Funciones y macros

En este apartado se incluirán algunas técnicas referentes a la programación del lenguaje Basic integrado en las hojas de cálculo. La gran mayoría de los textos se referirán a OpenOffice.org Calc, aunque algunos serán de aplicación a Microsoft Excel.

Las técnicas incluidas no seguirán un orden estricto, sino que se irán publicando según surjan las cuestiones que motiven su estudio.

Contenido

Define tus propias funciones

En ocasiones desearás definir funciones que la hoja de cálculo no ofrece. Por ejemplo, en Electricidad nos puede convenir definir la resistencia equivalente a otras dos

situadas en paralelo, o en Geometría, la función que devuelve una hipotenusa en función de los dos catetos. Mediante un uso sencillo de las macros puedes lograrlo.

Secuencia para definir tus propias funciones

1) Abrir el Editor de Basic

En OpenOffice

Sigue el menú Herramientas > Macros > Organizar macros > OpenOffice Basic para abrir el editor.

Si es la primera función que defines, busca la carpeta Standard correspondiente al nombre de tu modelo (si lo acabas de crear, se llamará Sin Nombre). No señales la otra carpeta Standard, que es más general.

Una vez elegida la carpeta, pulsa el botón Nuevo para abrir un módulo contenedor. Se te ofrecerá el nombre de module1, module2 u otro similar. Acepta el nombre o cámbialo según tu criterio. Al aceptar el nombre se abrirá el editor de macros. Por defecto aparecerá la macro Main, que puedes borrar o ignorar.

Escribe debajo el código de tu función, según se explica en el siguiente apartado.

En Excel

Sigue el menú Herramientas > Macro > Editor de Visual Basic, o pulsa Alt + F11
Si es la primera función que defines, la pantalla aparecerá en gris. Debes crear un módulo nuevo con Insertar - Módulo, y Excel le dará el nombre de Módulo 1.

Escribe debajo el código de tu función, según se explica en el siguiente apartado.

2) Escritura del código

Terminada la secuencia anterior, comienza a escribir el código de una función-
Debes comenzar con

Function nombre de la función (argumento)
y terminar con
End function

y entre ambas, el código de la función.

En ese código debemos usar el nombre de la función seguida del signo igual y de su definición
Es mejor verlo con un ejemplo:

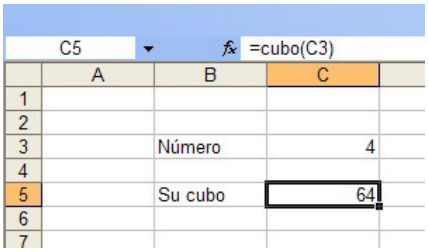
Function cubo (numero)
cubo=numero*numero*numero
End function

En el ejemplo, el nombre de la función es cubo, y su argumento numero (lo traduciríamos como "Cubo de un número")

Después volvemos a escribir cubo, el signo igual, y su definición.

3) Uso de la función

Una vez escrito el código, cierra el Editor de Basic y usa tu función en cualquier celda. En la imagen puedes ver una celda definida mediante la nueva función CUBO.



Con esto ya tienes definida la función.

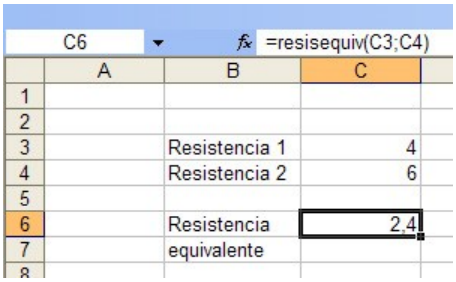
Con la técnica explicada, esa función sólo estará activa en la hoja de cálculo en la que la has creado, no en otras. Al cerrar la hoja ya no podrás usarla.

Función con varios argumentos

Una función puede actuar sobre varios argumentos, por ejemplo la resistencia equivalente a la que se aludía en el primer párrafo. En ese caso, se deberán separa mediante una coma:

Function resisequiv(r1, r2)
resisequiv = r1 * r2 / (r1 + r2)
End Function

Cuando uses esta función en una celda, debes sustituir la coma por un punto y coma., por ejemplo resisequiv(4;6). Estudia el ejemplo de la imagen:



Variables auxiliares

En una definición puedes usar las estructuras del Basic: FOR...NEXT, SELECT CASE, etc. Aquí sólo usaremos DIM, para crear variables auxiliares. Observa este ejemplo

