones, por supuesto el primero sera ***Nuevo***, el cual es muy simple pues solo tiene que preparar el cuadro de diálogo para recibir los datos del usuario, vamos a activar y vaciar los cuadros de texto y a establecer el estado de los demás botones de acuerdo a la tabla vista anteriormente.

```
Sub Nuevo_Clic()

   'Activamos los cuadros de texto
   Call ActivarCuadrosTexto( True )
   'Los vaciamos
   Call LimpiarCuadrosTexto( True )
   'Botones activados
   cmdGuardar.setEnable( True )
   cmdCancelar.setEnable( True )
   'Botones desactivados
   cmdNuevo.setEnable( False )
   cmdEditar.setEnable( False )
   cmdActualizar.setEnable( False )
   cmdEliminar.setEnable( False )
   cmdSalir.setEnable( False )
   'Desactivamos el cuadro de lista
   lstClientes.setEnable( False )
   'Enviamos el cursor al primer campo
```

```
   txtNombre.setFocus()

End Sub
```

Observa que también desactivamos el cuadro de lista con los nombres de los clientes, esto es importante porque a este control le agregaremos su evento Clic, el cual nos traerá los datos del cliente seleccionado, lo desactivamos para que el usuario no se le ocurra dar clic en el mientras esta agregando uno nuevo o mientras edita otro, ya ves como son los usuarios. El código de las nuevas subrutinas es el siguiente.

```
Sub ActivarCuadrosTexto( Activar As Boolean )
   'Recuerda que la clave la establecimos autonúmerico por ello este cuadro nunca lo activamos
   txtNombre.setEditable( Activar )
   txtCalle.setEditable( Activar )
   txtNumero.setEditable( Activar )
   txtColonia.setEditable( Activar )
   txtDelegacion.setEditable( Activar )
   txtCp.setEditable( Activar )
   txtCiudad.setEditable( Activar )
   txtRfc.setEditable( Activar )
End Sub


Sub LimpiarCuadrosTexto( Nuevo As Boolean )
   If Nuevo Then
         txtClave.setText( "<Nuevo>" )
   Else
         txtClave.setText( "" )
   End If
   txtNombre.setText( "" )
   txtCalle.setText( "" )
   txtNumero.setText( "" )
   txtColonia.setText( "" )
   txtDelegacion.setText( "" )
   txtCp.setText( "" )
   txtCiudad.setText( "" )
   txtRfc.setText( "" )
End Sub
```

El código del botón ***Cancelar*** es trivial.

```
Sub cmdCancelar_Clic()
Dim iClientes As Integer

   'Verificamos cuantos clientes hay
   iClientes = lstClientes.getItemCount()
   If iClientes = 0 Then
         'Si no hay clientes limpiamos todo
         Call LimpiarCuadrosTexto( False )
   Else
         'Si hay clientes activamos los controles necesarios
         cmdEditar.setEnable( True )
         cmdEliminar.setEnable( True )
         lstClientes.setEnable( True )
         'Mostramos los datos del cliente seleccionado actualmente
         Call lstClientes_Clic()
   End If
   'Activamos y desactivamos los demás controles necesarios
   cmdNuevo.setEnable( True )
   cmdSalir.setEnable( True )
   cmdGuardar.setEnable( False )
   cmdActualizar.setEnable( False )
```

```
   cmdCancelar.setEnable( False )
   Call ActivarCuadrosTexto( False )

End Sub
```

Observa que si hay datos, llamamos al evento Clic del control cuadro de lista que aun no codificamos porque primero haremos el código del botón ***Guardar***, el cual es el siguiente.

```
Sub cmdGuardar_Clic()
Dim mDatos()
Dim co1 As Integer
Dim sTmp As String
Dim sSQL As String

   'Obtenemos los datos de todos los cuadros de texto
   mDatos = Array( txtNombre.getText, txtCalle.getText, txtNumero.getText, txtColonia.getText,
txtDelegacion.getText, txtCp.getText, txtCiudad.getText, txtRfc.getText, )
   'Agregamos las comillas simples
   For co1 = LBound(mDatos) To UBound(mDatos)
         mDatos(co1) = "'" & mDatos(co1) & "'"
   Next co1
   'Juntamos y separamos por coma
   sTmp = Join( mDatos(), ",")
   'Construimos la instrucción SQL de inserción
   sSQL = "INSERT INTO ""Clientes""
(""Nombre"",""Calle"",""Numero"",""Colonia"",""Delegacion"",""Cp"",""Ciudad"",""Rfc"") VALUES (" &
sTmp & ")"
   'Insertamos los datos
   Call ActualizarDatos( oDatos, sSQL )

   'Activamos y desactivamos los demás controles necesarios
   cmdEditar.setEnable( True )
   cmdEliminar.setEnable( True )
   lstClientes.setEnable( True )
   cmdNuevo.setEnable( True )
   cmdSalir.setEnable( True )
   cmdGuardar.setEnable( False )
   cmdCancelar.setEnable( False )
   'Desactivamos los cuadros de texto
   Call ActivarCuadrosTexto( False )

End Sub
```

¿Que nos falto?, algo en lo que he hecho mucho énfasis a todo lo largo de estas notas, claro, no hemos validado ningún dato, lo cual es un **pésimo**, si no es que el peor hábito de programación. La instrucción SQL **debe estar perfectamente construida**, si no es así, el método ***executeUpdate*** usado en la macro ***ActualizarDatos*** fallará.

```
'Ejecuta la consulta de actualización pasada
Sub ActualizarDatos( BaseDatos As Object, SQL As String)
Dim oConexion As Object
Dim oDeclaracion As Object
Dim oResultado As Object

   'Creamos una conexion a la base de datos
   oConexion = BaseDatos.getConnection( "","" )
   'Creamos un objeto para las instrucciones SQL
   oDeclaracion = oConexion.createStatement()
   'Ejecutamos la inserción de datos
   oDeclaracion.executeUpdate( SQL )

End Sub
```

Modifiquemos la macro para validar los datos, hay muchas formas de hacer esto, todo dependerá de los tipos de datos y de las necesidades de tu proyecto. Reitero, la validación es un tema central en muchos programas, valida a conciencia, siempre.

Usaremos una función para validar todos los datos.

```
'Función para validar los datos que capture el usuario
Function ValidarDatos() As Boolean
Dim sDato As String
Dim col As Byte
Dim sLetra As String

   'Obtenemos el valor del cuadro de texto y le quitamos los espacios sobrantes
   sDato = Trim(txtNombre.getText())
   'Validamos que si esta vacio, si es un número o una fecha
   If sDato = "" Or IsNumeric( sDato ) Or IsDate( sDato )Then
         MsgBox "El campo NOMBRE no puede estar vacío, es un número o una fecha", 16, "Error en campo"
         'Vaciamos el contenido del control
         txtNombre.setText( "" )
         'Enviamos el cursor al control
         txtNombre.setFocus()
         'Salimos de la función, ValidarDatos = False
         Exit Function
   Else
         'Si el valor es correcto, lo establecemos en el control para su posterior guardado
         'aquí puedes por ejemplo; convertir en mayusculas
         txtNombre.setText( sDato )
   End If

   'Lo mismo para todos los restantes campos
   sDato = Trim(txtCalle.getText())
   If sDato = "" Or IsNumeric( sDato ) Or IsDate( sDato )Then
         MsgBox "El campo CALLE no puede estar vacío, es un número o una fecha", 16, "Error en campo"
         txtCalle.setText( "" )
         txtCalle.setFocus()
         Exit Function
   Else
         txtCalle.setText( sDato )
   End If

   'Tal vez en este campo quieras aceptar números, yo creo que es mejor que no y obligar a capturar
   'por ejemplo: Nº 7, Mz 43 Lt 7, Edif 8 Depto 6
   sDato = Trim(txtNumero.getText())
   If sDato = "" Or IsNumeric( sDato ) Or IsDate( sDato )Then
         MsgBox "El campo NUMERO no puede estar vacío, es un número o una fecha", 16, "Error en campo"
         txtNumero.setText( "" )
         txtNumero.setFocus()
         Exit Function
   Else
         txtNumero.setText( sDato )
   End If

   sDato = Trim(txtColonia.getText())
   If sDato = "" Or IsNumeric( sDato ) Or IsDate( sDato )Then
         MsgBox "El campo COLONIA no puede estar vacío, es un número o una fecha", 16, "Error en
campo"
         txtColonia.setText( "" )
         txtColonia.setFocus()
         Exit Function
   Else
         txtColonia.setText( sDato )
   End If

   sDato = Trim(txtDelegacion.getText())
```

```
    If sDato = "" Or IsNumeric( sDato ) Or IsDate( sDato )Then
          MsgBox "El campo DELEGACION no puede estar vacío, es un número o una fecha", 16, "Error en
campo"
          txtDelegacion.setText( "" )
          txtDelegacion.setFocus()
          Exit Function
    Else
          txtDelegacion.setText( sDato )
    End If

    'Este es un caso especial, en México el CP esta en formato 00000, validaremos
    'que todos los caracteres capturados sean digitos
    sDato = Trim(txtCp.getText())
    If sDato = "" Or IsDate( sDato )Then
          MsgBox "El campo C.P. no puede estar vacío o una fecha", 16, "Error en campo"
          txtCp.setText( "" )
          txtCp.setFocus()
          Exit Function
    Else
          'Iteramos en cada letra
          For co1 = 1 To Len(sDato)
                'Extraemos cada letra
                sLetra = Mid( sDato, co1, 1 )
                'Verificamos que sea un digito
                If InStr( 1, "0123456789", sLetra ) = 0 Then
                      MsgBox "El campo C.P. solo acepta dígitos", 16, "Error en campo"
                      txtCp.setText( "" )
                      txtCp.setFocus()
                      Exit Function
                End If
          Next co1
          'Si son puros dígitos, le damos el formato correcto
          sDato = Format( Val(sDato), "00000" )
          txtCp.setText( sDato )
    End If

    sDato = Trim(txtCiudad.getText())
    If sDato = "" Or IsNumeric( sDato ) Or IsDate( sDato )Then
          MsgBox "El campo CIUDAD no puede estar vacío, es un número o una fecha", 16, "Error en campo"
          txtCiudad.setText( "" )
          txtCiudad.setFocus()
          Exit Function
    Else
          txtCiudad.setText( sDato )
    End If

    'Este es otro caso especial, el RFC es una combinación de letras y números que identifica
    'de forma única a cada contribuyente, para personas morales es de 12 caracteres y para las
    'personas físicas es de 13 con el formato [L]LLLDDDDDDAAA
    sDato = Trim(txtRfc.getText())
    If sDato = "" Or IsNumeric( sDato ) Or IsDate( sDato )Then
          MsgBox "El campo RFC no puede estar vacío, es un número o una fecha", 16, "Error en campo"
          txtRfc.setText( "" )
          txtRfc.setFocus()
          Exit Function
    Else
          'Esta es tu tarea, validar que este bien capturado el RFC
          txtRfc.setText( sDato )
    End If

    'Si todos los datos están correctos
    ValidarDatos = True

End Function
```

¿Recuerdas que comentamos que muchas veces la validación nos lleva muchas más líneas de código que otros procesos?, aquí una prueba. Entre las propiedades de los controles, por ejemplo el cuadro de texto para el campo CP, lo hemos limitado a solo permitir cinco caracteres, las propiedades de tu tabla y la validación por código, deberías de tener completo control sobre los datos capturados por el usuario, un refuerzo más podría ser un controlador de errores como hemos aprendido ya, queda a tu criterio el establecerlo o no. La validación para el RFC te queda de tarea, en el archivo que acompaña a estos apuntes puedes ver mi propuesta, pero por supuesto, intenta antes hacer la tuya, y claro, no tienen que coincidir plenamente, recuerda que en informática (y en casi todo) hay más de una manera de resolver los problemas. Hay otro caso, dentro de la validación que tienes que considerar, pero este es tan obvio que también te queda de tarea averiguarlo.

Ahora, modifica la macro cmdGuardar_Clic para integrar la validación anterior.

```
Sub cmdGuardar_Clic()
Dim mDatos()
Dim co1 As Integer
Dim sTmp As String
Dim sSQL As String

   If ValidarDatos() Then
         'Obtenemos los datos de todos los cuadros de texto
         mDatos = Array( txtNombre.getText, txtCalle.getText, txtNumero.getText, txtColonia.getText,
txtDelegacion.getText, txtCp.getText, txtCiudad.getText, txtRfc.getText, )
         'Agregamos las comillas simples
         For co1 = LBound(mDatos) To UBound(mDatos)
               mDatos(co1) = "'" & mDatos(co1) & "'"
         Next co1
         'Juntamos y separamos por coma
         sTmp = Join( mDatos(), ",")
         'Construimos la instrucción SQL de inserción
         sSQL = "INSERT INTO ""Clientes""
(""Nombre"",""Calle"",""Numero"",""Colonia"",""Delegacion"",""Cp"",""Ciudad"",""Rfc"") VALUES (" &
sTmp & ")"
         'Insertamos los datos
         Call ActualizarDatos( oDatos, sSQL )

         'Activamos y desactivamos los demás controles necesarios
         cmdEditar.setEnable( True )
         cmdEliminar.setEnable( True )
         lstClientes.setEnable( True )
         cmdNuevo.setEnable( True )
         cmdSalir.setEnable( True )
         cmdGuardar.setEnable( False )
         cmdCancelar.setEnable( False )
         'Desactivamos los cuadros de texto
         Call ActivarCuadrosTexto( False )
   End If

End Sub
```

Ahora si, veamos el código del evento Clic para el cuadro de lista (*ListBox*) de los clientes, que nos traerá los datos completos del cliente seleccionado por el usuario.

```
Sub lstClientes_Clic()
Dim sCliente As String
Dim sSQL As String
Dim oConsulta As Object

   'El cliente seleccionado
   sCliente = lstClientes.getSelectedItem
   'Construimos la consulta SQL
```

```
    sSQL = "SELECT * FROM Clientes WHERE Nombre='" & sCliente & "'"
    'Hacemos la consulta
    oConsulta = HacerConsulta( oDatos, sSQL )
    'Mostramos los datos en sus respectivos controles
    With oConsulta
          txtClave.setText( .getInt(1) )
          txtNombre.setText( .getString(2) )
          txtCalle.setText( .getString(3) )
          txtNumero.setText( .getString(4) )
          txtColonia.setText( .getString(5) )
          txtDelegacion.setText( .getString(6) )
          txtCp.setText( .getString(7) )
          txtCiudad.setText( .getString(8) )
          txtRfc.setText( .getString(9) )
    End With

End Sub
```

El siguiente botón que vamos a programar es el de ***Eliminar***, que queda así.

```
Sub cmdEliminar_Clic()
Dim sClave As String
Dim sNombre As String
Dim sInfo As String
Dim iRes As Integer
Dim sSQL As String
Dim iSel As Integer
Dim iNum As Integer

   'Posición del cliente seleccionado
   iSel = lstClientes.getSelectedItemPos
   'La clave del cliente
   sClave = txtClave.getText
   'El nombre del cliente
   sNombre = txtNombre.getText
   'Construimos el mensaje a mostrar
   sInfo = "Se eliminará al cliente " & sNombre & " con clave " & sClave & Chr(13) & Chr(13) &
"¿Estás seguro de continuar?" & Chr(13) & Chr(13) & "ESTA ACCIÓN NO SE PUEDE DESHACER"
   'Mostramos y preguntamos al usuario
   iRes = MsgBox( sInfo, 36,"Eliminar cliente" )
   'Si el usuario responde Si
   If iRes = 6 Then
         'Construimos la instrucción SQL de borrado
         sSQL = "DELETE FROM ""Clientes"" WHERE ""Id""=" & sClave
         'Actualizamos la base de datos
         Call ActualizarDatos( oDatos, sSQL )
         'Quitamos al cliente de la lista
         lstClientes.removeItems( iSel,1)
         'Obtenemos el número de clientes
         iNum = lstClientes.getItemCount
         Select Case iNum
               Case 0      'Si ya no hay clientes
                     cmdEditar.setEnable( False )
                     cmdEliminar.setEnable( False )
                     lstClientes.setEnable( False )
                     Call LimpiarCuadrosTexto( False )
                     txtNombre.setFocus()
               Case 1      'Si al menos hay un cliente lo seleccionamos
                     lstClientes.selectItemPos( 0, True )
                     'Traemos sus datos
                     Call lstClientes_Clic()
               Case Else
                     'Solo si se eliminó el ultimo cliente
                     If iNum = iSel Then iSel = iSel - 1
```

```
                    'Seleccionamos el inmediato anterior
                    lstClientes.selectItemPos( iSel, True )
                    'Traemos sus datos
                    Call lstClientes_Clic()
            End Select
    End If

End Sub
```

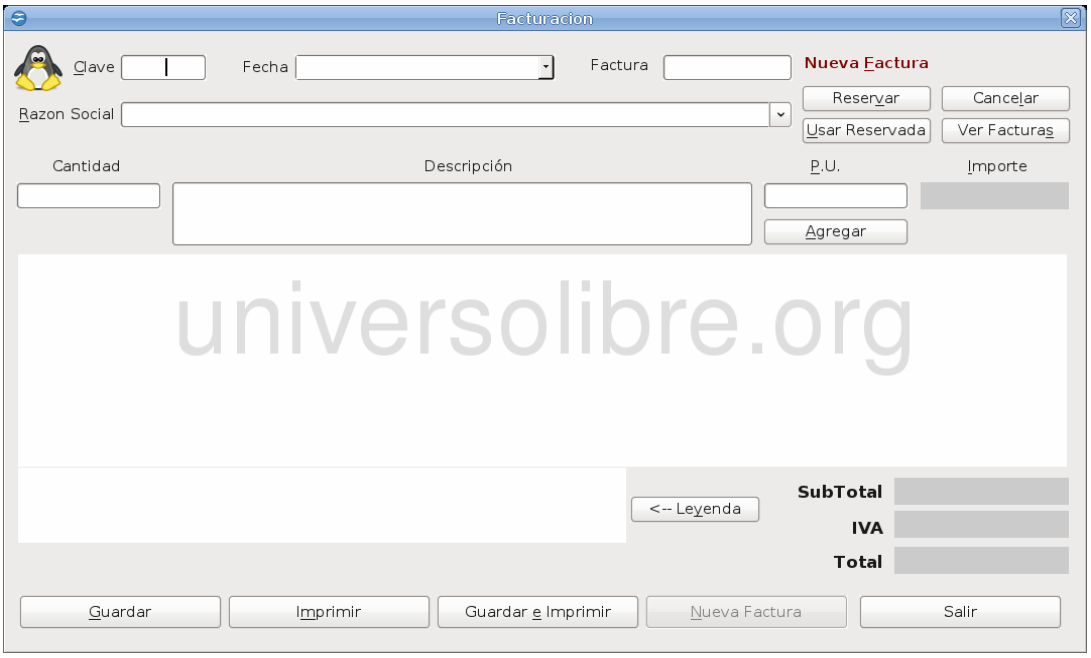
Siempre y perdona el absoluto, siempre pregunta al usuario y trata de informarle lo más claro posible, cuando se esta apunto de realizar una acción que no es posible deshacer, dependiendo del contexto, tal vez hasta dos veces hay que preguntar. En el código anterior nos hace falta una validación **antes** de eliminar al cliente, ¿cual?.

