

## Iniciación a las Macros de Excel

### 1.- Índice

Iniciación a las Macros de Excel .....	1
1.- Índice .....	1
2.- Objetivo .....	2
3.- Mi primera macro .....	3
3.1.- Activación de barra de herramientas .....	3
3.2.- Grabar una macro .....	4
3.3.- Ejecución de macros .....	4
3.4.- Acceder al editor de Visual Basic .....	4
3.4.1.- Configuración de ventanas .....	5
3.4.2.- Ver el código .....	6
3.4.3.- Estructura de Objetos .....	7
3.5.- Nivel de seguridad .....	7
4.- Inicio de Programación en Excel .....	8
4.1.- Insertar controles .....	8
4.2.- Acceder a las celdas .....	9
4.3.- Objeto Sheet y Workbook .....	10
4.4.- Objeto Application .....	11
4.4.1.- StatusBar .....	11
4.4.2.- ScreenUpdating .....	11
4.4.3.- MousePointer .....	11
4.5.- Función Active .....	11
4.6.- Escribir Fórmulas .....	11
4.7.- Insertar Objetos .....	12
4.8.- Mejorar rendimiento .....	12
5.- Detalles técnicos .....	14
5.1.- Compilar .....	14
5.2.- Depurar y Breakpoints .....	14

## 2.- Objetivo

El siguiente documento tiene como objetivo explicar cómo realizar macros en Excel mediante el lenguaje *Visual Basic for Applications* (VBA)<sup>1</sup>.

Para ello, es necesario disponer de conocimientos básicos de algoritmia y de programación en Visual Basic. En caso de no tenerlos, te aconsejo encarecidamente que te descargues los siguientes tutoriales:

Documentación > Programación > Manual de Programación

Documentación > Programación > Iniciación a Visual Basic

En Internet existen muchos tutoriales que explican todas las funcionalidades e instrucciones sobre este tema, pero ofrecerte este tipo de material NO es el objetivo de este manual. Pretendo que aprendas a utilizar la herramienta macros y que con este conocimiento puedas incrementar tus habilidades para configurar las opciones que desees a través de macros, sin necesidad de recurrir siempre a instrucciones predeterminadas, que por definición delimitan la flexibilidad de opciones.

**No os doy el pescado, os enseñaré a pescar**

---

<sup>1</sup> Existe un lenguaje de programación llamado Visual Basic, propiedad de Microsoft. Las macros (de excel, de word, de power point...) se generan en una "extensión" de este lenguaje llamado Visual Basic For Applications.

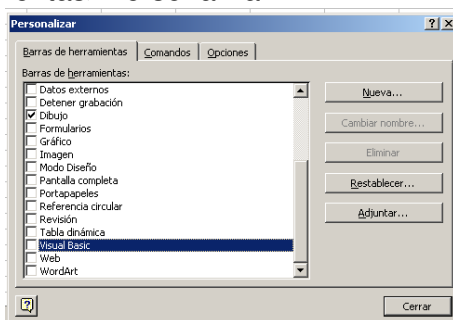
### 3.- Mi primera macro

En este apartado crearás una macro y te explicaremos como acceder a ella y como parametrizarla.

#### 3.1.- Activación de barra de herramientas

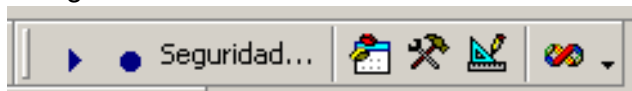
Antes de empezar vamos a explicarte como poner la barra de herramientas de macros.

Vamos a Herramientas>Personalizar...



Y activas la que se llama Visual Basic.

Te aparecerá la siguiente barra de herramientas.



La funcionalidad de estos botones es:

Icono	Funcionalidad
	Ejecución de una macro ya existente
	Grabar y parar la grabación de una nueva macro
	Acceder al formulario para la gestión de la seguridad de macros
	Ir al editor de macros (Visual Basic)
	Menú donde se muestran los controles a poner, léase botones, listas...
	Para poder acceder a los controles en modo diseño. Por ejemplo, si pulsas un botón en modo diseño, no se ejecutará.
	Acceder al entorno de programación de Microsoft

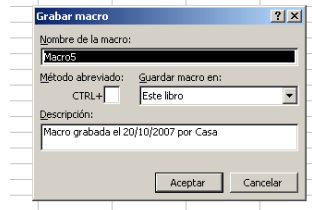
### 3.2.- Grabar una macro

Nuestra primera macro consistirá en escribir un texto en una celda y ponerla en negrita:

1.- Pulsar el botón de grabación de la barra de herramientas de macros.