Function area6(lado)
Dim perimetro, apotema

perimetro = 6 * lado
apotema = lado * Sqr(3) / 2
area6 = perimetro * apotema / 2
End Function

que devuelve el área de un hexágono en función del lado. El nombre de la función, en este caso area6, debe figurar en la definición, aunque uses otras variables

Macros de apertura

En ocasiones podemos desear que se ejecute cualquier operación al abrir una hoja de cálculo, como borrar un rango, abrir una hoja determinada, dar un valor a una celda, etc.
Para lograrlo debes, en primer lugar, escribir o grabar una macro con las operaciones que deseas. Una vez escrita, el procedimiento para que se ejecute al abrir una hoja cambia mucho si trabajas en Excel o si lo haces en OpenOffice.

En Excel

Basta con entrar en el Editor de Visual Basic (Alt - F11), buscar la macro que has escrito y cambiarle el nombre por Auto_Open. Nada más.

Ejemplos

Sub Auto_Open()

```
Sheets("Hoja3").Select
Range("E6").Select
ActiveCell.FormulaR1C1 = "22"
End Sub
```

Esta macro, al abrir el archivo, seleccionará la Hoja3, situará el cursor en la celda E6 y escribirá en ella un 22

```
Sub Auto_Open
Range("A1:D20").Select
Selection.ClearContents
End Sub
```

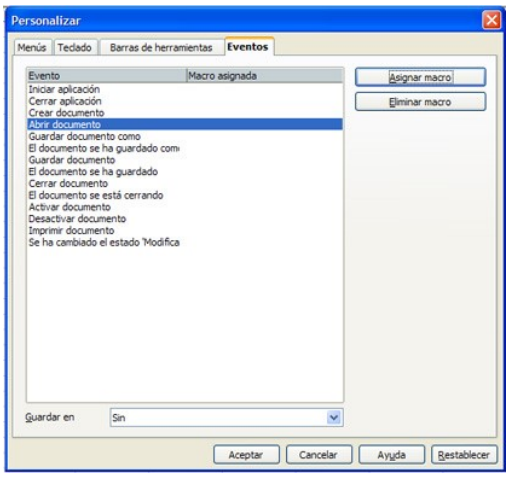
Esta otra selecciona el rango A1:D20 y borra su contenido

En OpenOffice

Aquí el procedimiento es totalmente distinto.

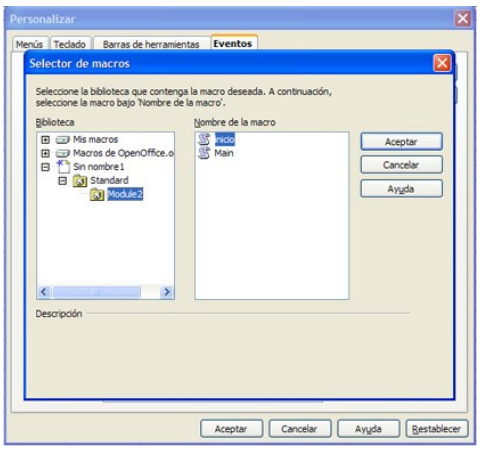
Escribes o grabas una macro y le das el nombre que desees. Supongamos que fuera Sub inicio().

Una vez construida la macro, abres el menú Herramientas - Personalizar, eliges la pestaña Eventos y señalas con el ratón el evento de Abrir Documento.



Pulsas sobre el botón Asignar macro, y la buscas (por ejemplo inicio). Debes saber dónde está. En la imagen estaría en el documento Sin nombre, macros Standard, módulo Module2.

Aceptas dos veces y ya tienes una macro de inicio.



en cuyo caso se considerará un desplazamiento de +1 fila y 0 columnas.

Cómo sumar datos dispersos etiquetados

En ocasiones se usan tablas de recogida de datos en las que existen algunos de la misma naturaleza pero que aparecen dispersos. Por ejemplo, calificaciones correspondientes a preguntas aisladas en una clase. Como no se pregunta cada día a los mismos alumnos o alumnas, sus notas aparecerán en la tablas de forma desordenada. Sin embargo, lo usual es que vengan acompañadas de una etiqueta que recuerde a quién pertenece la calificación. Si después se desean contar o sumar estos datos, ninguna función de Excel u OpenOffice nos resolvería el problema de forma satisfactoria.

Un ejemplo típico se da cuando la organización de los datos se efectúa mediante múltiples fichas personales, que pueden llenar toda una hoja. En la imagen se registran los pedidos de botellas que ha efectuado un socio de un Club de Vinos. A la derecha figuran los totales mensuales, que se habrán obtenido con las funciones generales de Excel.