El botón de ***Editar*** tiene un comportamiento muy similar al botón ***Nuevo***, con la diferencia de que no vaciamos los cuadros de texto y activamos otros botones de comando.

```
Sub cmdEditar_Clic()
   'Activamos los cuadros de texto
   Call ActivarCuadrosTexto( True )
   'Botones activados
   cmdActualizar.setEnable( True )
   cmdCancelar.setEnable( True )
   'Botones desactivados
   cmdNuevo.setEnable( False )
   cmdEditar.setEnable( False )
   cmdGuardar.setEnable( False )
   cmdEliminar.setEnable( False )
   cmdSalir.setEnable( False )
   'Desactivamos el cuadro de lista
   lstClientes.setEnable( False )
   'Enviamos el cursor al primer campo
   txtNombre.setFocus()
End Sub
```

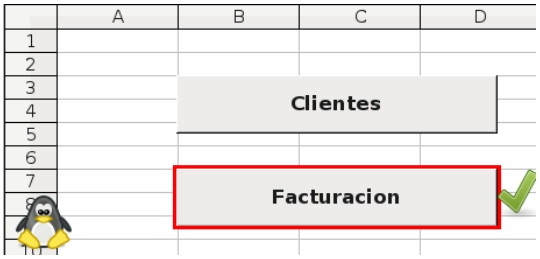
El código del botón ***Actualizar***, es similar al del botón ***Guardar***, pero claro, hay que hacer algunas validaciones un poco diferentes. Te queda de tarea terminarlo, en el archivo anexo de este libro esta mi propuesta, pero de nuevo, no hagas trampa y trata de terminarlo primero, después solo compara recordando que es solo una propuesta.

El siguiente cuadro de diálogo que haremos será el de la facturación, en este ya estarán involucradas dos tablas diferentes, pero su manipulación es muy similar al que acabamos de terminar aunque veras que necesitaremos muchas más líneas de código para poder controlarlo, diseña tu cuadro de dialogo de acuerdo a la siguiente imagen.



En el archivo anexo podrás ver el detalle de los controles usados, así como las principales propiedades usadas en ellos.

En nuestra hoja de calculo de inicio, agrega un segundo botón de comando que abrirá este cuadro de diálogo.



Todo el código de este cuadro de diálogo estará en un nuevo modulo, así que agrega uno al archivo desde el IDE como ya sabes. Lo primero que tenemos que hacer es declarar las variables a nivel de modulo que serán visibles para todas las macros y funciones que desarrollemos en él.

```
Option Explicit

Dim oDatos As Object
Dim cboClientes As Object
Dim txtClave As Object
Dim txtFecha As Object
Dim txtFactura As Object
Dim txtCantidad As Object
Dim txtDescripcion As Object
Dim txtPu As Object
Dim txtImporte As Object
Dim txtSubtotal As Object
Dim txtImpuesto As Object
Dim txtTotal As Object
Dim lstDetalle As Object
Dim lstLeyendas As Object
Dim lblImpuesto As Object
```

```
Dim lblEstado As Object
Dim cmdNuevaFactura As Object
Dim cmdAgregar As Object
Dim cmdGuardar As Object
Dim cmdReservar As Object
Dim cmdUsarReservada As Object

Type Articulo
   Cantidad As Double
   Descripcion As String
   Precio As Double
   Importe As Double
End Type

Dim EsReservada As Boolean
Dim dSubtotal As Double
Dim mDetalle() As New Articulo
```

La siguiente macro será la macro de inicio y la que se tiene que asociar al botón de comando agregado a la interfaz del usuario. Observa que hacemos uso de varias macros ya vistas en el cuadro de diálogo anterior, como; ***CargarDialogo***, ***ExistenDatos*** y ***TotalRegistros***.

```
Sub Facturacion()
Dim oDlg As Object
Dim lFactura As Long
Dim sFactura As String

   'Cargamos el cuadro de diálogo
   oDlg = CargarDialogo( "Standard","dlgFacturacion" )

   'Nos aseguramos de que sea válido
   If Not IsNull(oDlg) Then
         'Nos aseguramos de que existan los datos
         oDatos = ExistenDatos()
         If Not IsNull( oDatos ) Then
               'Nos aseguramos de que haya clientes registrados
               If TotalRegistros( oDatos, "Clientes", "Id" ) > 0 Then
                     'Cargamos los controles en memoria
                     Call CargarControles( oDlg )
                     txtDescripcion.setText( "" )
                     cboClientes.setText( "" )
                     lblImpuesto.setText( "IVA " & Format(IVA,"0 %" )
                     'Traemos los nombres de los clientes si lo hay
                     Call TraerClientes( oDatos )
                     'Establecemos la fecha actual
                     txtFecha.Date = Format(Now,"YYYYMMDD")
                     'Consultamos la ultima factura
                     lFactura = UltimaFactura( oDatos )
                     'Si es la primer factura pedimos él número
                     If lFactura = 0 Then
                           Do
                                 sFactura = InputBox( "Es la primer factura que haces, ¿escribe el
número de factura?" )
                           Loop Until IsNumeric(sFactura)
                           lFactura = Val( sFactura )
                     Else
                           'Si ya hay solo aumentamos en uno
                           lFactura = lFactura + 1
                     End If
                     'Si hay al menos una factura reservada activamos el botón de comando
                     If Not ExisteValor( oDatos, "Facturas", "Estado", "Reservada") Then
                           cmdUsarReservada.setEnable( False )
                     End If
```

```
                    'Establecemos el número de la nueva factura
                    txtFactura.setText( lFactura )
                    txtClave.setFocus()
                    'Mostramos el cuadro de diálogo
                    oDlg.execute()
                    'Lo liberamos
                    oDlg.dispose()
              Else
                    MsgBox "No hay clientes registrados, al menos registra uno", 16, "Sin clientes"
              End If
        Else
              MsgBox "No se encontró la base de datos", 16, "Error grave"
        End If
   Else
        MsgBox "No se pudo mostrar el cuadro de diálogo de Clientes"
   End If

End Sub
```

La macro ***CargarControles*** es igual pero claro, en esta cargamos los controles usado en este cuadro de diálogo, no te confundas, son dos macros con igual nombre pero contenido diferente, al estar en módulos diferentes, el IDE sabe a cual llamar.

```
'Para inicializar las variables de todos los controles usados
Sub CargarControles( Dialogo As Object)
   With Dialogo
        cboClientes = .getControl("cboClientes")
        txtClave = .getControl("txtClave")
        txtFecha = .getControl("txtFecha")
        txtFactura = .getControl("txtFactura")
        txtCantidad = .getControl("txtCantidad")
        txtDescripcion = .getControl("txtDescripcion")
        txtPu = .getControl("txtPu")
        txtImporte = .getControl("txtImporte")
        lstDetalle = .getControl("lstDetalle")
        lstLeyendas = .getControl("lstLeyendas")
        txtSubtotal = .getControl("txtSubtotal")
        txtImpuesto = .getControl("txtImpuesto")
        txtTotal = .getControl("txtTotal")
        lblImpuesto = .getControl("lblImpuesto")
        lblEstado = .getControl("lblEstado")
        cmdNuevaFactura = .getControl("cmdNuevaFactura")
        cmdAgregar = .getControl("cmdAgregar")
        cmdGuardar = .getControl("cmdGuardar")
        cmdReservar = .getControl("cmdReservar")
        cmdUsarReservada = .getControl("cmdUsarReservada")
   End With
End Sub
```

Observa que en la siguiente línea.

```
lblImpuesto.setText( "IVA " & Format(IVA,"0 %" )
```

Establecemos el texto de una etiqueta (Label), y usamos el valor de una constante llamada IVA, esta constante la declaramos en el mismo modulo de las otras constantes. Nota las nuevas constantes que usaremos, por supuesto, no son limitativas, agrega todas las que consideres que usaras en tú sistema.

```
Option Explicit
```

```
'Cambia aquí el nombre de tu conexión
Global Const DATOS As String = "datos"

Global Const AMARILLO_PASTEL As Long = 16777164
Global Const BLANCO As Long = 16777215
Global Const IVA As Single = 0.16
Global Const COPIAS As Byte = 3
Global Const MAX_COPIAS As Byte = 5
```

La macro ***TraerClientes*** de este modulo, es un poco diferente.

```
'Traemos todos los clientes de la base de datos
Sub TraerClientes( oDatos )
Dim oConsulta As Object
Dim sSQL As String
Dim mClientes()
Dim lTotalRegistros As Long

   'Obtenemos el total de clientes
   lTotalRegistros = TotalRegistros( oDatos, "Clientes", "Id" )
   'Construimos la consulta para regresar los clientes ordenados por Nombre
   sSQL = "SELECT Nombre FROM Clientes ORDER BY Nombre"
   'Hacemos la consulta
   oConsulta = HacerConsulta( oDatos, sSQL )
   'Movemos los datos a una matriz
   mClientes = CampoAMatriz( oConsulta, lTotalRegistros, 1 )
   'Y los agregamos al cuadro de lista
   Call MatrizACuadroLista( mClientes(), cboClientes )

End Sub
```

La función ***UltimaFactura*** tiene el siguiente código.

```
'Para obtener el número de la ultima factura guardada
Function UltimaFactura( oDatos ) As Long
Dim sSQL As String
Dim oConsulta As Object

   sSQL = "SELECT MAX(Numero) FROM Facturas"
   oConsulta = HacerConsulta( oDatos, sSQL )
   UltimaFactura = oConsulta.getLong( 1 )

End Function
```

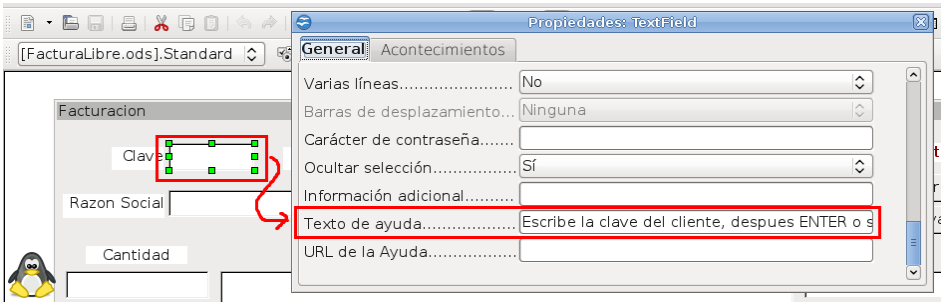
Observa que obtenemos el máximo del campo Numero, esto no necesariamente tiene que ser así, tal ves la estructura de tú tabla te permita obtener solo el ultimo valor agregado, esto tú lo determinas. La función ***ExisteValor*** tiene el siguiente código.

```
'Función para saber si un registro ya fue dado de alta
Function ExisteValor( BaseDatos As Object, Tabla As String, Campo As String, Valor As String) As
Boolean
Dim sSQL As String
Dim oConsulta As Object

   'Construimos la consulta
   sSQL = "SELECT " & Campo & " FROM " & Tabla & " WHERE " & Campo & " = '" & Valor & "'"

   'Hacemos la consulta
   oConsulta = HacerConsulta( BaseDatos, sSQL )
   'Verificamos si existe
   If oConsulta.getRow = 1 Then
```

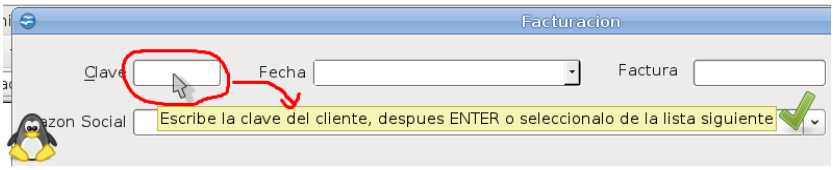
```
        ExisteValor = True
    End If

End Function
```

El resto del código creo que es autoexplicativo.

El primero control que codificaremos será el cuadro de texto ***txtClave***, usaremos su evento ***Tecla Pulsada*** y buscaremos el cliente con dicha clave cuando el usuario presione la tecla ***Enter***. Esta información de ayuda, puedes establecerla en la propiedad ***Texto de ayuda*** de cada control, si tu sistema no viene (que debería) acompañado de un archivo de ayuda, esta propiedad puede ser muy util para mostrarle al usuario que se espera que capture o que haga y se muestra en cuanto el cursor entra en el área del control.



Que mostrará en tiempo de ejecución.



El código de este evento es.

```
'Cuando el usuario presiona una tecla en el cuadro texto Clave del cliente
Sub txtClave_TeclaPulsada( Evento )
Dim lClave As Long
Dim sCliente As String

   'Si la tecla presiona es la tecla ENTER buscamos al cliente por clave
   If Evento.KeyCode = 1280 Then
         lClave = Val( txtClave.getText )
         txtClave.setText( Format(lClave) )
         sCliente = ExisteCliente( oDatos, Format(lClave), True )
         cboClientes.setText( sCliente )
         If sCliente = "" Then
               MsgBox "El cliente con la clave " & Format(lClave) & " no existe", 16,"Facturacion"
         Else
               txtCantidad.setFocus()
         End If

   End If

End Sub
```

El código de la tecla ***Enter*** es 1280, tomamos el valor del cuadro de texto, lo convertimos a valor y buscamos esta clave de cliente, el código de la función ***ExisteCliente*** es el siguiente. Observa que dependiendo del tercer argumento (Clave) podemos hacer la busqueda

por la clave o por el nombre del cliente, en los dos casos regresamos una cadena con el nombre o con la clave del cliente encontrado o una cadena vacia si no es contrado.

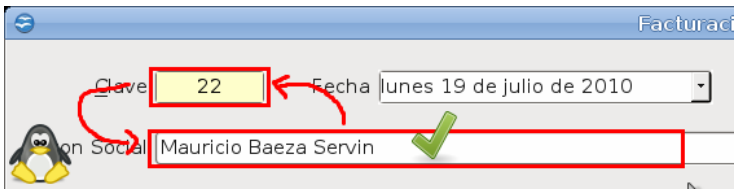
```
'Para saber si un cliente existe, podemos buscar por cliente o por clave
Function ExisteCliente( oDatos, Dato As String, Clave As Boolean ) As String
Dim sSQL As String
Dim oConsulta As Object

   'Construimos la consulta correcta
   If Clave Then
         sSQL = "SELECT Nombre FROM Clientes WHERE Id=" & Dato
   Else
         sSQL = "SELECT Id FROM Clientes WHERE Nombre='" & Dato & "'"
   End If
   oConsulta = HacerConsulta( oDatos, sSQL )
   'Si hay una fila al menos se encontró el cliente
   If oConsulta.getRow > 0 Then
         ExisteCliente = oConsulta.getString( 1 )
   End If

End Function
```

A parte de la clave, ponemos a disposición del usuario, la posibilidad de seleccionar al cliente desde el control cuadro combinado (*ComboBox*), usamos un cuadro combinado, por qué este combinado las virtudes de un cuadro de texto para permitir escribir el nombre directamente o seleccionarlo desde la lista desplegable, por esto, tenemos que controlar tanto el evento ***"Estado modificado"***, que es llamado cuando el usuario selecciona algún elemento de la lista, como el evento ***"Al perder el foco"***, podemos usar también el evento ***"Texto modificado"***, pero al ser un campo donde puede haber muchos caracteres implicados nos decantamos por otros eventos, cuyo código es el siguiente.

```
'Cuando sale el foco del cuadro combinado de los clientes
Sub cboClientes_PierdeFoco( Evento )
Dim sCliente As String
Dim sClave As String

   sCliente = Trim(Evento.Source.getText)
   If sCliente <> "" Then
         sClave = ExisteCliente( oDatos, sCliente, False )
         If sClave = "" Then
               MsgBox "El cliente " & sCliente & " no esta registrado", 16,"Cliente inexistente"
               txtClave.setText( "" )
         Else
               txtClave.setText( sClave )
         End If
   End If

End Sub


'Este evento es llamado cuando se usa el raton para cambiar el elemento dentro del cuadro combinado
Sub cboClientes_Cambia( Evento )
Dim sCliente As String
Dim sClave As String

   'Obtenemos el texto y bucamos al cliente por nombre
   sCliente = Evento.Source.getText
   sClave = ExisteCliente( oDatos, sCliente, False )
   If sClave = "" Then
         MsgBox "El cliente " & sCliente & " no esta registrado", 16,"Cliente inexistente"
         txtClave.setText( "" )
   Else
         txtClave.setText( sClave )
```

```
        txtCantidad.setFocus()
    End If

End Sub
```

Observando el código de los controles ***txtClave*** y ***cboClientes***, puedes deducir fácilmente que uno se complementa con el otro, si el usuario escribe una clave correcta el nombre es mostrado, si el usuario escribe un nombre directamente o selecciona uno existente de la lista desplegable, es mostrada su clave correspondiente.