Socio núm	23
Antonio Gómez García	
Fecha	Botellas
03/04/2007	5
22/04/2007	6
05/05/2007	2
02/06/2007	1
15/06/2007	3

Mes	
Enero	
Febrero	
Marzo	
Abril	11
Mayo	2
Junio	4
Julio	
Agosto	
Septiembre	
Octubre	
Noviembre	
Diciembre	
Total	17

Imaginemos que existen numerosas fichas de este tipo y que se desea sumar o contar todas las botellas enviadas en el mes de Abril. En ese caso se deberá explorar toda la hoja, y cuando se encuentre la etiqueta "Abril", sumar la

cantidad que figure a su derecha. Para lograrlo podemos definir en Basic dos funciones nuevas. Habrá que tener en cuenta cuántas filas y columnas separan la etiqueta del dato. En este ejemplo sería +1 columna (está a su derecha) y 0 filas, pero la etiqueta puede estar escrita superiormente al dato, como en

Alfredo Gómez
8,3

en cuyo caso se considerará un desplazamiento de +1 fila y 0 columnas.
Se desarrollan a continuación posibles códigos para resolver la situación. Se escriben en negro las líneas que funcionan indistintamente en Excel y OpenOffice, en verde las que hay que usar sólo en Excel, y en rojo las que sólo se escribirán en OpenOffice. Finalmente, lo escrito en azul y en cursiva corresponderá a comentarios. Para editar este código se deberán seguir las instrucciones del apartado Define tus propias funciones.

Sumar los datos de todas las apariciones de una etiqueta

Esta función sumaría los datos de las apariciones de la etiqueta: mes, alumno/a, marca de coche, etc. a$ representa a la etiqueta que estamos buscando. Los parámetros filas y columnas representan el desplazamiento

que existe entre etiqueta y dato. En el ejemplo de los vinos serían 0 y +1 respectivamente.

Public Function sumar_etiq(a$,filas, columnas)
Dim i, j, suma, suma0
Dim g$

suma = 0
For i = 1 To 3000 filas que abarcan los datos. Hay que cambiar el 3000 por el número de cada ejemplo concreto
For j = 1 To 20 columnas que abarquen los datos

g$ = ActiveWorkbook.Sheets(1).Cells(i, j).Value Línea sólo para Excel. Lee el contenido de las celdas para descubrir la etiqueta y recogerla en la variable g$
g$= StarDesktop.CurrentComponent.sheets(0).GetCellByPosition(j-1,i-1).String Línea sólo para OpenOffice. Similar a la anterior.

If g$ = a$ Then Comprueba si la celda contiene la etiqueta solicitada
suma0 = ActiveWorkbook.Sheets(1).Cells(i+filas, j+columnas).Value Lee el dato asignado a la etiqueta (para Excel)
suma0= StarDesktop.CurrentComponent.sheets(0).GetCellByPosition(j-1`columnas,i-1+filas).String Línea sólo para

OpenOffice. Similar a la anterior. Las filas y columnas presentan orden inverso

```
suma = suma + suma0
End If
Next j

Next i

sumar_etiq = suma

End Function
```

Una vez definida, se puede usar en cualquier celda. En el ejemplo de los vinos, para buscar Abril deberíamos escribir en cualquier celda
=SUMAR_ETIQ("Abril",0,1)
El 1 representa el desplazamiento de una columna a la derecha y el 0 que el dato se encuentra en la misma fila.

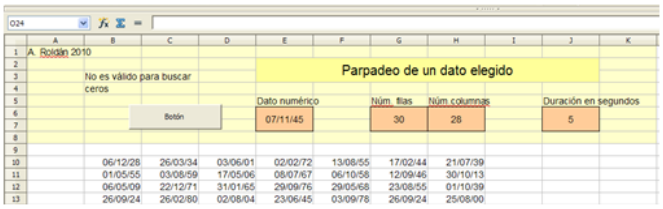
Parpadeo de un dato elegido

En algunas situaciones prácticas podemos tener una gran abundancia de datos que hagan casi imposible su exploración visual. Entre ellos pueden existir algunos cuya ubicación nos interese. Por ejemplo, un lector me indicaba que para él sería útil que se destacaran los pagos que vencieran en la fecha actual. Así, cada vez que abriera la

hoja, encontraría parpadeando los que hubieran llegado a su fecha.

No es fácil conseguir que parpadeen las celdas que contengan un dato que nos interese. Por eso nos vamos a tener que basar en unos datos previos que le indiquen a la hoja de cálculo qué tiene que buscar, dónde y cuántos segundos ha de mantener el parpadeo. Con más calma quizás se pudiera prescindir de alguno de ellos, pero el tema no merece más atención.

En la imagen vemos una posible cabecera. En las celdas situadas debajo podemos imaginarnos que existen grandes cantidades de datos, y que entre ellos está el elegido. En este caso hemos usado una fecha 07/11/45 y deberán parpadear todas las celdas inferiores que la contengan.



La macro que construyamos deberá leer las celdas y asignarles una variable. Supongamos que en E6 se lee Valor, en G6 Fila, H6 Columna y J6 Pausa. Esas son las variables que usaremos en el código Basic.

Una vez leídas se inician también t0, que leerá el reloj interno (timer) e indi, que llevará la cuenta de las celdas que contienen el dato elegido. También hay que preparar

memorias fil y col que nos indiquen dónde está situado el dato.

Así, la estructura de nuestra macro podría ser:

(A) Se leen las variables

Leemos Valor, Fila, Columna y Pausa, el reloj en t0 y se pone a cero el contador de veces que aparece el dato (indi) y las memorias fil y col

```
pausa=StarDesktop.CurrentComponent.sheets(0).GetCellByPosition(9,5).value
valor=StarDesktop.CurrentComponent.sheets(0).GetCellByPosition(4,5).value
fila=StarDesktop.CurrentComponent.sheets(0).GetCellByPosition(6,5).value
columna=StarDesktop.CurrentComponent.sheets(0).GetCellByPosition(7,5).value
t0=timer
t1=t0
indi=0
for i=0 to 50:fil(i)=0:col(i)=0:next i
```

(B) Se recorren las filas y columnas.

Cuando se encuentre el dato elegido se incrementa indi y se rellenan fil(indi) y col(indi) con la referencia de la celda.

(C) Se programa una pausa temporal.

Se puede hacer con esta rutina:

```
t1=t0
while t1
‘Se realiza el trabajo de recorrer celdas y parpadear
t1=timer
wend
```

La hora del reloj está almacenada en t0. La otra variable t1 va leyendo el timer y cuando sobrepasa la pausa se detiene. Dentro del bucle se realiza el parpadeo.

(D) Parpadeo de las celdas elegidas

Consistirá simplemente en asignar dos colores distintos a las celdas y separarlos mediante un pequeño intervalo de tiempo:

```
if indi>0 then
for i=1 to indi
StarDesktop.CurrentComponent.sheets(0).GetCellByPositio
n(col(i),fil(i)).cellbackcolor=rgb(255,100,200)
next i
end if
‘colorea de rojo apagado
```

for i=1 to 1e4:next i

‘mantiene ocupado el ordenador por un tiempo

if indi>0 then
for i=1 to indi
StarDesktop.CurrentComponent.sheets(0).GetCellByPositio
n(col(i),fil(i)).cellbackcolor=rgb(255,255,255)
next i

‘Vuelve a colorear de blanco

Como ejercicio de programación es divertido. Seguro que alguien lo puede mejorar.

Copiamos a continuación todo el código en Basic de OpenOffice.org

Sub parpadeo
dim pausa,valor,fila,columna,celda
dim t0,t1,i,j,k,indi
dim fil(50),col(50)

***pausa=StarDesktop.CurrentComponent.sheets(0).GetC
ellByPosition(9,5).value***
***valor=StarDesktop.CurrentComponent.sheets(0).GetCel
lByPosition(4,5).value***
***fila=StarDesktop.CurrentComponent.sheets(0).GetCellB
yPosition(6,5).value***

```
columna=StarDesktop.CurrentComponent.sheets(0).Ge
tCellByPosition(7,5).value

t0=timer
t1=t0
indi=0
for i=0 to 50:fil(i)=0:col(i)=0:next i
for i=8 to fila
for j=1 to columna
celda=StarDesktop.CurrentComponent.sheets(0).GetCe
llByPosition(j,i).value
if celda=valor then

indi=indi+1
fil(indi)=i:col(indi)=j
end if
next j
next i
while t1

if indi>0 then
for i=1 to indi
StarDesktop.CurrentComponent.sheets(0).GetCellByPo
sition(col(i),fil(i)).cellbackcolor=rgb(255,100,200)
next i
end if
for i=1 to 1e4:next i
if indi>0 then
for i=1 to indi
```

```
StarDesktop.CurrentComponent.sheets(0).GetCellByPo
sition(col(i),fil(i)).cellbackcolor=rgb(255,255,255)
next i
end if
t1=timer
wend

End Sub
```