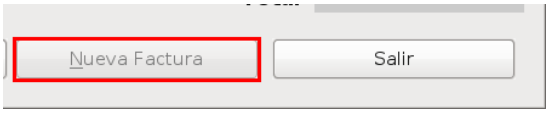
Para el control de la fecha, hemos usado un control fecha (DateField) en donde, al abrir el cuadro de diálogo, mostramos la fecha actual del sistema y las únicas validaciones que hacemos es informarle al usuario si esta usando una fecha anterior o posterior a la actual, en algunos casos tal ves sea necesario evitar que el usuario seleccione una fecha futura, esto, lo dejo a tu criterio, por ahora, lo dejamos a criterio del usuario y solo se lo informamos en la función de validación que implementaremos más adelante.

El botón de comando ***Reservar***, lo usamos para cambiar el estatus de la factura actual a “reservada”, no se guarda ningún otro dato.

El código de este botón es.

```
'Para marcar una factura como reservada para su uso posterior
Sub cmdReservar_Clic()
Dim iRes As Integer
Dim sSQL As String

   iRes = MsgBox( "Esta acción solo marcara esta factura: " & txtFactura.getText() & " como reservada
para uso posterior. No se guardará ningún otro dato" & _
                  Chr(10) & Chr(10) & "¿Estás seguro de marcarla como reservada?", 36,
"Reservar factura" )
   If iRes= 6 Then
        'Solo nuevas facturas se pueden marcar como reservadas
        sSQL = "INSERT INTO ""Facturas"" (""Numero"",""Estado"") VALUES (" & txtFactura.getText() &
",'Reservada')"
        Call ActualizarDatos( oDatos, sSQL )
        cmdUsarReservada.setEnable( True )
        Call cmdNuevaFactura_Clic()
   End If

End Sub
```

Como usaremos el botón de comando Nueva Factura, mostramos su apariencia, recuerda que de forma predeterminada esta desactivado, más adelante te muestro en que caso queda activado.



Y su código.

```
'Para preparar el ingreso de una nueva factura
Sub cmdNuevaFactura_Clic()
Dim lFactura As Long

   dSubtotal = 0
   EsReservada = False
   Erase mDetalle
   Call ActualizarTotal( 0 )
   lFactura = UltimaFactura( oDatos ) + 1
   txtFactura.setText( Format(lFactura) )
   txtClave.setText( "" )
   cboClientes.setText( "" )
   txtCantidad.setText( "" )
   txtDescripcion.setText( "" )
   txtPu.setText( "" )
   lstDetalle.removeItems( 0, lstDetalle.getItemCount )
   lstLeyendas.removeItems( 0, lstLeyendas.getItemCount )
   cmdNuevaFactura.setEnable( False )
   cmdAgregar.setEnable( True )
   cmdGuardar.setEnable( True )
   cmdReservar.setEnable( True )
   cmdUsarReservada.setEnable( True )
   lblEstado.setText( "Nueva Factura" )
   txtClave.setFocus()

End Sub
```

El botón de comando ***Cancelar***, nos sirve para establecer el estatus de la factura actual como “cancelada”, podemos cancelar tanto una factura nueva como una existente.

El código de este botón es.

```
'Botón de comando Cancelar
Sub cmdCancelar_Clic()
Dim iRes As Integer
Dim sSQL As String

   iRes = MsgBox( "Esta acción NO SE PUEDE DESHACER la factura: " & txtFactura.getText() & " se
marcará como CANCELADA." & Chr(10) & Chr(10) & "¿Estás seguro de CANCELAR esta factura?", 36,
"Cancelar factura" )
   'Nos aseguramos de la respuesta del usuario
   If iRes= 6 Then
         If ExisteValor( oDatos, "Facturas", "Numero", txtFactura.getText() ) Then
               'Si la factura ya existe solo actualizamos su estado
               sSQL = "UPDATE ""Facturas"" SET ""Estado""='Cancelada' WHERE ""Numero""=" &
txtFactura.getText()
         Else
               'Si no existe la insertamos como nueva
```

```
            sSQL = "INSERT INTO ""Facturas"" (""Numero"",""Estado"") VALUES (" &
txtFactura.getText() & ",'Cancelada')"
        End If
        'Hacemos la consulta de actualización
        Call ActualizarDatos( oDatos, sSQL )
        'Preparamos para una nueva factura
        Call cmdNuevaFactura_Clic()
    End If

End Sub
```

El botón ***Usar Reservada***, nos servirá para permitirle al usuario seleccionar y usar una factura previamente marcada como “reservada”.

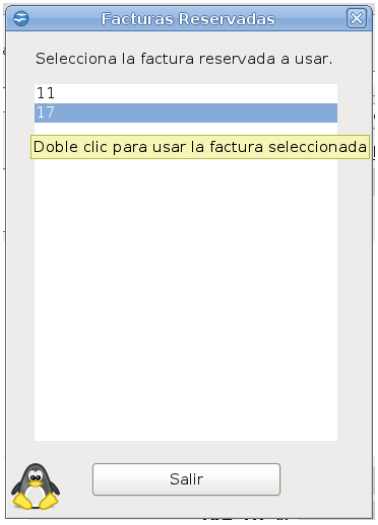
Y su código.

```
'Botón para mostrar las facturas reservadas
Sub cmdUsarReservada_Clic()
Dim oDlg As Object
Dim sSQL As String
Dim oConsulta As Object
Dim mReservadas()

    'Verificamos si hay facturas reservadas
    If ExisteValor( oDatos, "Facturas", "Estado", "Reservada") then
        'Cargamos el cuadro de diálogo
        oDlg = CargarDialogo( "Standard","dlgReservadas" )
        'Traemos las facturas reservadas
        sSQL = "SELECT Numero FROM Facturas WHERE Estado='Reservada'"
        oConsulta = HacerConsulta( oDatos, sSQL )
        'Contamos cuantas son
        sSQL = "SELECT COUNT(Estado) FROM Facturas WHERE Estado='Reservada'"
        mReservadas = CampoAMatriz( oConsulta, TotalRegistrosConsulta( oDatos, sSQL ), 3 )
        Call MatrizACuadroLista( mReservadas(), oDlg.getControl( "lstReservadas" ) )
        'Mostramos el cuadro de diálogo
        oDlg.execute()
        oDlg.dispose()
    Else
        MsgBox "No hay facturas reservadas", 16, "Sin facturas reservadas"
        txtClave.setFocus()
    End If

End Sub
```

La macro anterior usa la siguiente macro nueva.

```
'Función que devuelve el total de registros de una consulta
Function TotalRegistrosConsulta( BaseDatos As Object, SQL As String) As Long
Dim oConsulta As Object

    oConsulta = HacerConsulta( BaseDatos, SQL )
    TotalRegistrosConsulta = oConsulta.getLong(1)

End Function
```

Observa que mostramos las facturas en un cuadro de lista de un nuevo cuadro de diálogo que solo tiene dicho cuadro de lista, una etiqueta y un botón de comando para salir.



Con doble clic sobre la factura deseada, la seleccionamos, salimos del cuadro de diálogo y establecemos esta factura como actual en el cuadro de diálogo de facturación. El código de este evento es el siguiente.

```
'Evento doble clic para seleccionar una factura reservada
Sub lstReservadas_Doble_Clic( Evento )
Dim sFactura As String

   'Contamos el número de clics y si hay un elemento seleccionado
   If Evento.ClickCount = 2 And ( Evento.Source.getSelectedItemPos > -1 ) Then
         sFactura = Evento.Source.getSelectedItem
         txtFactura.setText( sFactura )
         lblEstado.setText( "Reservada" )
         cmdReservar.setEnable( False )
         cmdNuevaFactura.setEnable( True )
         'Finalizamos el cuadro de diálogo
         Evento.Source.getContext.endExecute()
         EsReservada = True
   End If

End Sub
```

Observa que le mostramos al usuario que es una factura reservada, actualizando el texto de la etiqueta lblEstado, también desactivamos el botón ***Reservar***.

Los campos Cantidad y P.U. son controles numéricos (*NumericField*), por lo que solo permiten la introducción de números, como el cambio de cualquiera de ellos afecta al importe, es decir a su producto, haremos una sola macro para los dos que asignaremos a su evento ***Texto modificado*** y cuyo código es el siguiente.

```
'Cada vez que cambia la cantidad o el precio
'actualizamos el importe
Sub Importe_Modificado( Evento )
Dim dCantidad As Double
Dim dPU As Double
```

```
Dim dImporte As Double

   dCantidad = txtCantidad.getValue
   dPU = txtPu.getValue
   dImporte = dCantidad * dPU
   'Nos aseguramos de que el resultado tenga solo dos decimales
   dImporte = FuncionCalc( "ROUND", Array(dImporte,2) )
   txtImporte.setValue( dImporte )

End Sub
```

La función ***FuncionCalc***, nos permite llamar a cualquier función integrada de Calc, pasarle los argumentos respectivos y obtener su resultado, esta función ya la hemos visto pero dado que es muy breve la repetimos aquí.

```
'Función para llamar a función incorporada de Calc, es importante
'pasarle los argumentos correctamente en una matriz de datos y
'usar la función deseada con su nombre en ingles
Function FuncionCalc( Nombre As String, Datos() )
Dim oSFA As Object

   oSFA = createUnoService( "com.sun.star.sheet.FunctionAccess" )
   FuncionCalc = oSFA.callFunction( Nombre, Datos() )

End Function
```

Para mostrar el importe, usamos un control moneda (*CurrencyField*) por lo que solo establecemos el valor, el formato lo establecemos como propiedad de este control.

El botón de comando ***Agregar*** que puedes ver en la imagen anterior, nos sirve para agregar el artículo o producto a la lista de detalle de nuestra factura, su código es el siguiente.

```
'Para agregar un artículo al detalle de la factura
Sub cmdAgregar_Clic()
Dim dCantidad As Double
Dim dPU As Double
Dim dImporte As Double
Dim sDescripcion As String
Dim sInfo As String

   'La cantidad, el precio y el importe
   dCantidad = txtCantidad.getValue
   dPU = txtPu.getValue
   dImporte = txtImporte.getValue
   'La descripcion
   sDescripcion = Trim( txtDescripcion.getText )
   'Como en el cuadro de texto permitimos varias líneas, el usuario puede introducir
   'saltos de líneas innecesarios al inicio y al final de la cadena
   sDescripcion = QuitarSaltosInicio( sDescripcion )
   sDescripcion = QuitarSaltosFin( sDescripcion )
   'Establecemos la cadena ya limpia
   txtDescripcion.setText( sDescripcion )
   'La cantidad y el precio no pueden ser cero
   If dCantidad > 0 Then
         If dPU > 0 Then
               'La descripción no puede estar vacia
```

```
            If sDescripcion <> "" Then
                  'Compañero si facturas artículos por más de un millón
                  'que esperas para hacerte mi mecenas
                  If dImporte < 1000000 Then
                        'Agregamos el artículo y actualizamos el total
                        Call AgregarArticulo( dCantidad, sDescripcion, dPU, dImporte)
                        Call ActualizarTotal( dImporte )
                        'Limpiamos los campos para un nuevo ingreso
                        txtCantidad.setText( "" )
                        txtDescripcion.setText( "" )
                        txtPu.setText( "" )
                        txtCantidad.setFocus()
                  Else
                        MsgBox "El importe es muy alto, verificalo", "16", "Importe"
                        txtPu.setFocus
                  End If
            Else
                  MsgBox "La descripción no puede estar vacía", 16, "Descripción"
                  txtDescripcion.setFocus
            End If
      Else
            MsgBox "El P.U. no puede ser cero", 16, "Precio Unitario"
            txtPu.setFocus
      End If
   Else
      MsgBox "La cantidad no puede ser cero", 16, "Cantidad"
      txtCantidad.setFocus
   End If
End Sub
```

Las nuevas funciones y macros usados en este evento son.

```
'Función recursiva para quitar los saltos de línea
'que estén al inicio de la cadena pasada
Function QuitarSaltosInicio( Cadena ) As String
Dim co1 As Integer

   'Si la cadena esta vacia salimos de la función
   If Len(Cadena) = 0 Then
         QuitarSaltosInicio = ""
         Exit Function
   'Si el primer caracter de la izquiera NO es un salto de línea
   'regresamos la cadena y salimos de la función
   ElseIf Left(Cadena,1) <> Chr(10) Then
         QuitarSaltosInicio = Cadena
         Exit Function
   Else
         'Si es un salto de línea, lo omitimos y regresamos el resto de la cadena
         Cadena = Mid( Cadena,2,Len(Cadena) )
         'Volvemos a llamar a la función
         QuitarSaltosInicio = QuitarSaltosInicio(Cadena)
   End If

End Function


'Función recursiva igual a la anterior
'pero los quita al final de la cadena pasada
Function QuitarSaltosFin( Cadena ) As String
Dim co1 As Integer

   If Len(Cadena) = 0 Then
         QuitarSaltosFin = ""
```

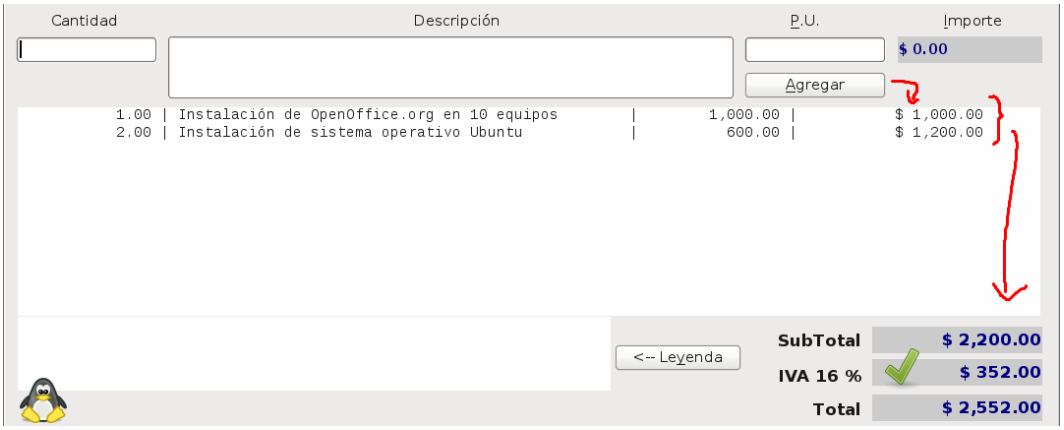
```
        Exit Function
    ElseIf Right(Cadena,1) <> Chr(10) Then
        QuitarSaltosFin = Cadena
        Exit Function
    Else
        Cadena = Left( Cadena,Len(Cadena)-1 )
        QuitarSaltosFin = QuitarSaltosFin(Cadena)
    End If

End Function
```

Ten cuidado con las funciones recursivas, establece siempre como primera o primeras condiciones aquellas con las que salimos de la función. Con un poco de ingenio puedes juntar estas dos funciones en una sola, esa es tu tarea. Las macros restantes son.

```
'Para agregar un artículo al cuadro de lista
Sub AgregarArticulo( Cantidad, Descripcion, Precio, Importe)
Dim sTmp As String
Dim iIndice As Integer
Dim sLinea As String
Dim mTmp()

    'Obtenemos el nuevo indice para la matriz
    If IsNull( mDetalle ) Then
        iIndice = 0
    Else
        iIndice = UBound(mDetalle)+1
    End If
    'y redimensionamos con este indice
    Redim Preserve mDetalle( iIndice )
    'Guardamos los datos en el nuevo elemento de la matriz
    With mDetalle( iIndice )
        .Cantidad = Cantidad
        .Descripcion = Descripcion
        .Precio = Precio
        .Importe = Importe
    End With
    'Formateamos la línea de texto para agregarla al cuadro de lista a cada campo
    'le asignamos un espacio concreto y rellenamos con espacio en blanco el resto
    sTmp = Format( Cantidad, "#,##0.00" )
    sLinea = Space(15-Len(sTmp)) & sTmp & " | "
    'Cuando la descripción es muy larga, la recortamos solo para mostrarla, la cadena
    'completa original queda guardada correctamente en la matriz de detalle
    sTmp = Left(Descripcion,47)
    'Averiguamos si hay saltos de línea
    mTmp = Split( sTmp, Chr(10) )
    If Len(sTmp) >= 47 Then
        'Como los saltos los cuenta pero no se ven, los aumentamos como espacios
        sTmp = sTmp & Space(UBound(mTmp)) & "..."
    Else
        sTmp = sTmp & Space( 50-Len(sTmp) ) & Space(UBound(mTmp))
    End If
    sLinea = sLinea & sTmp & " | "
    'Quince espacios para el precio
    sTmp = Format( Precio, "#,##0.00" )
    sLinea = sLinea & Space(15-Len(sTmp)) & sTmp & " | "
    'Veinte para el importe
    sTmp = Format( Importe, "$ #,##0.00" )
    sLinea = sLinea & Space(20-Len(sTmp)) & sTmp
    'Agregamos la línea formateada completa
    lstDetalle.addItem( sLinea, lstDetalle.getItemCount )

End Sub
```

```
'Para actualizar el total de la factura, hacemos las operaciones necesarios
'damos el formato correcto y los mostramos en los controles correctos
Sub ActualizarTotal( Importe )
Dim dImpuesto As Double
Dim dTotal As Double

   dSubtotal = dSubtotal + Importe
   txtSubtotal.setText( Format( dSubtotal, "$ #,##0.00" ) )
   'Nos aseguramos de que el resultado tenga solo dos decimales
   dImpuesto = FuncionCalc( "ROUND", Array( dSubtotal*IVA, 2 ) )
   txtImpuesto.setText( Format( dImpuesto, "$ #,##0.00" ) )
   dTotal = dSubtotal + dImpuesto
   txtTotal.setText( Format( dTotal, "$ #,##0.00" ) )

End Sub
```

Con el código anterior ya debes de poder agregar artículos. Hay todavía un caso en el que al agregar un artículo, falle el guardado de la factura, tu tarea es encontrarlo, como pista, concentrate en lo que permites guardar en el campo Descripción.