Tus funciones, disponibles en todas las hojas de cálculo

Procedimiento para Excel

El autor necesita frecuentemente descomponer un número en factores primos. Como esta función no viene implementada en la hoja de cálculo, ha tenido que programarla en el Basic de Excel. El problema que surge es que sólo está disponible en la hoja que contiene el código y no en cualquier otra que se cree. Esto tiene un remedio, y es la construcción de un complemento de Excel que nos permita acceder a esa factorización cuando se abra cualquier hoja.

Complementos de Excel

Para saber de qué estamos hablando, entra en las Opciones de Excel y busca Complementos. En la ventana

que se abre podrás comprobar qué complementos tienes instalados en tu equipo

(el volcado de pantalla corresponde al Excel 2007 sobre Windows XP, una querida antigüedad, pero igual te funciona en Excel 2010)

En la imagen vemos que el autor tiene instaladas dos herramientas de análisis, el Solver y un complemento suyo titulado Micomplemento. Como habrás comprendido, los cuatro contienen funciones y rutinas que no vienen implementadas en Excel originariamente.

Crea tu propio complemento

Al final de este artículo se ha incluido el código mínimo necesario para implementar la descomposición factorial de un número entero (dentro de los límites de Excel y del propio código, no le pidas milagros) como un regalo del autor a sus lectores.

Pasos a seguir

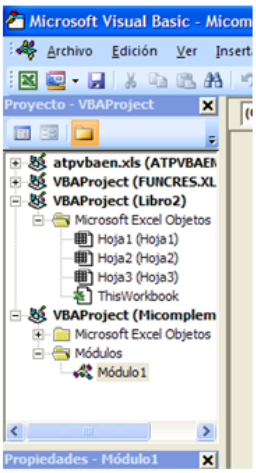
En primer lugar tienes que escribir tus funciones. En el caso que estamos desarrollando basta con que las copies desde el final de esta entrada. Abre un archivo nuevo y pega en él las definiciones que desees según te explicamos a continuación:

Una vez decidido el código deberás pasarlo a Excel. Para ello acude a la pestaña Programador de la cinta de opciones. Si no la tienes visible deberás activarla en Opciones de Excel – Más frecuentes.

Entras en el ámbito de programación mediante 

el primer botón de la ficha Programador:

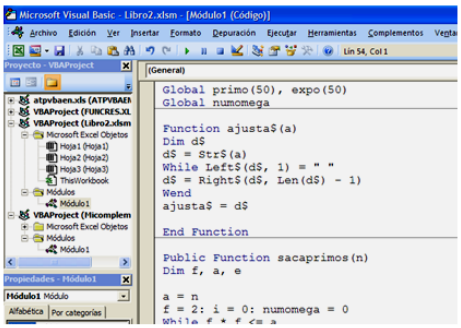
Te aparecerá el acceso a las macros que utiliza tu hoja de cálculo en este momento:



A ti no te aparecerá la referencia a Micomplemento. También, si usas la versión 2010 los colores podrán cambiar, pero el contenido será el mismo.

Ahora debes crear un módulo que aloje tu código. Pide Insertar – Módulo y Excel lo hará con el nombre de Modulo 1 (salvo que tengas otro anterior).

En la hoja en blanco que aparece pega el código que habrás copiado desde esta entrada o que haya sido creado por ti:

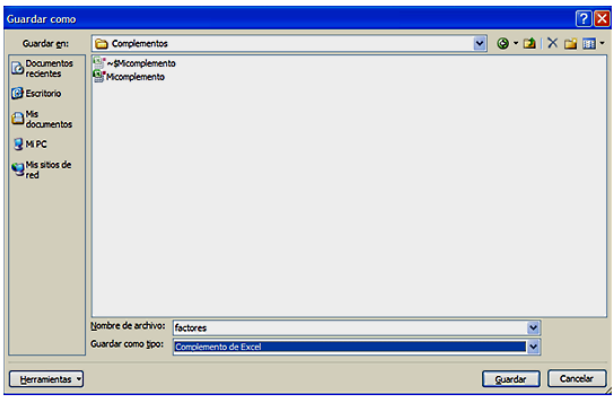


Ahora puede ser un buen momento para comprobar si todo va bien. Guarda el archivo nuevo como Libro habilitado para macros. Vuelve a la hoja. Escribe cualquier número entero, por ejemplo 366220 en la celda B4. En otra celda escribe =factores(B4). Si ves escrito [2,2][5,1][18311,1] es que tu función se comporta bien. La interpretación de lo que ves es que el primer número de cada corchete es el factor primo y el segundo el exponente al que está elevado. En este caso 366220=22*5*18311. No intentes cálculos con esta expresión, que tiene formato de texto.