Por supuesto debemos de permitir al usuario quitarlos, para ello usamos el evento Clic del cuadro de lista, pero tiene que dar doble clic.

```
'Para quitar un articulo del detalle
Sub lstDetalle_Doble_Clic( Evento )
Dim Pos As Integer

   If Evento.ClickCount = 2 And ( Evento.Source.getSelectedItemPos > -1 ) Then
         Pos = Evento.Source.getSelectedItemPos
         Evento.Source.removeItems( Pos, 1 )
         Call QuitarArticulo( Pos )
   End If

End Sub
```

La macro ***QuitarArticulo*** tiene el siguiente código.

```
'Al quitar un articulo hay que actualizar el total y quitarlo de la matriz
Sub QuitarArticulo( Indice )
Dim col As Integer
Dim dImporte As Double
Dim mTmp() As New Articulo
```

```
    'Obtenemos el importe y lo ponemos en negativo
    dImporte = mDetalle( Indice ).Importe * -1
    'Actualizamos el total
    Call ActualizarTotal( dImporte )
    'Si hay más de un artículo
    If UBound(mDetalle) > 0 Then
          'Redimencionamos la matriz temporal
          Redim Preserve mTmp( UBound(mDetalle)-1 )
          'Movemos los datos a esta matriz
          For co1 = LBound( mTmp ) To UBound( mTmp )
                'Los que están antes del actual los pasamos tal cual
                If co1 < Indice Then
                      mTmp(co1).Cantidad = mDetalle(co1).Cantidad
                      mTmp(co1).Descripcion = mDetalle(co1).Descripcion
                      mTmp(co1).Precio = mDetalle(co1).Precio
                      mTmp(co1).Importe = mDetalle(co1).Importe
                Else
                      'Los que están después del actual los movemos del inmediato posterior
                      mTmp(co1).Cantidad = mDetalle(co1+1).Cantidad
                      mTmp(co1).Descripcion = mDetalle(co1+1).Descripcion
                      mTmp(co1).Precio = mDetalle(co1+1).Precio
                      mTmp(co1).Importe = mDetalle(co1+1).Importe
                End If
          Next co1
          'Redimenciona la matriz original
          Redim Preserve mDetalle( UBound(mTmp) )
          'Movemos los datos correctos
          For co1 = LBound( mTmp ) To UBound( mTmp )
                mDetalle(co1).Cantidad = mTmp(co1).Cantidad
                mDetalle(co1).Descripcion = mTmp(co1).Descripcion
                mDetalle(co1).Precio = mTmp(co1).Precio
                mDetalle(co1).Importe = mTmp(co1).Importe
          Next co1
    Else
          'Si no quedan elementos borramos la matriz
          Erase mDetalle
    End If

End Sub
```

Todo ese código para mover los datos entre las matrices es por la limitada forma que tienen los lenguajes Basic en la manipulación de matrices.

Verifica que ya puedas agregar y quitar artículos o productos al detalle de la factura y que el importe total se actualice correctamente.

Nuestra siguiente tarea es permitirle al usuario agregar leyendas a la factura, estas, son útiles para agregar notas de cualquier tipo. El código del botón ***Leyenda*** es el siguiente.

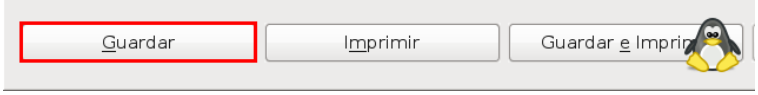
```
'Para agregar una nueva leyenda a la factura
Sub cmdLeyenda_Clic()
Dim sLeyenda As String

   'Cambia aquí el limite de leyendas que quieras permitir, te recomiendo un número pequeño
   'tienes que adaptar el cuadro de lista del formato de impresión para que visualice
   'correctamente este número de líneas
   If lstLeyendas.getItemCount < 4 Then
         sLeyenda = Trim(InputBox( "Introduce la nueva leyenda para esta factura" ))
         If sLeyenda <> "" Then
               lstLeyendas.addItem( sLeyenda, lstLeyendas.getItemCount )
         End If
   Else
         MsgBox "Solo se permiten cuatro líneas de leyendas", 16,"Leyendas"
```

```
   End If

End Sub
```

Y por supuesto el código para quitarlas.

```
'Con doble clic borramos las leyendas del cuadro de lista
Sub lstLeyendas_Doble_Clic( Evento )
Dim Pos As Integer

   If Evento.ClickCount = 2 And ( Evento.Source.getSelectedItemPos > -1 ) Then
         Pos = Evento.Source.getSelectedItemPos
         Evento.Source.removeItems( Pos, 1 )
   End If

End Sub
```

Solo nos faltan tres botones por codificar, veamos el primero que es ***Guardar***.



Y su código.

```
'Para guardar una factura
Sub cmdGuardar_Clic()
Dim sFactura As String
Dim sCliente As String
Dim sFecha As String
Dim sSubtotal As String
Dim sImpuesto As String
Dim sTotal As String
Dim sSQL As String
Dim sTmp As String
Dim col As Integer
Dim sLeyendas As String
Dim iRes As Integer

   'Validamos todos los datos
   If ValidarDatos() Then
         iRes = MsgBox( "Todos los datos con correctos, ¿guardar la factura?", 36, "Guardar factura" )
         'si el usuarios responde que si
         If iRes = 6 Then
               'Obtenemos los datos de la factura, la fecha y los valores los convertimos a texto para
               'su uso correcto en la cadena SQL de inserción
               sFactura = txtFactura.getText()
               sCliente = txtClave.getText()
               sFecha = Format(txtFecha.getDate())
               sFecha = "'" & Left(sFecha,4) & "-" & Mid(sFecha,5,2) & "-" & Right(sFecha,2) & "'"
               sSubtotal = Format(dSubtotal)
               sImpuesto = Format( FuncionCalc( "ROUND", Array( dSubtotal*IVA, 2 ) ) )
               sTotal = Format( dSubtotal + Val(sImpuesto) )
               'Si hay leyendas las juntamos en una sola cadena separa por el caracter pipe (|)
               If lstLeyendas.getItemCount > 0 Then
                     sLeyendas = "'" & Join( lstLeyendas.getItems(), "|" ) & "'"
               Else
                     'Si no hay leyendas, se pasa una cadena vacía, observa el par de comillas simples
                     sLeyendas = "''"
               End If
```

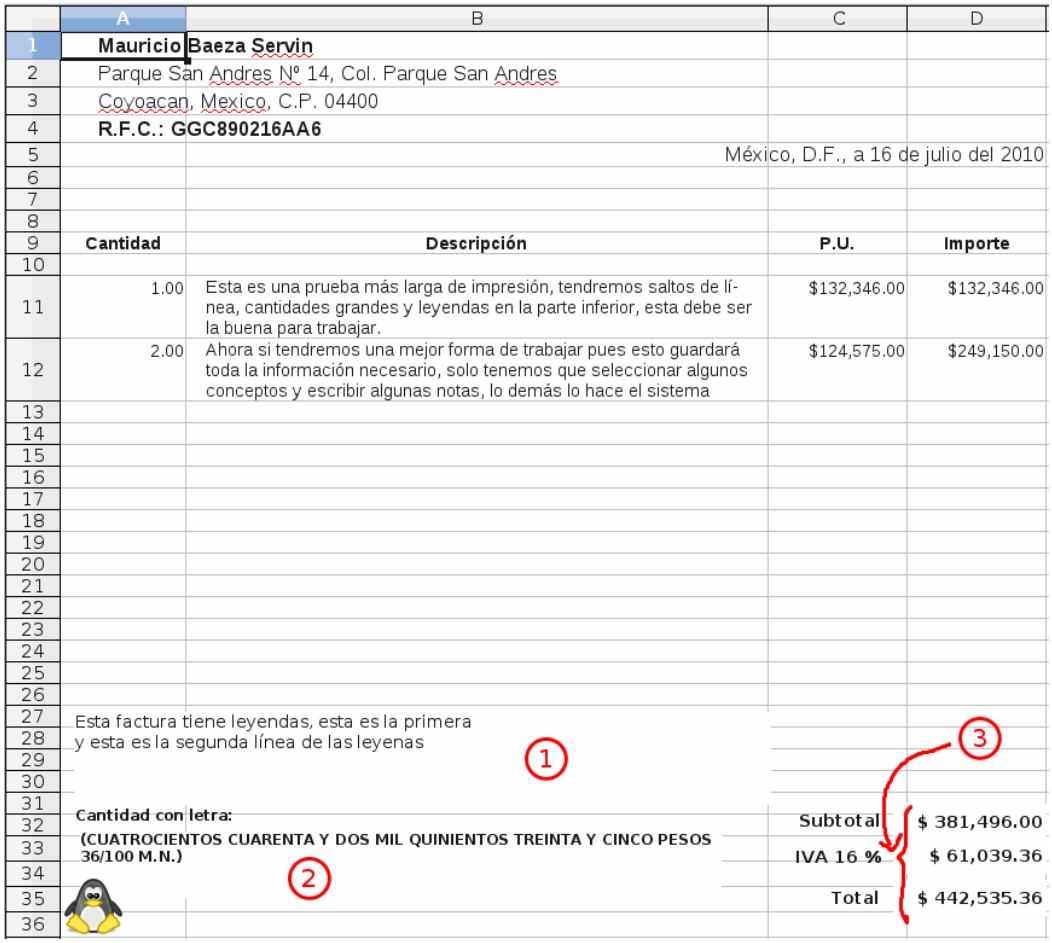
```
                'Si la factura es reservada ya esta guardada por lo que solo actualizamos los campos
restantes
                If EsReservada Then
                      sTmp = Chr(34) & "Cliente" & Chr(34) & "=" & sCliente & ","
                      sTmp = sTmp & Chr(34) & "Fecha" & Chr(34) & "=" & sFecha & ","
                      sTmp = sTmp & Chr(34) & "Subtotal" & Chr(34) & "=" & sSubtotal & ","
                      sTmp = sTmp & Chr(34) & "Impuesto" & Chr(34) & "=" & sImpuesto & ","
                      sTmp = sTmp & Chr(34) & "Total" & Chr(34) & "=" & sTotal & ","
                      sTmp = sTmp & Chr(34) & "Estado" & Chr(34) & "='Elaborada',"
                      sTmp = sTmp & Chr(34) & "Leyendas" & Chr(34) & "=" & sLeyendas
                      sSQL = "UPDATE ""Facturas"" SET " & sTmp & " WHERE ""Numero""=" & sFactura
                Else
                      'Si es nueva usamos todos los campos
                      sTmp = sFactura & "," & sCliente & "," & sFecha & "," & sSubtotal & "," &
sImpuesto & "," & sTotal & ",'Elaborada'," & sLeyendas
                      sSQL = "INSERT INTO ""Facturas""
( ""Numero"",""Cliente"",""Fecha"",""Subtotal"",""Impuesto"",""Total"",""Estado"",""Leyendas"" )
VALUES (" & sTmp & ")"
                End If
                'Insertamos los datos de la factura
                Call ActualizarDatos( oDatos, sSQL )
                'Insertamos el detalle de la factura
                For co1 = LBound( mDetalle ) To UBound( mDetalle )
                      sTmp = sFactura & "," & mDetalle(co1).Cantidad & ",'" &
mDetalle(co1).Descripcion & "'," & mDetalle(co1).Precio & "," & mDetalle(co1).Importe
                      sSQL = "INSERT INTO ""Detalle""
( ""Factura"",""Cantidad"",""Descripcion"",""Precio"",""Importe"" ) VALUES (" & sTmp & ")"
                      Call ActualizarDatos( oDatos, sSQL )
                Next

                'Prepara todo para una nueva factura
                Call cmdNuevaFactura_Clic()

          End If
   End If

End Sub
```

El código de la función ***ValidarDatos*** para este cuadro de diálogo.

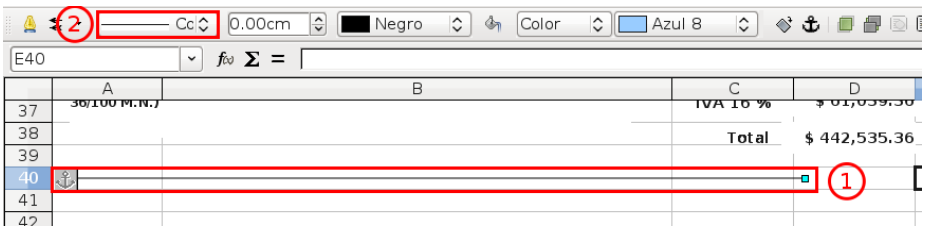
```
'Función para validar todos los datos
Function ValidarDatos() As Boolean
Dim sClave As String
Dim sFecha As String
Dim dFecha As Date
Dim iRes As Integer

   'La clave del cliente
   sClave = Format(Val(txtClave.getText()))
   'Como usamos Val y después Format, en ves de vacía, si no hay nada siempre da 0
   'cuidado, tal vez, dependiendo de tu tabla Clientes, tal vez si puedas tener un
   'cliente con la clave 0
   If sClave <> "0" Then
         'Verificamos que exista el cliente
         If ExisteCliente( oDatos, sClave, True ) <> "" Then
               'Verificamos que se hallan agregado productos
               If lstDetalle.getItemCount > 0 Then
                     'Obtenemos la fecha
                     sFecha = Format(txtFecha.getDate)
                     dFecha = DateSerial( Val(Left(sFecha,4)), Val(Mid(sFecha,5,2)),
Val(Right(sFecha,2)) )
                     'Si es menor a la fecha actual, se puede usar pero le avisamos al usuario
                     If dFecha < Date() Then
```

```
                        iRes = MsgBox( "La fecha " & Format(dFecha,"dd-mmm-yy") & " es una fecha
pasada" & Chr(10) & Chr(10) & "¿Estás seguro de usar esta fecha?", 36, "Fecha pasada")
                        If iRes <> 6 Then
                              txtFecha.setFocus()
                              Exit Function
                        End If
                  End If
                  'Si la fecha es mayor, también se puede usar pero avisamos
                  If dFecha > Date() Then
                        iRes = MsgBox( "La fecha " & Format(dFecha,"dd-mmm-yy") & " es una fecha
futura" & Chr(10) & Chr(10) & "¿Estás seguro de usar esta fecha?", 36, "Fecha futura")
                        If iRes <> 6 Then
                              txtFecha.setFocus()
                              Exit Function
                        End If
                  End If
                  'Si la factura es reservada avisamos
                  If EsReservada Then
                        iRes = MsgBox( "Esta factura " & txtFactura.getText() & " tiene estatus de
RESERVADA" & Chr(10) & Chr(10) & "¿Estás seguro de usarla?", 36, "Guardar factura" )
                        If iRes <> 6 Then
                              Exit Function
                        End If
                  End If

                  'Si todas las pruebas se pasaron regresamos Verdadero, cualquier validación que
necesites
                  'agregala ANTES de esta línea
                  ValidarDatos = True

            Else
                  txtCantidad.setFocus()
                  MsgBox "No hay artículos a facturar, agrega al menos uno", 16, "Sin artículos"
            End If
      Else
            cboClientes.setText( "" )
            MsgBox "El cliente con la clave " & sClave & " no existe", 16,"Facturacion"
            txtClave.setFocus()
      End If
   Else
      cboClientes.setText( "" )
      MsgBox "Escribe una clave de cliente a buscar", 16,"Facturacion"
      txtClave.setFocus()
   End If

End Function
```

Antes ver el código del botón de comando ***Imprimir***, es importante que ya tengas preparado el formato donde se vaciará la información de la factura a imprimir, tener perfectamente determinadas e identificadas las celdas que usaras para cada dato. En mi propuesta usaremos además, algunos controles de formulario que te muestro en la siguiente imagen.



Un cuadro de lista (1) para las leyendas, un cuadro de texto (2) para la cantidad con letra con su etiqueta respectiva, y otros tres cuadros de texto (3) para los totales. Para evitarnos el manipular directamente estos controles, los relacionaremos con celdas de la hoja de calculo y será en estas donde vaciaremos los datos respectivos. Es importante que todos estos controles los ancles a la página, **no** a la celda, con esto logramos que, independientemente del número de líneas que ocupe un concepto a facturar, la posición de estos controles no cambie y siempre se impriman en el lugar correcto. Por supuesto tienes que adaptar estas posiciones al formato físico de tu factura. Agrega además, una línea horizontal (1) a la altura de la ultima fila a imprimir, esta línea estará invisible (2), es importante que este anclada a la celda, de este modo, conforme insertemos conceptos, sé desplazará, la moveremos a la distancia que queramos por código y al estar anclada a la celda, sabremos en donde establecer el área de impresión.



Para ubicarla rápidamente por indice, asegurate de enviarla al fondo de todos los elementos, de este modo sabemos que tendrá como indice el valor cero.

El código del botón de comando ***Imprimir*** es el siguiente.

```
'Para imprimir la factura actual
```

```
Sub cmdImprimir_Clic()
Dim iRes As Integer

   'Validamos que todo este bien
   If ValidarDatos() Then
         iRes = MsgBox( "Todos los datos con correctos, esta factura no esta guardada, puedes
imprimirla pero NO SE GUARDARÁ ningún dato, ¿deseas imprimirla?", 36, "Imprimir factura" )
         If iRes = 6 Then
               'Imprimimos tomando los datos actuales del cuadro de diálogo
               Call ImprimirFactura( "" )
         End If
   End If

End Sub
```

La nueva macro ***ImprimirFactura*** tiene el código siguiente.

```
'Macro para vaciar los datos al formato de impresion
Sub ImprimirFactura( NumeroFactura As String )
Dim sCliente As String
Dim sFecha As String
Dim dFecha As Date
Dim oFecha As Object
Dim sLinea As String
Dim sSubtotal As String
Dim sImpuesto As String
Dim sTotal As String
Dim sEstado As String
Dim oDatosCliente As Object
Dim oDatosFactura As Object
Dim oDatosDetalle As Object
Dim sSQL As String
Dim mLeyendas()
Dim HayLeyendas As Boolean
Dim co1 As Integer
Dim sCelda As String
Dim iCopias As Integer
Dim lTotalDetalle As Integer

   'Borramos las celdas donde vaciamos las leyendas
   Call BorrarDatosRango( ThisComponent, "Impresion", "A51:A54" )
   'Las celdas donde mostramos el detalle
   Call BorrarDatosRango( ThisComponent, "Impresion", "A11:D30" )
   Call BorrarAreasImpresion( ThisComponent, "Impresion" )
   'Copias a imprimir
   iCopias = 1

   'Tomamos los datos del cuadro de diálogo, con esto permitimos hacer pruebas
   'e imprimir las llamadas pre-facturas
   If NumeroFactura = "" Then
         sCliente = txtClave.getText()
         sFecha = Format(txtFecha.getDate())
         dFecha = DateSerial( Val(Left(sFecha,4)), Val(Mid(sFecha,5,2)), Val(Right(sFecha,2)) )
         sSubtotal = txtSubtotal.getText()
         sImpuesto = txtImpuesto.getText()
         sTotal = txtTotal.getText()
         'Si tenemos leyendas
         If lstLeyendas.getItemCount > 0 Then
               mLeyendas = lstLeyendas.getItems()
               HayLeyendas = True
         End If
         'Vaciamos el detalle de la factura
         For co1 = LBound( mDetalle ) To UBound( mDetalle )
               sCelda = "A" & Format(11+co1)
```

```
            Call EscribirEnCelda( ThisComponent, "Impresion", sCelda, 2, mDetalle(co1).Cantidad )
            sCelda = "B" & Format(11+co1)
            Call EscribirEnCelda( ThisComponent, "Impresion", sCelda, 1, mDetalle(co1).Descripcion
)
            sCelda = "C" & Format(11+co1)
            Call EscribirEnCelda( ThisComponent, "Impresion", sCelda, 2, mDetalle(co1).Precio )
            sCelda = "D" & Format(11+co1)
            Call EscribirEnCelda( ThisComponent, "Impresion", sCelda, 2, mDetalle(co1).Importe )
      Next co1
   Else
      'Tomamos los datos de la base de datos
      sSQL = "SELECT * FROM Facturas WHERE Numero=" & NumeroFactura
      oDatosFactura = HacerConsulta( oDatos, sSQL )
      sCliente = oDatosFactura.getString( 2 )
      'getDate nos devuelve una estructura com.sun.star.util.Date
      oFecha = oDatosFactura.getDate( 3 )
      dFecha = DateSerial( oFecha.Year, oFecha.Month, oFecha.Day )
      sSubtotal = Format( oDatosFactura.getDouble( 4 ), "$ #,##0.00" )
      sImpuesto = Format( oDatosFactura.getDouble( 5 ), "$ #,##0.00" )
      sTotal = Format( oDatosFactura.getDouble( 6 ), "$ #,##0.00" )
      sEstado = oDatosFactura.getString( 7 )
      If sEstado = "Elaborada" Then
         'Cambiamos el número de copias predeterminadas a imprimir, generalmente las
         'facturas se imprimen en juegos
         iCopias = COPIAS
      End If
      'Si hay leyendas
      If oDatosFactura.getString( 8 ) <> "" Then
         'Las obtenemos y llenamos la matriz
         mLeyendas = Split( oDatosFactura.getString( 8 ), "|" )
         HayLeyendas = True
      End If
      'El detalle de la factura
      sSQL = "SELECT * FROM Detalle WHERE Factura=" & NumeroFactura
      oDatosDetalle = HacerConsulta( oDatos, sSQL )
      sSQL = "SELECT COUNT(Factura) FROM Detalle WHERE Factura=" & NumeroFactura
      lTotalDetalle = TotalRegistrosConsulta( oDatos, sSQL )
      For co1 = 1 To lTotalDetalle
         sCelda = "A" & Format(10+co1)
         Call EscribirEnCelda( ThisComponent, "Impresion", sCelda, 2,
oDatosDetalle.getFloat(2) )
         sCelda = "B" & Format(10+co1)
         Call EscribirEnCelda( ThisComponent, "Impresion", sCelda, 1, oDatosDetalle.getString(3)
)
         sCelda = "C" & Format(10+co1)
         Call EscribirEnCelda( ThisComponent, "Impresion", sCelda, 2,
oDatosDetalle.getFloat(4) )
         sCelda = "D" & Format(10+co1)
         Call EscribirEnCelda( ThisComponent, "Impresion", sCelda, 2, oDatosDetalle.getDouble(5)
)
         oDatosDetalle.next()
      Next co1

   End If
   'Obtenemos y vaciamos los datos del cliente
   sSQL = "SELECT * FROM Clientes WHERE Id=" & sCliente
   oDatosCliente = HacerConsulta( oDatos, sSQL )
   sLinea = oDatosCliente.getString( 2 )
   Call EscribirEnCelda( ThisComponent, "Impresion", "A1", 1, sLinea )
   sLinea = oDatosCliente.getString( 3 ) & " " & oDatosCliente.getString( 4 ) & ", Col. " &
oDatosCliente.getString( 5 )
   Call EscribirEnCelda( ThisComponent, "Impresion", "A2", 1, sLinea )
   sLinea = oDatosCliente.getString( 6 ) & ", " & oDatosCliente.getString( 8 ) & ", C.P. " &
oDatosCliente.getString( 7 )
   Call EscribirEnCelda( ThisComponent, "Impresion", "A3", 1, sLinea )
   sLinea = "R.F.C.: " & oDatosCliente.getString( 9 )
```

```
    Call EscribirEnCelda( ThisComponent, "Impresion", "A4", 1, sLinea )
    'La fecha de la factura
    sLinea = "México, D.F., a " & Format( dFecha, "dd \d\e mmmm \d\e\l yyyy" )
    Call EscribirEnCelda( ThisComponent, "Impresion", "D5", 1, sLinea )
    'Los importes
    Call EscribirEnCelda( ThisComponent, "Impresion", "D50", 1, sSubtotal )
    Call EscribirEnCelda( ThisComponent, "Impresion", "D51", 1, sImpuesto )
    Call EscribirEnCelda( ThisComponent, "Impresion", "D52", 1, sTotal )
    'La cantidad con letra
    sLinea = Numeros_Letras( Val(Mid(sTotal,3,50)), "peso", False, "", "(", "/100 m.n.)", 1)
    Call EscribirEnCelda( ThisComponent, "Impresion", "A50", 1, sLinea )
    'Si hay leyendas
    If HayLeyendas Then
         Call EscribirEnRango( ThisComponent, "Impresion", "A51", 1, mLeyendas() )
    End If
    'Establecemos el área de impresión e imprimimos
    Call EstablecerAreaImpresion(iCopias, sEstado, NumeroFactura)

End Sub
```

Tenemos algunas macros nuevas, para borra el contenido de un rango.

```
'Macro que borra cualquier valor, texto o formula del rango pasado
Sub BorrarDatosRango( Documento As Object, Hoja As String, Rango As String )
Dim oRango As Object

   oRango = Documento.getSheets.getByName( Hoja ).getCellRangeByname( Rango )
   oRango.clearContents( 21 )

End Sub
```

Para borra todas las áreas de impresión.

```
'Borrar todas las áreas de impresión de la hoja
Sub BorrarAreasImpresion( Documento As Object, Hoja As String )
Dim mAI()

   'Borramos todas las áreas de impresión de la hoja
   Documento.getSheets.getByName( Hoja ).setPrintAreas( mAI() )

End Sub
```

Para escribir en una celda determinada.

```
'Vaciamos el dato pasado en la Celda pasada, hay que establecer el Tipo correcto
Sub EscribirEnCelda( Documento As Object, Hoja As String, Celda As String, Tipo As Byte, Dato )
Dim oCelda As Object

   oCelda = Documento.getSheets.getByName( Hoja ).getCellRangeByname( Celda )
   Select Case Tipo
         Case 1 : oCelda.setString( Dato )
         Case 2 : oCelda.setValue( Dato )
         Case 3 : oCelda.setFormula( Dato )
   End Select

End Sub
```

Para vaciar una matriz unidimensional en un rango, puedes mejorar esta macro de modo que acepte matrices de dos dimensiones y en ves de usar un ciclo usas el método setDataArray para vaciar toda la matriz.

```
'Escribe una matriz en un rango, solo hay que pasarle la celda de inicio
Sub EscribirEnRango( Documento As Object, Hoja As String, Celda As String, Tipo As Byte, Datos() )
Dim oCelda As Object
Dim co1 As Integer
Dim Fil As Long, Col As Long

   oCelda = Documento.getSheets.getByName( Hoja ).getCellRangeByname( Celda )
   'fila y columna de inicio
   Fil = oCelda.getRangeAddress.StartRow
   Col = oCelda.getRangeAddress.StartColumn
   For co1 = LBound( Datos ) To UBound( Datos )
         oCelda = Documento.getSheets.getByName( Hoja ).getCellByPosition( Col, Fil+co1 )
         Select Case Tipo
               Case 1 : oCelda.setString( Datos(co1) )
               Case 2 : oCelda.setValue( Datos(co1) )
               Case 3 : oCelda.setFormula( Datos(co1) )
         End Select
   Next co1

End Sub
```

La macro para establecer el área de impresión y por fin, imprimir. Asumimos que se envía a la impresora predeterminada, si usas otra impresora, usa las propiedades aprendidas en el capítulo de impresión.

```
'Establece el área de impresión e imprime
Sub EstablecerAreaImpresion( NumCopias As Integer, Estado As String, NumeroFactura As String )
Dim mAI(0) As New com.sun.star.table.CellRangeAddress
Dim oDir As New com.sun.star.table.CellAddress
Dim oDoc As Object
Dim oHoja As Object
Dim oPaginaDibujo As Object
Dim oLinea As Object
Dim oAncla As object
Dim oPos As New com.sun.star.awt.Point
Dim mOpc(0) As New com.sun.star.beans.PropertyValue
Dim oGlobal As Object
Dim sSQL As String

   'Servicio global de configuración de Calc
   oGlobal = createUnoService( "com.sun.star.sheet.GlobalSheetSettings" )
   oDoc = ThisComponent
   oHoja = oDoc.getSheets.getByName( "Impresion" )
   oPaginaDibujo = oHoja.getDrawPage()

   'Esta línea oculta nos auxilia para obtener el área de impresión
   oLinea = oPaginaDibujo.getByINdex( 0 )
   oPos.X = 0
   'La posicionamos a 18 cm de la parte superior que es el área de impresión disponible
   oPos.Y = 18000
   oLinea.setPosition( oPos )
   'Obtenemos la dirección de la celda donde haya quedado
   oDir = oLinea.Anchor.getCellAddress

   'Construimos el área de impresión correctamente
   'Rango A1:D ?
   mAI(0).Sheet = oDir.Sheet
   mAI(0).StartColumn = 0
   mAI(0).StartRow = 0
   mAI(0).EndColumn = 3
   mAI(0).EndRow = oDir.Row

   'Agregamos el área de impresión
```

```
   oHoja.setPrintAreas( mAI() )
   'Preguntamos cuantas copias se imprimirán
   NumCopias = Val( InputBox( "¿Cuantas copias se imprimirán?" & Chr(10) & Chr(10) & "Asegurate de
que la impresora este lista para imprimir", "Imprimir factura", NumCopias ) )
   'Tienes que ser mayor a cero
   If NumCopias > 0 Then
         'El número de copias
         mOpc(0).Name = "CopyCount"
         mOpc(0).Value = NumCopias
         'Enviamos a imprimir, dado que imprimiremos una hoja diferente a la seleccionada
         'es importante activar esta propiedad, si no, no imprimirá nada
         oGlobal.PrintAllSheets = True
         'Imprimimos
         oDoc.Print( mOpc() )
         'Desactivamos la propiedad
         oGlobal.PrintAllSheets = False
         'Actualizamos el estado de la factura, solo si su estado anterior es Elaborada
         If Estado = "Elaborada" Then
               sSQL = "UPDATE ""Facturas"" SET ""Estado""='Impresa' WHERE ""Numero""=" & NumeroFactura
               Call ActualizarDatos( oDatos, sSQL )
         End If
   Else
         MsgBox "Proceso cancelado, no se imprimió la factura", 0, "Factura Libre"
   End If

End Sub
```

Solo nos resta hacer el código del botón de comando ***Guardar e Imprimir*** que es el siguiente, esta macro usa las mismas subrutinas anteriores. Como le pasamos el número de factura a la macro ***ImprimirFactura***, toma los datos de la factura de la base de datos.

```
'Guarda e imprime la factura
Sub cmdGuardarImprimir_Clic()
Dim sFactura As String
Dim iRes As Integer

   'Guardamos el número de factura actual
   sFactura = txtFactura.getText()
   'Intentamos guardar
   Call cmdGuardar_Clic()
   'Si cambia el número de factura esta se guardó correctamente
   If sFactura <> txtFactura.getText() Then
         iRes = MsgBox( "La factura " & sFactura & " se guardó correctamente, puedes imprimirla ahora,
¿deseas imprimirla?", 36, "Imprimir factura" )
         If iRes = 6 Then
               'Imprimimos la factura guardada
               Call ImprimirFactura( sFactura )
         End If
   End If

End Sub
```

Mira que bien esta quedando.

Tal vez pensaras que se me olvido el código del botón ***Ver Facturas***, pero no, no se me olvido si no que es tu tarea escribirlo, la idea de este botón es mostrar en un nuevo cuadro de diálogo, las facturas guardadas, darle la posibilidad al usuario de filtrar por varias formas, de ver el detalle de la factura que quiera y de cambiar el estatus de las mismas, este cuadro de diálogo tiene que ser capaz también, de imprimir o reimprimir la factura que sea, así como de mostrar los reportes necesarios de la facturación. No me decepciones e intenta resolverlo antes de ver mi propuesta en los archivos de código que acompañan a este libro.

Posibles mejoras que te quedan de tarea:

- Si la base de datos no esta registrada, pero el archivo ODB esta en el mismo directorio del archivo usado como interfaz, podría registrarse automáticamente o preguntarle al usuario si quiere usar este archivo como base de datos fuente.
- Evitar que el usuario cierre los cuadros de diálogo con el botón de cierre, obligarlo a usar siempre el botón Salir.
- Puedes cambiar los controles de texto por controles de formato donde se requiera, por ejemplo: el C.P. Y el RFC.
- Puedes permitir al usuario moverse entre los campos con la tecla ***Enter***.
- ¿Se te ocurre alguna otra?

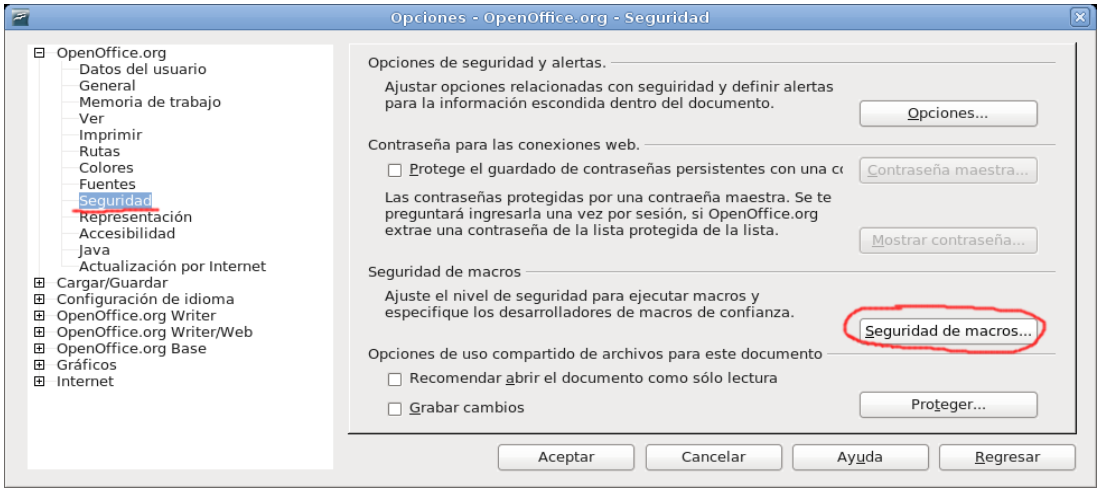
Solo me resta decirte; para no cometer errores, tienes que convertirte en un experimentado programador, para convertirte en un programador experimentado, tienes que cometer muchos errores...

¡¡Feliz programación!!