Lo que has construido hasta ahora sólo te vale para el archivo que contiene el código. Para que se active en cualquier hoja hay que convertirlo en complemento.

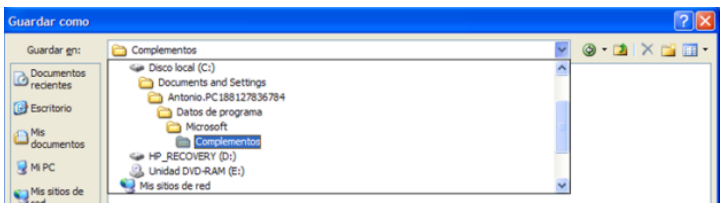
Instalación del complemento

Borra si acaso los cálculos efectuados y vuelve a guardar el libro como complemento de Excel. Puedes cambiarle el nombre a factores. Guíate por la imagen

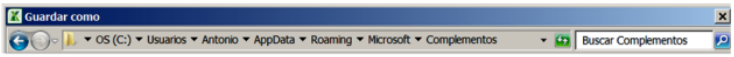


Observa que Excel te lleva a la carpeta Complementos, que es donde debe estar alojado el tuyo. No cambies esa carpeta, que si no, no podrás instalar el complemento.

Puedes acceder a la ruta en la que está situada la carpeta:



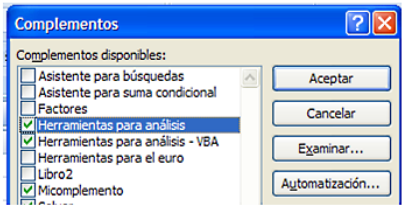
En Office 2010 se te muestra también toda la ruta, que es distinta a la anterior:



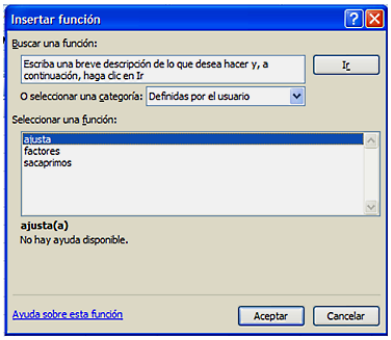
Es interesante conocer esa ruta, por si deseas borrar el archivo.

Instalación

Ya sólo te falta instalar tu complemento. Vuelve a las opciones de Excel y busca Complementos. En la parte inferior de la ventana tendrás el botón Ir… Úsalo y descubrirás que tu trabajo está preparado ya para ser usado:



Activa la casilla de verificación que está junto al nombre Factores y pulsa Aceptar. Si todo ha ido bien, cuando abras Excel de nuevo, en el catálogo de funciones definidas por el usuario dispondrás de la función factores:



Las otras dos funciones ajusta y sacaprimos son auxiliares y no tienes por qué usarlas, ya que quizás no interpretarías bien su resultado.

Ahora define tú un complemento propio ¡Suerte!

Código en Basic

```
Global primo(50), expo(50)
Global numomega

Function ajusta$(a)
Dim d$
d$ = Str$(a)
While Left$(d$, 1) = " "
d$ = Right$(d$, Len(d$) - 1)
Wend
ajusta$ = d$

End Function

Public Function sacaprimos(n)
Dim f, a, e
```

```
a = n
f = 2: i = 0: numomega = 0
While f * f <= a
e = 0
While a / f = Int(a / f)
e = e + 1
a = a / f
Wend
If e > 0 Then
numomega = numomega + 1
primo(numomega) = f
expo(numomega) = e
End If
If f = 2 Then f = 3 Else f = f + 2
Wend
If a > 1 Then
numomega = numomega + 1
primo(numomega) = a
expo(numomega) = 1
End If
sacaprimos = numomega

End Function

Public Function factores(n) As String
Dim a, nn
Dim s$
```

```
'saca factores en forma de string
a = n
nn = sacaprimos(a)
s$ = ""
For i = 1 To numomega
s$ = s$ + "[" + ajusta(primo(i)) + "," + ajusta(expo(i)) +
"]"
Next i
factores = s$
End Function
```