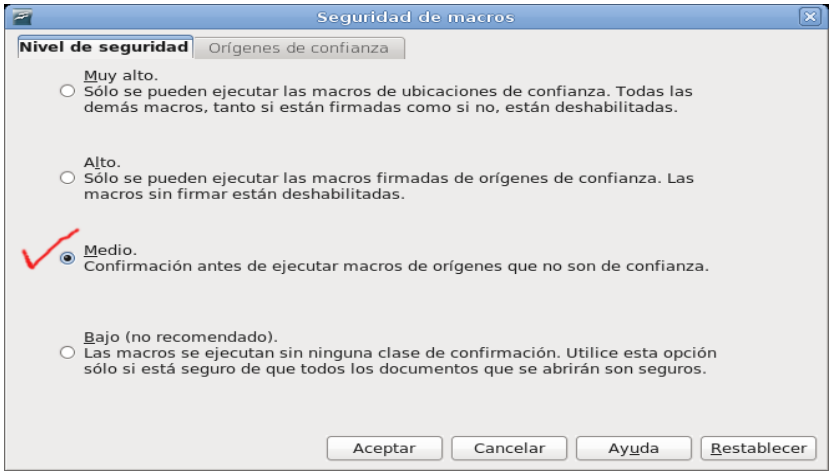
11 Apéndices

11.1 Seguridad en macros

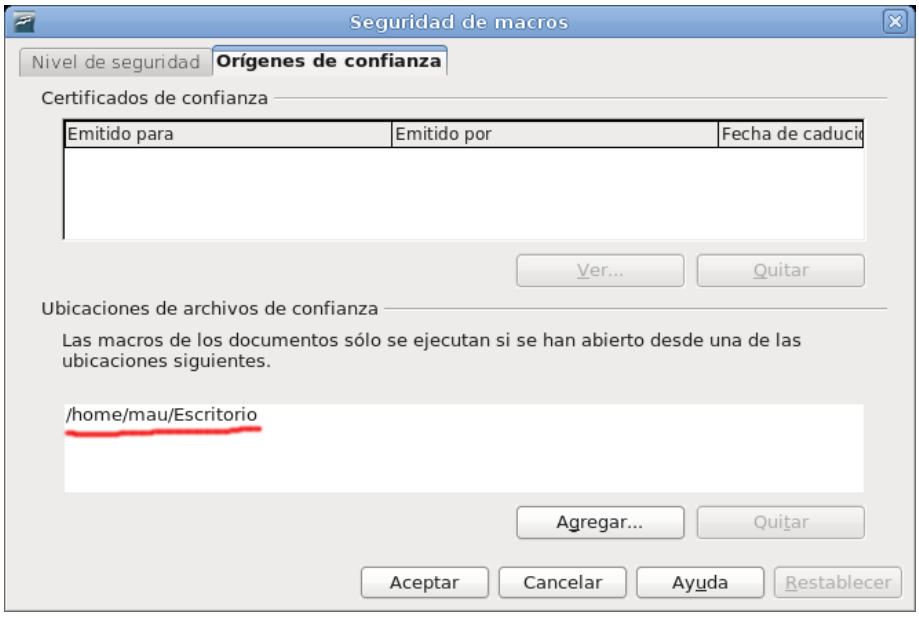
Como ya lo habrás notado, el limite para lo que se puede hacer con macros es solo tu conocimiento e imaginación, por ello, no tomes como un tema menor el de la seguridad, es decir, asegúrate de que las macros que estás activando y ejecutando sean de una fuente confiable. Para establecer el nivel de seguridad de las macros, ve al menú Herramientas | Opciones... dentro de la sección OpenOffice.org, selecciona la rama Seguridad y dentro de las opciones mostradas, da un clic en el botón de comando Seguridad de macro...



El nivel de confianza es bastante explicito y creo que no requiere aclaraciones, queda a tu criterio establecer el nivel adecuado para tu área y entorno de trabajo, para los fines didácticos de estos apuntes, el nivel ***Medio*** es más que suficiente.



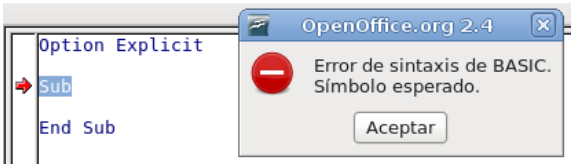
Si no quieres que te muestre el cuadro de dialogo de confirmación cada vez que abres tu archivo de trabajo, puedes agregar su ubicación como un origen de confianza, en el mismo cuadro de dialogo anterior pero en la ficha ***Orígenes de Confianza***, puedes agregar tantas rutas como quieras, desde las cuales, se abrirán los archivos con las macros activadas de forma predeterminada sin pedirte confirmación. Cuidado con esto, asegurate de establecer solo rutas de verdadera confianza, como dicen en mi pueblo -sobre advertencia no hay engaño-.



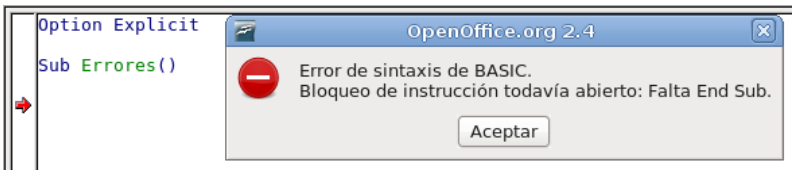
11.2 Errores más comunes en tiempo de diseño

Como ya explicamos en el tema 4.10 Control de errores, este tipo de errores solo se producen mientras programamos y en ese momento hay que corregirlos, esta es una lista incompleta de los más comunes, para estos ejemplos, solo te mostrare la imagen del código con el error, espero que los títulos sean lo suficientemente claro para tu consulta:

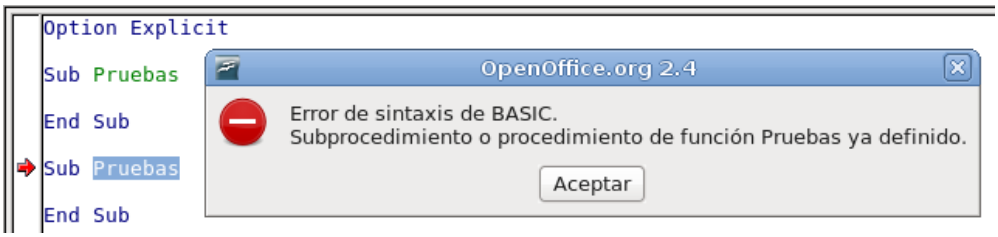
- No puede haber una macro sin nombre



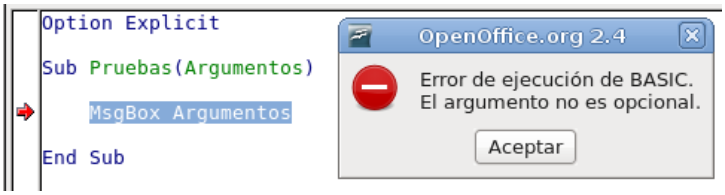
- Toda macro debe cerrarse con End Sub



- No puede haber dos macros o funciones con el mismo nombre

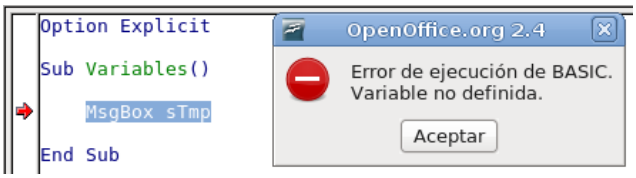


- No puedes ejecutar directamente una macro que requiera argumentos

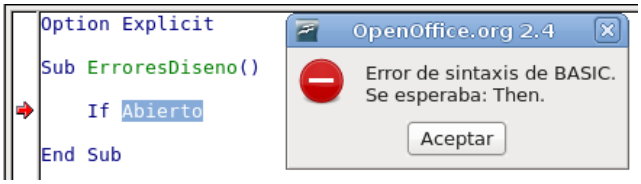


- Por supuesto, tampoco llamarla sin los argumentos correctos:

- Si usas Option Explicit (recomendado), tienes que declarar todas tus variables.

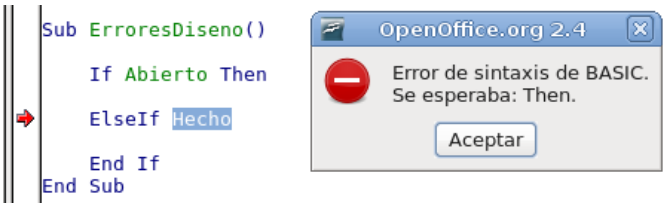


- Todas las estructuras deben ir completas, aquí falta Then al If

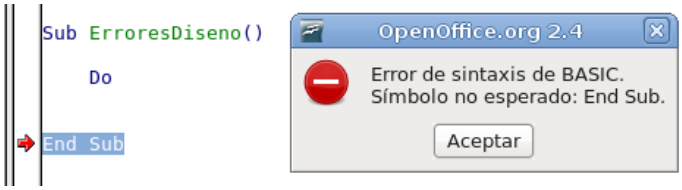


- Por supuesto también debe llevar su correspondiente End If

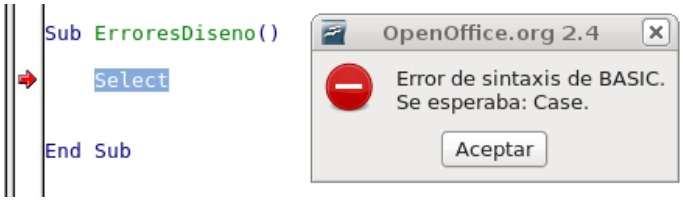
- Si usas ElseIf, recuerda que también termina en Then



- La estructura Do, siempre debe cerrarse con Loop, te cuidado de establecer siempre la condición de terminación del bucle, si no lo haces no te dará un error y puedes quedar dentro de un bucle infinito

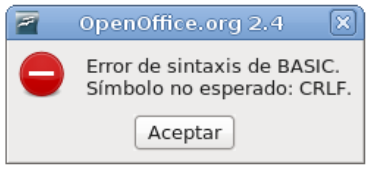


- La estructura Select debe terminar en Case



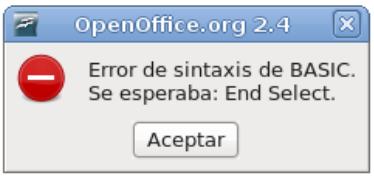
- Y no te olvides de establecer la variable a evaluar

```
Sub ErroresDiseno()

    Select Case
        Case 1


End Sub
```



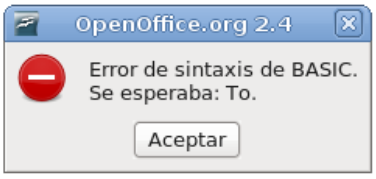
- Y cerrar con su respectivo End Select

```
Sub ErroresDiseno()

    Select Case col
        Case 1


End Sub
```



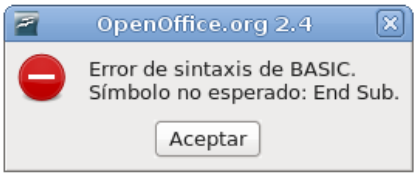
- Los ciclos For, deben estar completa, con su correspondiente To

```
Sub ErroresDiseno()

    For col = 1

End Sub
```



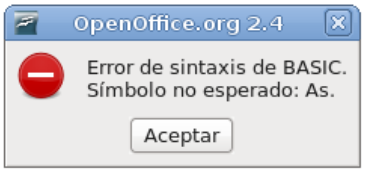
- Y su respectivo Next

```
Sub ErroresDiseno()

    For col = 1 To 5


End Sub
```



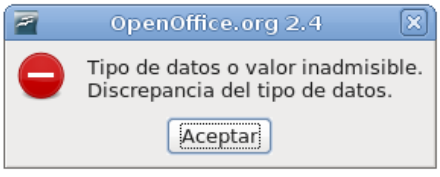
- Se requiere el uso de Dim para declarar variables

```
Sub ErroresDiseno()
col As Integer

End Sub
```

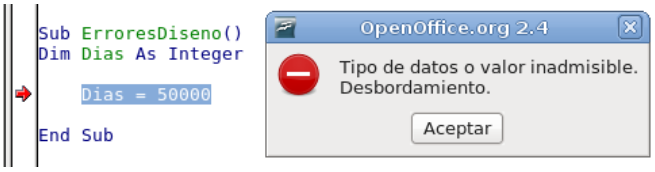


- Algunos datos no soportan la conversión de datos implícita, es decir, tienes que establecer correctamente el tipo de información que guardas en estas variables

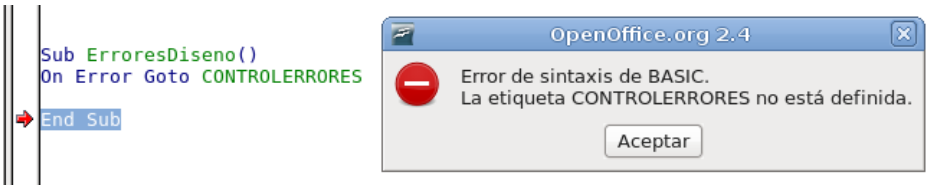
```
Sub ErroresDiseno()
Dim Fecha As Date

    Fecha = "hola"

End Sub
```



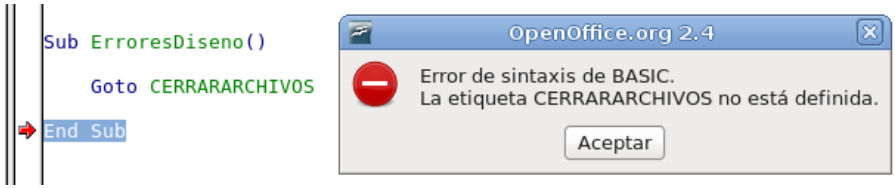
- Este es muy común, sobrepasar los limites que acepta un tipo de dato



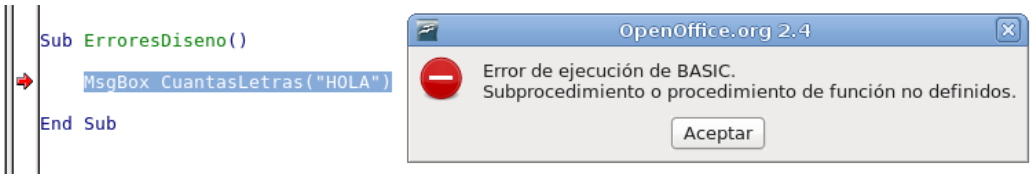
- Debes declarar la etiqueta correspondiente en el controlador de errores On Error Goto



- Y en la instrucción Goto



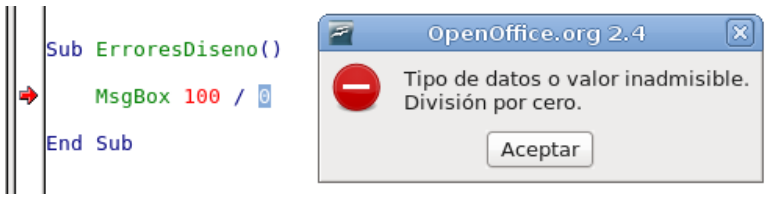
- La macro o función llamada debe existir



- No puedes hacer referencia al índice de una matriz fuera de sus rangos declarados

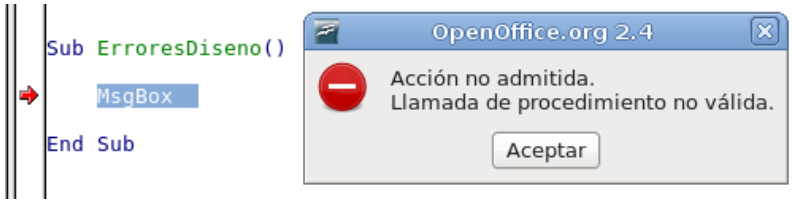
- No puedes, dentro de la misma macro, definir dos veces la misma variable

- No puedes dividir por cero



- Debes de usar la instrucción Exit, con su correspondiente predicado Sub, Do, For o Function y dentro de la estructura correspondiente correcta

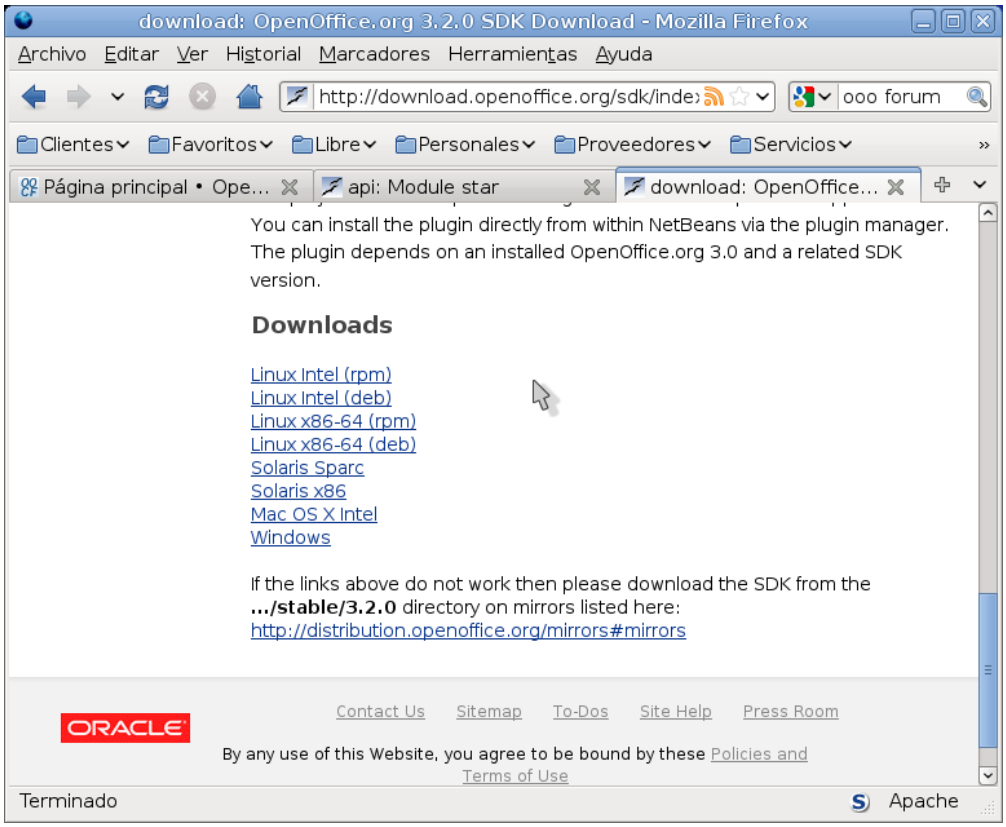
- Otra muy común, todas las instrucciones y funciones de OOo Basic, tienes que llamarlas con sus correctos argumentos



11.3 Instalando SDK

Lo primero que necesitamos es descargarlo desde la página de OpenOffice.org

http://download.openoffice.org/sdk/index.html



Tienes que seleccionar el archivo correcto para tu sistema operativo y arquitectura.

Para sistemas GNU/Linux, primero tienes que descomprimir el archivo, ve a la carpeta donde descargaste el archivo y desde una terminal escribes, según corresponda a tu sistema y arquitectura.

```
sh Ooo-SDK_3.2.0_LinuxIntel_install_en-US_deb.sh

sh Ooo-SDK_3.2.0_LinuxIntel_install_en-US_rmp.sh

sh Ooo-SDK_3.2.0_LinuxX86-64_install_en-US_deb.sh

sh Ooo-SDK_3.2.0_LinuxX86-64_install_en-US_rmp.sh
```

Te solicitará una carpeta para descomprimir el contenido, proporciona una accesible donde tengas derechos de escritura. Si todo salio bien te indicará que todos los archivos se extrajeron correctamente, tienes que indicarle una carpeta nueva, el script la creara automáticamente, si le das la ruta de una carpeta existente, fallará la extracción.

```
Select the directory in which to save the unpacked files. [/var/tmp/unpack_openofficeorg]
/home/usuario/ooosdk32/
```

```
File is being checked for errors ...
Unpacking ...
All files have been successfully unpacked.
```

Muevete a la nueva carpeta recién creada, veras una subcarpeta llamada DEBS (para sistemas basados en Debian y derivados), en la terminal escribes:

```
sudo dpkg -i ooobasis3.2-sdk_3.2.0-12_amd64.deb
```

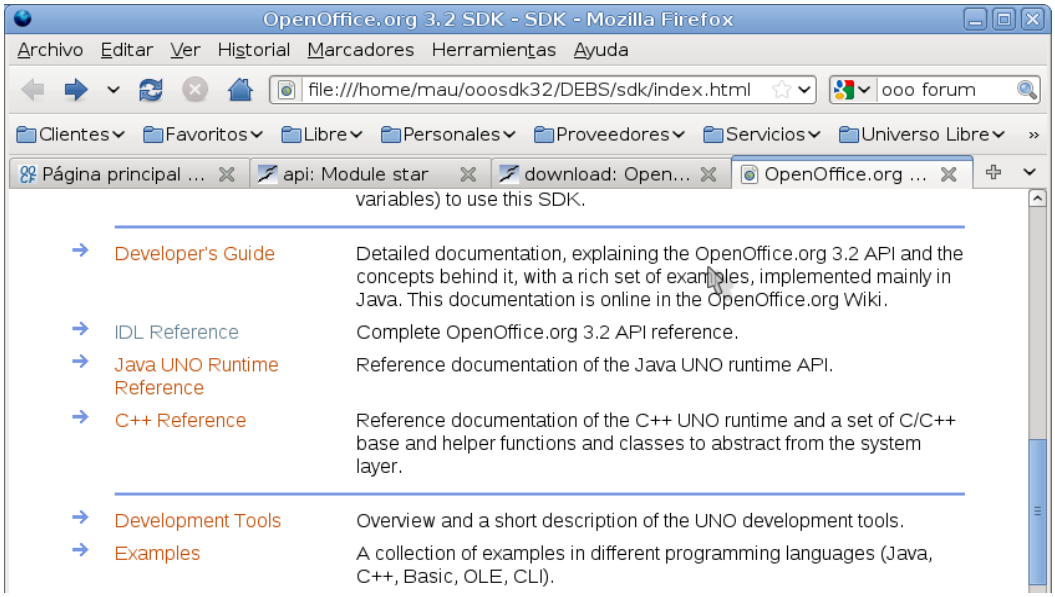
Para sistema basados en RMPS (Fedora y derivados), escribes:

```
rpm -i ooobasis3.2-sdk-3.2.0-9483.x86_64.rpm
```

Si todo salio bien, ahora puedes abrir tu navegador favorito y abrir el archivo

file:///opt/openoffice/basis3.2/sdk/index.html

Donde, entre otras herramientas y ejemplos, tendrás la referencia completa del API de OpenOffice.org



Para sistemas Windows solo tienes que instalar el ejecutable correspondiente, la ruta predeterminada para los archivos ya instalados es:

C:\Program Files\OpenOffice.org_3.2_SDK\sdk\index.html

11.4 Mostrar información de un objeto en un archivo de Calc

```
Sub ObtenerDatos( Objeto As Object )
Dim sRuta As String
Dim oDoc As Object
Dim mArg()
Dim a1() As String, a2() As String
Dim co1 As Integer, sTmp As String

   sRuta = "private:factory/scalc"
   oDoc = StarDesktop.LoadComponentFromURL( sRuta,"_default",0,mArg )

   a1 = Split( Objeto.DBG_Properties, ":" )
   a2 = Split( Trim(a1(1)), ";" )
   oDoc.Sheets().getByIndex(0).getCellByPosition( 0, 0 ).setString( a1(0) )
   For co1 = LBound( a2 ) To UBound( a2 )
         sTmp = Trim(a2(co1))
         If Left( sTmp, 1) = Chr(10) Then sTmp = Right( sTmp, Len(sTmp)-1 )
         oDoc.Sheets().getByIndex(0).getCellByPosition( 0, co1+1 ).setString( sTmp )
   Next co1
   If Right( sTmp, 1) = Chr(10) Then sTmp = Left( sTmp, Len(sTmp)-1 )
   oDoc.Sheets().getByIndex(0).getCellByPosition( 0, co1 ).setString( sTmp )

   a1 = Split( Objeto.DBG_Methods, ":" )
   a2 = Split( Trim(a1(1)), ";" )
   oDoc.Sheets().getByIndex(0).getCellByPosition( 1, 0 ).setString( a1(0)
   For co1 = LBound( a2 ) To UBound( a2 )
         sTmp = Trim(a2(co1))
         If Left( sTmp, 1) = Chr(10) Then sTmp = Right( sTmp, Len(sTmp)-1 )
         oDoc.Sheets().getByIndex(0).getCellByPosition( 1, co1+1 ).setString( sTmp )
   Next co1
   If Right( sTmp, 1) = Chr(10) Then sTmp = Left( sTmp, Len(sTmp)-1 )
   oDoc.Sheets().getByIndex(0).getCellByPosition( 1, co1 ).setString( sTmp )

   a1 = Split( Objeto.DBG_SupportedInterfaces, ":" )
   a2 = Split( Trim(a1(1)), Chr(10) )
   oDoc.Sheets().getByIndex(0).getCellByPosition( 2, 0 ).setString( a1(0) )
   For co1 = LBound( a2 ) To UBound( a2 )
         sTmp = Trim(a2(co1))
         If Left( sTmp, 1) = Chr(10) Then sTmp = Right( sTmp, Len(sTmp)-1 )
         oDoc.Sheets().getByIndex(0).getCellByPosition( 2, co1 ).setString( sTmp )
   Next co1

End Sub
```

Asegurate de pasarle como parámetro una variable de objeto, si el objeto no implementa métodos, la macro te dará un error y se detendrá, pero te creará el archivo con al menos las propiedades que implementa.

```
Sub Pruebas
Dim obj As Object

   obj = ThisComponent

   Call ObtenerDatos ( obj )

End Sub
```

11.5 Formulas de Calc español-ingles

Nº	Español	Ingles
1	360DIAS	DAYS360
2	ABS	ABS
3	ACOS	ACOS
4	ACOSH	ACOSH
5	ACOT	ACOT
6	ACOTH	ACOTH
7	ACTUAL	CURRENT
8	AHORA	NOW
9	ALEATORIO	RAND
10	ALEATORIO.ENTRE	com.sun.star.sheet.addin.Analysis.getRandbetween
11	AMORTIZ.LIN	com.sun.star.sheet.addin.Analysis.getAmorlinc
12	AMORTIZ.PROGRE	com.sun.star.sheet.addin.Analysis.getAmordegrc
13	AÑO	YEAR
14	AÑOS	com.sun.star.sheet.addin.DateFunctions.getDiffYears
15	ÁRABE	ARABIC
16	ÁREAS	AREAS
17	ASC	ASC
18	ASENO	ASIN
19	ASENOH	ASINH
20	ATAN	ATAN
21	ATAN2	ATAN2
22	ATANH	ATANH
23	B	B
24	BAHTTEXT	BAHTTEXT
25	BASE	BASE
26	BDCONTAR	DCOUNT
27	BDCONTARA	DCOUNTA
28	BDDESVEST	DSTDEV
29	BDDESVESTP	DSTDEVP
30	BDEXTRAER	DGET
31	BDMÁX	DMAX
32	BDMÍN	DMIN
33	BDPRODUCTO	DPRODUCT
34	BDPROMEDIO	DAVERAGE
35	BDSUMA	DSUM
36	BDVAR	DVAR
37	BDVARP	DVARP
38	BESSELI	com.sun.star.sheet.addin.Analysis.getBesseli
39	BESSELJ	com.sun.star.sheet.addin.Analysis.getBesselj

Nº	Español	Ingles
40	BESSELK	com.sun.star.sheet.addin.Analysis.getBesselk
41	BESSELY	com.sun.star.sheet.addin.Analysis.getBessely
42	BIN.A.DEC	com.sun.star.sheet.addin.Analysis.getBin2Dec
43	BIN.A.HEX	com.sun.star.sheet.addin.Analysis.getBin2Hex
44	BIN.A.OCT	com.sun.star.sheet.addin.Analysis.getBin2Oct
45	BINOM.CRIT	CRITBINOM
46	BUSCAR	LOOKUP
47	BUSCARH	HLOOKUP
48	BUSCARV	VLOOKUP
49	CANTIDAD.RECIBIDA	com.sun.star.sheet.addin.Analysis.getReceived
50	CARÁCTER	CHAR
51	CELDA	CELL
52	CHISQDIST	CHISQDIST
53	CHISQINV	CHISQINV
54	COCIENTE	com.sun.star.sheet.addin.Analysis.getQuotient
55	CÓDIGO	CODE
56	COEF.DE.CORREL	CORREL
57	COEFICIENTE.ASIMETRIA	SKEW
58	COEFICIENTE.R2	RSQ
59	COINCIDIR	MATCH
60	COLUMNA	COLUMN
61	COLUMNAS	COLUMNS
62	COMBINAT	COMBIN
63	COMBINATA	COMBINA
64	COMPLEJO	com.sun.star.sheet.addin.Analysis.getComplex
65	CONCATENAR	CONCATENATE
66	CONFIANZA	CONFIDENCE
67	CONTAR	COUNT
68	CONTAR.BLANCO	COUNTBLANK
69	CONTAR.SI	COUNTIF
70	CONTARA	COUNTA
71	CONVERTIR	CONVERT
72	CONVERTIR_ADD	com.sun.star.sheet.addin.Analysis.getConvert
73	COS	COS
74	COSH	COSH
75	COT	COT
76	COTH	COTH
77	COVAR	COVAR
78	CRECIMIENTO	GROWTH
79	CUARTIL	QUARTILE
80	CUPON.DIAS	com.sun.star.sheet.addin.Analysis.getCoupdays
81	CUPON.DIAS.L1	com.sun.star.sheet.addin.Analysis.getCoupdaybs
82	CUPON.DIAS.L2	com.sun.star.sheet.addin.Analysis.getCoupdaysnc
83	CUPON.FECHA.L1	com.sun.star.sheet.addin.Analysis.getCouppcd

Nº	Español	Ingles
84	CUPON.FECHA.L2	com.sun.star.sheet.addin.Analysis.getCoupncd
85	CUPON.NUM	com.sun.star.sheet.addin.Analysis.getCoupnum
86	CURTOSIS	KURT
87	DB	DB
88	DDB	DDB
89	DDE	DDE
90	DEC.A.BIN	com.sun.star.sheet.addin.Analysis.getDec2Bin
91	DEC.A.HEX	com.sun.star.sheet.addin.Analysis.getDec2Hex
92	DEC.A.OCT	com.sun.star.sheet.addin.Analysis.getDec2Oct
93	DECIMAL	DECIMAL
94	DELTA	com.sun.star.sheet.addin.Analysis.getDelta
95	DERECHA	RIGHT
96	DESREF	OFFSET
97	DESVEST	STDEV
98	DESVESTA	STDEVA
99	DESVESTP	STDEVP
100	DESVESTPA	STDEVPA
101	DESVIA2	DEVSQ
102	DESVPROM	AVEDEV
103	DÍA	DAY
104	DIA.LAB	com.sun.star.sheet.addin.Analysis.getWorkday
105	DIAS	DAYS
106	DIAS.LAB	com.sun.star.sheet.addin.Analysis.getNetworkdays
107	DÍASEM	WEEKDAY
108	DÍASENAÑO	com.sun.star.sheet.addin.DateFunctions.getDaysInYear
109	DÍASENMES	com.sun.star.sheet.addin.DateFunctions.getDaysInMonth
110	DIRECCIÓN	ADDRESS
111	DIST.WEIBULL	WEIBULL
112	DISTR.BETA	BETADIST
113	DISTR.BETA.INV	BETAINV
114	DISTR.BINOM	BINOMDIST
115	DISTR.CHI	CHIDIST
116	DISTR.EXP	EXPONDIST
117	DISTR.F	FDIST
118	DISTR.F.INV	FINV
119	DISTR.GAMMA	GAMMADIST
120	DISTR.GAMMA.INV	GAMMAINV
121	DISTR.HIPERGEOM	HYPGEOMDIST
122	DISTR.LOG.NORM	LOGNORMDIST(;0;
123	DISTR.NORM	NORMDIST
124	DISTR.NORM.ESTAND	NORMSDIST
125	DISTR.NORM.ESTAND.INV	NORMSINV
126	DISTR.NORM.INV	NORMINV
127	DISTR.T	TDIST

Nº	Español	Ingles
128	DISTR.T.INV	TINV
129	DOMINGOPASCUA	EASTERSUNDAY
130	DURACIÓN	DURATION
131	DURACION_ADD	com.sun.star.sheet.addin.Analysis.getDuration
132	DURACION.MODIF	com.sun.star.sheet.addin.Analysis.getMduration
133	DVS	VDB
134	ELEGIR	CHOOSE
135	ENCONTRAR	FIND
136	ERROR.TÍPICO.XY	STEYX
137	ES.IMPAR_ADD	com.sun.star.sheet.addin.Analysis.getIsodd
138	ESAÑOBISIESTO	com.sun.star.sheet.addin.DateFunctions.getIsLeapYear
139	ESBLANCO	ISBLANK
140	ESERR	ISERR
141	ESERROR	ISERROR
142	ESFÓRMULA	ISFORMULA
143	ESIMPAR	ISODD
144	ESLOGICO	ISLOGICAL
145	ESNOD	ISNA
146	ESNOTEXTO	ISNONTEXT
147	ESNÚMERO	ISNUMBER
148	ESPAR	ISEVEN
149	ESPAR_ADD	com.sun.star.sheet.addin.Analysis.getIseven
150	ESREF	ISREF
151	ESTEXTO	ISTEXT
152	ESTILO	STYLE
153	ESTIMACIÓN.LINEAL	LINEST
154	ESTIMACIÓN.LOGARÍTMICA	LOGEST
155	EUROCONVERT	EUROCONVERT
156	EXP	EXP
157	FACT	FACT
158	FACT.DOBLE	com.sun.star.sheet.addin.Analysis.getFactdouble
159	FALSO	FALSE
160	FECHA	DATE
161	FECHA.MES	com.sun.star.sheet.addin.Analysis.getEdate
162	FECHANÚMERO	DATEVALUE
163	FIJO	FIXED
164	FILA	ROW
165	FILAS	ROWS
166	FIN.MES	com.sun.star.sheet.addin.Analysis.getEomonth
167	FISHER	FISHER
168	FÓRMULA	FORMULA
169	FRAC.AÑO	com.sun.star.sheet.addin.Analysis.getYearfrac
170	FRECUENCIA	FREQUENCY
171	FUN.ERROR	com.sun.star.sheet.addin.Analysis.getErf

Nº	Español	Ingles
172	FUN.ERROR.COMPL	com.sun.star.sheet.addin.Analysis.getErfc
173	GAMMA	GAMMA
174	GAMMA.LN	GAMMALN
175	GAUSS	GAUSS
176	GETPIVOTDATA	GETPIVOTDATA
177	GRADOS	DEGREES
178	HALLAR	SEARCH
179	HEX.A.BIN	com.sun.star.sheet.addin.Analysis.getHex2Bin
180	HEX.A.DEC	com.sun.star.sheet.addin.Analysis.getHex2Dec
181	HEX.A.OCT	com.sun.star.sheet.addin.Analysis.getHex2Oct
182	HIPERVÍNCULO	HYPERLINK
183	HOJA	SHEET
184	HOJAS	SHEETS
185	HORA	HOUR
186	HORANÚMERO	TIMEVALUE
187	HOY	TODAY
188	IGUAL	EXACT
189	IM.ABS	com.sun.star.sheet.addin.Analysis.getImabs
190	IM.ANGULO	com.sun.star.sheet.addin.Analysis.getImargument
191	IM.CONJUGADA	com.sun.star.sheet.addin.Analysis.getImconjugate
192	IM.COS	com.sun.star.sheet.addin.Analysis.getImcos
193	IM.DIV	com.sun.star.sheet.addin.Analysis.getImdiv
194	IM.EXP	com.sun.star.sheet.addin.Analysis.getImexp
195	IM.LN	com.sun.star.sheet.addin.Analysis.getImln
196	IM.LOG10	com.sun.star.sheet.addin.Analysis.getImlog10
197	IM.LOG2	com.sun.star.sheet.addin.Analysis.getImlog2
198	IM.POT	com.sun.star.sheet.addin.Analysis.getImpower
199	IM.PRODUCT	com.sun.star.sheet.addin.Analysis.getImproduct
200	IM.RAIZ2	com.sun.star.sheet.addin.Analysis.getImsqrt
201	IM.REAL	com.sun.star.sheet.addin.Analysis.getImreal
202	IM.SENO	com.sun.star.sheet.addin.Analysis.getImsin
203	IM.SUM	com.sun.star.sheet.addin.Analysis.getImsum
204	IM.SUSTR	com.sun.star.sheet.addin.Analysis.getImsub
205	IMAGINARIO	com.sun.star.sheet.addin.Analysis.getImaginary
206	ÍNDICE	INDEX
207	INDIRECTO	INDIRECT
208	INFO	INFO
209	INT	INT
210	INT.ACUM	com.sun.star.sheet.addin.Analysis.getAccrint
211	INT.ACUM.V	com.sun.star.sheet.addin.Analysis.getAccrintm
212	INT.EFECTIVO	EFFECTIVE
213	INT.EFECTIVO_ADD	com.sun.star.sheet.addin.Analysis.getEffect
214	INT.PAGO.DIR	ISPMT
215	INT.RENDIMIENTO	ZGZ