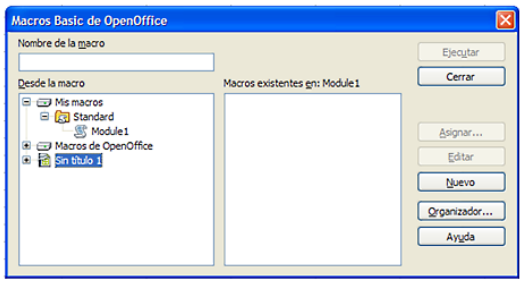
Procedimiento para Apache OpenOffice y LibreOffice

En los párrafos anteriores proponíamos un procedimiento para extender tus funciones a cualquier hoja de cálculo que abras. Propusimos el procedimiento de crear e instalar un complemento para Excel. Si pasamos a Apache OpenOffice o LibreOffice, la creación de complementos (extensiones) se complica, porque está orientada al uso de terceros. Como aquí sólo nos interesa que tengas disponibles tus funciones en cualquier archivo nuevo que crees para tu propio uso, desarrollaremos un método mucho más sencillo.

Usaremos el código que se presentó más arriba para descomponer un número natural en sus factores primos. Cópialo y guárdalo, porque te servirá aquí.

Una vez decidido el código deberás pasarlo a Apache OpenOffice o a LibreOffice. El procedimiento es similar en ambos programas, y sólo añadiremos los detalles específicos de LibreOffice si fuera necesario.

Abre una hoja nueva. Acude al menú Herramientas y en él elige Macros, después Organizar macros y finalmente OpenOffice Basic (o LibreOffice Basic)



Observa que tu archivo aparecerá en la parte baja (en la imagen aún no tiene título). Tú has de ir a la superior, “Mis macros - Standard”. Pide crear un módulo nuevo con el botón “Nuevo” de la parte derecha. Si ya existe uno, como ocurre en la imagen, le asignará el nombre de Module 2 u otro similar.

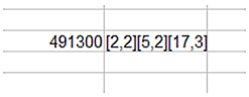
Abre el nuevo módulo que has creado (pinchando sobre su nombre) y pégale el código que desees, como el que te ofrecimos en la anterior entrada. Acepta y cierra todo.

Ahora puede ser un buen momento para comprobar si todo va bien. Vuelve a la hoja. Escribe cualquier número entero,

por ejemplo 366220 en la celda B4. En otra celda escribe =factores(B4). Si ves escrito [2,2][5,1][18311,1] es que tu función se comporta bien. La interpretación de lo que ves es que el primer número de cada corchete es el factor primo y el segundo el exponente al que está elevado. En este caso 366220=22*5*18311. No intentes cálculos con esta expresión, que tiene formato de texto.

Como has usado el contenedor “Mis macros”, todo lo que has construido hasta ahora lo encontrarás implementado en cualquier libro que abras. Prueba a hacerlo. Cierra el archivo, abre uno nuevo, y en cualquier celda escribe un número entero y aplícale la función factores (no aparecerá en ningún catálogo. Te lo tienes que aprender)

En la imagen se ha descompuesto el número 491300 en factores dentro de un archivo recién creado:



Ahora inténtalo tú.

Funciones recursivas en las hojas de cálculo

Cuando yo programaba hace años en Pascal se nos vendía su posibilidad de usar la recursión, es decir, que una función se llamara a sí misma, en declaraciones del tipo

Factorial(n)=n*factorial(n-1)

Toda función recursiva contiene una llamada a sí misma, directa o indirectamente a través de otra función. Como esto nos puede llevar a un proceso sin fin, debe contener también un código de parada, que suele ser una definición en un caso concreto, como veremos en los ejemplos.

La recursividad no se resuelve hasta que no desemboca en ese caso de parada. Mientras tanto hay que guardar los datos pendientes situándolos en una pila. Por tanto, ahí está el único problema de usar la recursividad en las hojas, y es que se puede agotar la pila si se alargan mucho los cálculos, con el consiguiente mensaje de error. Un fallo de principiante es programar una función recursiva sin facilitar su salida. En ese caso el error será más grave aún: un cálculo sin fin.

Explicamos a continuación algunas funciones recursivas, comenzando con el factorial, que es la más popular y que nos servirá para explicar algunos detalles:

```
Public Function factorial(n)
Dim f
If n = 0 Then f = 1 Else f = factorial(n - 1) * n
```

factorial = f
End Function

Es fácil entender el código: Para evitar confusiones, comenzamos almacenando el factorial en la variable f, para al final recoger su valor en factorial. El cálculo de f es el clásico de la función n!: si n es cero, definimos el factorial como 1 y en caso contrario multiplicamos por n el factorial de n-1.