Nº	Español	Ingles
216	INTERSECCIÓN.EJE	INTERCEPT
217	INV.LOG	LOGINV
218	IZQUIERDA	LEFT
219	JERARQUÍA	RANK
220	JIS	JIS
221	K.ESIMO.MAYOR	LARGE
222	K.ESIMO.MENOR	SMALL
223	LARGO	LEN
224	LETRA.DE.TES.EQV.A.BONO	com.sun.star.sheet.addin.Analysis.getTbilleq
225	LETRA.DE.TES.PRECIO	com.sun.star.sheet.addin.Analysis.getTbillprice
226	LETRA.DE.TES.RENDTO	com.sun.star.sheet.addin.Analysis.getTbillyield
227	LIMPIAR	CLEAN
228	LN	LN
229	LOG	LOG(;1
230	LOG10	LOG10
231	M.C.D	GCD
232	M.C.D_ADD	com.sun.star.sheet.addin.Analysis.getGcd
233	M.C.M	LCM
234	M.C.M_ADD	com.sun.star.sheet.addin.Analysis.getLcm
235	MÁX	MAX
236	MÁXA	MAXA
237	MAYOR.O.IGUAL	com.sun.star.sheet.addin.Analysis.getGestep
238	MAYÚSC	UPPER
239	MDETERM	MDETERM
240	MEDIA.ACOTADA	TRIMMEAN
241	MEDIA.ARMO	HARMEAN
242	MEDIA.GEOM	GEOMEAN
243	MEDIANA	MEDIAN
244	MES	MONTH
245	MESES	com.sun.star.sheet.addin.DateFunctions.getDiffMonths
246	MID	MID
247	MÍN	MIN
248	MÍNA	MINA
249	MINÚSC	LOWER
250	MINUTO	MINUTE
251	MINVERSA	MINVERSE
252	MIRR	MIRR
253	MMULT	MMULT
254	MODO	MODE
255	MONEDA	DOLLAR
256	MONEDA.DEC	com.sun.star.sheet.addin.Analysis.getDollarde
257	MONEDA.FRAC	com.sun.star.sheet.addin.Analysis.getDollarfr
258	MULTINOMIAL	com.sun.star.sheet.addin.Analysis.getMultinomial
259	MÚLTIPLO.INFERIOR	FLOOR

Nº	Español	Ingles
260	MÚLTIPLO.SUPERIOR	CEILING
261	MUNITARIA	MUNIT
262	N	N
263	NEGBINOMDIST	NEGBINOMDIST
264	NO	NOT
265	NOD	NA
266	NOMPROPIO	PROPER
267	NORMALIZACIÓN	STANDARDIZE
268	NPER	NPER
269	NUM.DE.SEMANA_ADD	com.sun.star.sheet.addin.Analysis.getWeeknum
270	O	OR
271	OCT.A.BIN	com.sun.star.sheet.addin.Analysis.getOct2Bin
272	OCT.A.DEC	com.sun.star.sheet.addin.Analysis.getOct2Dec
273	OCT.A.HEX	com.sun.star.sheet.addin.Analysis.getOct2Hex
274	PAGO	PMT
275	PAGO.INT.ENTRE	CUMIPMT
276	PAGO.INT.ENTRE_ADD	com.sun.star.sheet.addin.Analysis.getCumipmt
277	PAGO.PRINC.ENTRE	CUMPRINC
278	PAGO.PRINC.ENTRE_ADD	com.sun.star.sheet.addin.Analysis.getCumprinc
279	PAGOINT	IPMT
280	PAGOPRIN	PPMT
281	PEARSON	PEARSON
282	PENDIENTE	SLOPE
283	PERCENTIL	PERCENTILE
284	PERMUTACIONES	PERMUT
285	PERMUTACIONESA	PERMUTATIONA
286	PHI	PHI
287	PI	PI
288	POISSON	POISSON
289	POTENCIA	POWER
290	PRECIO	com.sun.star.sheet.addin.Analysis.getPrice
291	PRECIO.DESCUENTO	com.sun.star.sheet.addin.Analysis.getPricedisc
292	PRECIO.PER.IRREGULAR.1	com.sun.star.sheet.addin.Analysis.getOddfprice
293	PRECIO.PER.IRREGULAR.2	com.sun.star.sheet.addin.Analysis.getOddlprice
294	PRECIO.VENCIMIENTO	com.sun.star.sheet.addin.Analysis.getPricemat
295	PROBABILIDAD	PROB
296	PRODUCTO	PRODUCT
297	PROMEDIO	AVERAGE
298	PROMEDIOA	AVERAGEA
299	PRONÓSTICO	FORECAST
300	PRUEBA.CHI	CHITEST
301	PRUEBA.CHI.INV	CHIINV
302	PRUEBA.F	FTEST
303	PRUEBA.FISHER.INV	FISHERINV

Nº	Español	Ingles
304	PRUEBA.T	TTEST
305	PRUEBA.Z	ZTEST
306	RADIANES	RADIANS
307	RAIZ	SQRT
308	RAIZ2PI	com.sun.star.sheet.addin.Analysis.getSqrtpi
309	RANGO.PERCENTIL	PERCENTRANK
310	REDOND.MULT	com.sun.star.sheet.addin.Analysis.getMround
311	REDONDEA.IMPAR	ODD
312	REDONDEA.PAR	EVEN
313	REDONDEAR	ROUND
314	REDONDEAR.MAS	ROUNDUP
315	REDONDEAR.MENOS	ROUNDDOWN
316	REDUCIR	TRIM
317	REEMPLAZAR	REPLACE
318	RENDTO	com.sun.star.sheet.addin.Analysis.getYield
319	RENDTO.DESC	com.sun.star.sheet.addin.Analysis.getYielddisc
320	RENDTO.PER.IRREGULAR.1	com.sun.star.sheet.addin.Analysis.getOddfyield
321	RENDTO.PER.IRREGULAR.2	com.sun.star.sheet.addin.Analysis.getOddlyield
322	RENDTO.VENCTO	com.sun.star.sheet.addin.Analysis.getYieldmat
323	REPETIR	REPT
324	RESIDUO	MOD
325	ROMANO	ROMAN
326	ROT13	com.sun.star.sheet.addin.DateFunctions.getRot13
327	SEGUNDO	SECOND
328	SEM.DEL.AÑO	WEEKNUM
329	SEMANAS	com.sun.star.sheet.addin.DateFunctions.getDiffWeeks
330	SEMANASENAÑO	com.sun.star.sheet.addin.DateFunctions.getWeeksInYear
331	SENO	SIN
332	SENOH	SINH
333	SI	IF
334	SIGNO	SIGN
335	SLN	SLN
336	SUBTOTALES	SUBTOTAL
337	SUMA	SUM
338	SUMA.CUADRADOS	SUMSQ
339	SUMA.PRODUCTO	SUMPRODUCT
340	SUMA.SERIES	com.sun.star.sheet.addin.Analysis.getSeriessum
341	SUMAR.SI	SUMIF
342	SUMAX2MASY2	SUMX2PY2
343	SUMAX2MENOSY2	SUMX2MY2
344	SUMAXMENOSY2	SUMXMY2
345	SUSTITUIR	SUBSTITUTE
346	SYD	SYD
347	T	T

Nº	Español	Ingles
348	TAN	TAN
349	TANH	TANH
350	TASA	RATE
351	TASA.DESC	com.sun.star.sheet.addin.Analysis.getDisc
352	TASA.INT	com.sun.star.sheet.addin.Analysis.getIntrate
353	TASA.NOMINAL	NOMINAL
354	TASA.NOMINAL_ADD	com.sun.star.sheet.addin.Analysis.getNominal
355	TENDENCIA	TREND
356	TEXTO	TEXT
357	TIEMPO	TIME
358	TIPO	TYPE
359	TIPO.DE.ERROR	ERRORTYPE
360	TIR	IRR
361	TIR.NO.PER	com.sun.star.sheet.addin.Analysis.getXirr
362	TRANSPONER	TRANSPOSE
363	TRUNCAR	TRUNC
364	UNICHAR	UNICHAR
365	UNICODE	UNICODE
366	VA	PV
367	VALOR	VALUE
368	VAR	VAR
369	VARA	VARA
370	VARP	VARP
371	VARPA	VARPA
372	VERDADERO	TRUE
373	VF	FV
374	VF.PLAN	com.sun.star.sheet.addin.Analysis.getFvschedule
375	VNA	NPV
376	VNA.NO.PER	com.sun.star.sheet.addin.Analysis.getXnpv
377	Y	AND

11.6 Listar fuentes en un archivo de Calc

La siguientes macros están basadas en la que muestra Andrew Pitonyak en su excelente libro de macros (ve la bibliografía), en el original se muestran los nombres de las fuentes en un cuadro de mensaje, con unos pequeños cambios las mostramos en un archivo nuevo de Calc. Las dos son muy parecidas, esta primera te muestra el nombre de la fuente junto con los estilos o variantes que soporta.

```
'Original de Andrew Pitonyak
'Estas versiones Mauricio Baeza
Sub ListarFuentes1()
```

```
Dim oToolkit As Object
Dim oDevice As Object
Dim oFontDescriptors As Object
Dim co1 As Long
Dim mArg()
Dim oNuevoDocumento As Object
Dim Campos(0) As New com.sun.star.table.TableSortField
Dim aSD(0) As New com.sun.star.beans.PropertyValue

   oToolkit = CreateUnoService("com.sun.star.awt.Toolkit")
   oDevice = oToolkit.createScreenCompatibleDevice(0, 0)
   oFontDescriptors = oDevice.FontDescriptors()

   oNuevoDocumento = StarDesktop.loadComponentFromURL( "private:factory/scalc", "_default", 0, mArg()
)
   With oNuevoDocumento.getSheets().getByIndex(0)
         .getCellByPosition(0,0).setString( "NÂº" )
         .getCellByPosition(1,0).setString( "Fuente" )
         For co1 = LBound(oFontDescriptors) To UBound(oFontDescriptors)
               .getCellByPosition(0,co1+1).setValue( co1+1 )
               .getCellByPosition(1,co1+1).setString( oFontDescriptors(co1).Name & " " &
oFontDescriptors(co1).StyleName )
         Next
         Campos(0).Field = 0
         Campos(0).IsAscending = True
         aSD(0).Name = "SortFields"
         aSD(0).Value = Campos()
         .getCellRangeByName( "B1:B" & CStr(co1+1) ).sort( aSD() )
         .getCellRangeByName("A1:B1").getColumns().OptimalWidth = True
   End With
End Sub
```

La segunda es similar, excepto que solo te muestra el nombre de la fuente, sin variantes, esto es por que la mayoría de estas variantes las establecemos por código con otros métodos y propiedades (negrita, cursiva, etc).

```
Sub ListarFuentes2()
Dim oToolkit As Object
Dim oDevice As Object
Dim oFontDescriptors As Object
Dim co1 As Long
Dim mArg()
Dim oNuevoDocumento As Object
Dim oFD As Object
Dim oCursor As Object
Dim Campos(0) As New com.sun.star.table.TableSortField
Dim aSD(0) As New com.sun.star.beans.PropertyValue

   oToolkit = CreateUnoService("com.sun.star.awt.Toolkit")
   oDevice = oToolkit.createScreenCompatibleDevice(0, 0)
   oFontDescriptors = oDevice.FontDescriptors()

   oNuevoDocumento = StarDesktop.loadComponentFromURL( "private:factory/scalc", "_default", 0, mArg()
)
   With oNuevoDocumento.getSheets().getByIndex(0)
         .getCellByPosition(0,0).setString( "Fuente" )
         For co1 = LBound(oFontDescriptors) To UBound(oFontDescriptors)
               .getCellByPosition(0,co1+1).setString( oFontDescriptors(co1).Name )
         Next

         'Con un filtro obtenemos registros únicos en la columna C
         oFD = .getCellRangeByName( "A1:A" & CStr(co1+1) ).createFilterDescriptor( True )
         oFD.SkipDuplicates = True
         oFD.ContainsHeader = True
```

```
        oFD.CopyOutputData = True
        oFD.OutputPosition = .getCellByPosition(2,0).getCellAddress()
        .getCellRangeByName( "A1:A" & CStr(co1+1) ).filter( oFD )

        'Eliminamos los datos originales, columna A
        .getColumns().removeByIndex(0,1)

        'Como filtramos, averiguamos cuantas filas quedaron
        oCursor = .createCursorByRange( .getCellByPosition(1,0) )
        oCursor.collapseToCurrentRegion()

        'Ordenamos los nombres de las fuentes
        Campos(0).Field = 0
        Campos(0).IsAscending = True
        aSD(0).Name = "SortFields"
        aSD(0).Value = Campos()
        .getCellRangeByName( "B1:B" & CStr(oCursor.getRows.getCount) ).sort( aSD )

        'Insertamos la numeracion de las fuentes
        .getCellByPosition(0,0).setString( "NÂº" )
        .getCellByPosition(0,1).setValue( 1 )
        .getCellRangeByName("A2:A" & CStr(oCursor.getRows.getCount) ).fillAuto( 0, 1 )

        'Autoajustamos el ancho de las columnas
        .getCellRangeByName("A1:B1").getColumns.OptimalWidth = True
    End With

End Sub
```

11.7 Listar formatos en un archivo de Calc

Solo tienes que pasarle la referencia al documento del cual te interese ver sus formatos, si le pasas un documento nuevo, siempre te devolverá los mismos formatos, si le pasas alguno de tus documento, incluirá todos tus formatos personalizados. Esta macro puede tener dos pequeñas mejoras que te quedan de tarea, primero, puedes discriminar los documentos para que solo soporte los que tengan el método getNumberFormats y segundo, en la columna de ejemplos, podrás ver en la columna E un ejemplo de cada formato, excepto los de la categoría Definidas por el usuario, esto es normal, pues el nuevo documento no tiene estos formatos, pero podrías implementar la copia de estos formatos para que se vea el resultado.

```
'Basada en el original de Andrew Pitonyak
'Esta versión Mauricio Baeza
Sub Todos_Los_Formatos(Documento As Object)
Dim oFormatos As Object
Dim mClaves, mArg()
Dim oFormato As Object
Dim co1 As Integer
Dim aLocal As New com.sun.star.lang.Locale
Dim oNuevoDocumento As Object
Dim sTipo As String

   'Para escribir los formatos en un nuevo documento de Calc
   oNuevoDocumento = StarDesktop.loadComponentFromURL( "private:factory/scalc", "_default", 0, mArg()
)
   With oNuevoDocumento.getSheets().getByIndex(0)
        'Obtenemos los formatos del documento pasado
        oFormatos = Documento.getNumberFormats()
```

```
        'Obtenemos todas las claves de los formatos
        mClaves = oFormatos.queryKeys( 0,aLocal,False )
        For co1 = LBound( mClaves ) To UBound( mClaves )
            'Obtenemos cada formato
            oFormato = oFormatos.getByKey( mClaves(co1) )
            'Numero secuencial
            .getCellByPosition( 0,co1 ).setValue( co1+1 )
            'Definimos la categoría del formato
            Select Case oFormato.Type
                Case 2 : sTipo = "Fecha"
                Case 4 : sTipo = "Tiempo"
                Case 6 : sTipo = "Fecha Tiempo"
                Case 8 : sTipo = "Moneda"
                Case 16 : sTipo = "Numero"
                Case 32 : sTipo = "Cientifico"
                Case 64 : sTipo = "Fraccion"
                Case 128 : sTipo = "Porcentaje"
                Case 256 : sTipo = "Texto"
                Case 1024 : sTipo = "Logica"
                Case 2048 : sTipo = "Indefinida"
                Case Else : sTipo = "Definido por el usuario"
            End Select
            .getCellByPosition( 1,co1 ).setString( sTipo )
            'Escribimos la cadena del formato
            .getCellByPosition( 2,co1 ).setString( oFormato.FormatString )
            'y su clave correspondiente
            .getCellByPosition( 3,co1 ).setValue( mClaves(co1) )
            'Escribimos un numero aleatorio entre 100 y 1000
            .getCellByPosition( 4,co1 ).setValue( co1+100+(Rnd*1000) )
            'Le establecemos el formato correspondiente
            .getCellByPosition( 4,co1 ).NumberFormat = mClaves(co1)
            'Ajustamos el ancho de las columnas
            .getCellRangeByName( "A1:E1" ).getColumns.OptimalWidth = True
        Next
    End With

End Sub
```

12 Bibliografía

Una de las motivaciones de este libro, es la falta de documentación es español, son realmente escasos los sitios que tratan la programación con OOo Basic en nuestro idioma.

En español:

Ariel Constela es un excelente compañero, siempre dispuesto a ayudar en las listas de correo, en su página encontraras interesantes ejemplos de código en OOo Basic, este libro le debe mucho a sus ejemplos: www.arielconstenlahaile.com.ar/ooo
NOTA: al día de hoy 19-Jul-10, el sitio esta desactivado, esperemos pronto tenerlo de vuelta.

La Guía de programación de StarOffice 8 para BASIC, es esencial para introducirte en la programación con OOo Basic: http://docs.sun.com/app/docs/doc/819-1327?l=es

En ingles:

En los foros de ayuda de OpenOffice.org, se publica mucho código interesante y siempre hay gente dispuesta a ayudarte:
http://user.services.openoffice.org/en/forum/viewforum.php?f=20

Un sitio con muchas macros de todos los niveles y sabores: www.ooomacros.org

El libro de Andrew Pitonyak si bien no es para novatos ya es todo un clásico en la red, cuidado, no es un libro que pueda tomarse como guia para aprender desde cero, es solo un conjunto bastante grande de ejemplos, eso si, muy buenos, pero serán incomprensibles si no tienes las bases elementales.

www.pitonyak.org/oo.php

StarOffice 8 Programming Guide for BASIC
http://docs.sun.com/app/docs/doc/819-0439

13 Índice Alfabético