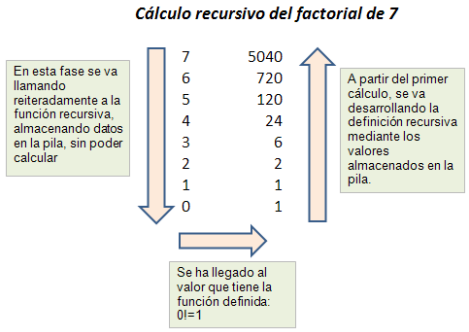
Prueba esta función en cualquier hoja de cálculo

Este primer ejemplo contiene las dos partes imprescindibles en una función de este tipo:

- Alguna asignación de un valor concreto, que servirá para detener la primera fase de lectura de datos y comenzar los cálculos hacia atrás. Aquí es la asignación 0!=1

- La definición recursiva propiamente dicha, que, como es conocido, consiste en exigir que n!=n*(n-1)!

Lo explicamos con un esquema:



Al intentar calcular el factorial de 7, el programa se encuentra con una referencia al factorial de 6, guarda el 7 en la pila y se dedica a calcular el nuevo factorial. Como no puede, almacena el 6 encima del 7 (es una pila) y lo intenta con el 5, y así va de fracaso en fracaso (flecha descendente) hasta llegar al 0.

El valor 0 admite el cálculo, porque está definido como 1. Resuelto esto, es como si el programa se preguntara: ¿por dónde iba? Acude a la pila y ve un 1, con lo que ya puede calcular 1!=1*0!=1 y así sigue (flecha ascendente) buscando datos en la pila y resolviendo los cálculos según la definición recursiva.

Es evidente que para números grandes la pila se puede agotar por falta de memoria asignada.

Con esta función recursiva (la más inútil que me he inventado nunca) puedes tener una idea de la amplitud de la pila de tu hoja de cálculo.

Public Function identidad(n)

```
Dim i
If n = 1 Then i = 1 Else i = identidad(n - 1) + 1
identidad = i
End Function
```

La he elegido para que no influya la magnitud de los números, sino la cantidad de ellos que permanecen en la pila. Con ella he llegado la valor n=3270 como el último que no me da error. En los siguientes no consigo realizar el cálculo.

¿Qué margen tendrá tu hoja de cálculo? Prueba a ver.

Ejemplos varios

Si deseas el enésimo número triangular, sólo tienes que usar este código:

```
Public Function triang(a)
Dim p
If a = 1 Then p = 1 Else p = triang(a - 1) + a
triang = p
End Function
```

También se entiende bien: en los números triangulares vamos añadiendo en cada paso una base del triángulo

nueva con a elementos. Prueba esta función y si quieres compara los resultados con la clásica fórmula Tn=n(n+1)/2.

¿Puedes analizar esta función?

```
Public Function cuad(a)
Dim p
If a = 1 Then p = 1 Else p = cuad(a - 1) + 2 * a - 1
cuad = p
End Function
```

¿Por qué produce como resultado el cuadrado de a? Este es un bonito ejemplo de elevar un número al cuadrado sin multiplicar en ningún momento.

Y ya que estamos con números poligonales, podríamos generarlos todos con una función recursiva única que dependiera de a y también del número de lados del polígono. ¿Te atreves con ella?

¿Y qué opinas de esta, con dos variables?¿Qué resultado produce?

```
Public Function combi(m, n)
Dim c
If n = 0 Then c = 1 Else c = combi(m, n - 1) * (m - n + 1) /
n
combi = c
End Function
```

Un ejemplo elegante

Define esta función de texto

Public Function simetrico$(a$)
Dim s$
If a$ = "" Then s$ = "" Else s$ = simetrico(Right$(a$, Len(a$) - 1)) + Mid$(a$, 1, 1)
simetrico = s$
End Function

Escribe una palabra en una celda y aplícale esta función desde otra celda ¿Cuál es el resultado?

Como ves, todo esto es bastante divertido, pero no muy útil a causa del agotamiento del espacio de memoria asignado a la pila de datos.

Y ahora tú. ¿Cómo hallarías, mediante una función recursiva, el término general en estas sucesiones?

Progresiones aritméticas y geométricas.
La sucesión de Fibonacci (¡cómo no!)
La enésima potencia de un número dado.