Al pulsar el botón te aparecerá esta ventana; a continuación, pulsar el botón de Aceptar:



Los campos del formulario asociados a la grabación de la macro son:

- *Nombre de la macro:* nombre del procedimiento que se creará, ¡acuérdate del nombre de la macro!
- *Método abreviado:* para que la macro se ejecute con la combinación de Ctrl + la tecla que definas
- *Guardar macro en...:* si deseas que se grabe en este libro de trabajo o en otro, que determines.
- *Descripción:* comentario del procedimiento

2.- Escribir algo y ponerlo en negrita; por ejemplo, en la celda B5 poner *prueba*.

3.- Pulsar el botón de parar grabación de la barra de herramientas de macros



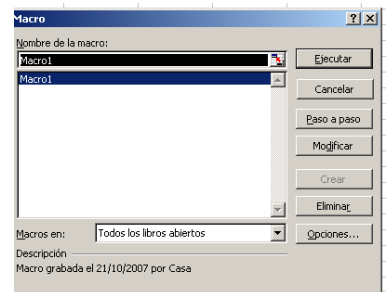
Ya has creado tu primera macro.

### 3.3.- Ejecución de macros

Vas a realizar una ejecución de tu macro. Primero debes borrar el contenido y quitar las negritas de la celda B5 para así poder comprobar que se ejecuta correctamente la macro definida en el primer punto.

Al pulsar el botón de ejecución se te mostrará la siguiente pantalla:

En esta pantalla se muestran todas las macros creadas en todos los libros de trabajo abiertos. Marca tu macro (la puedes distinguir por el nombre que la macro) y pulsa el botón ejecutar:



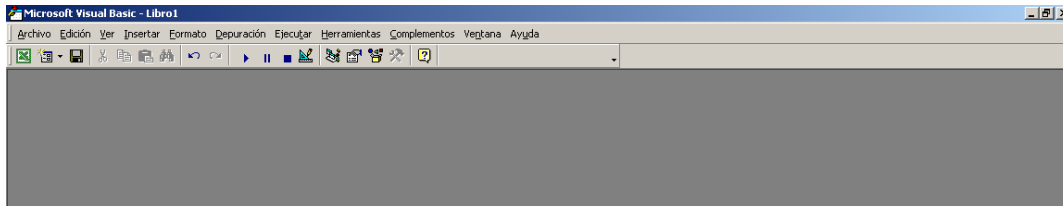
### 3.4.- Acceder al editor de Visual Basic

En este punto explicaremos como ver el código de nuestra macro. Empieza pulsando el botón de editor de macros:



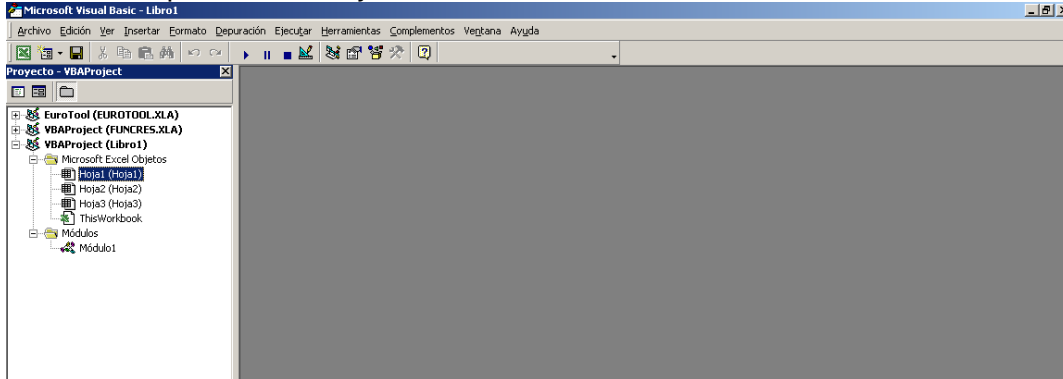
### 3.4.1.- Configuración de ventanas

Es posible que se te muestre una pantalla un tanto vacía:

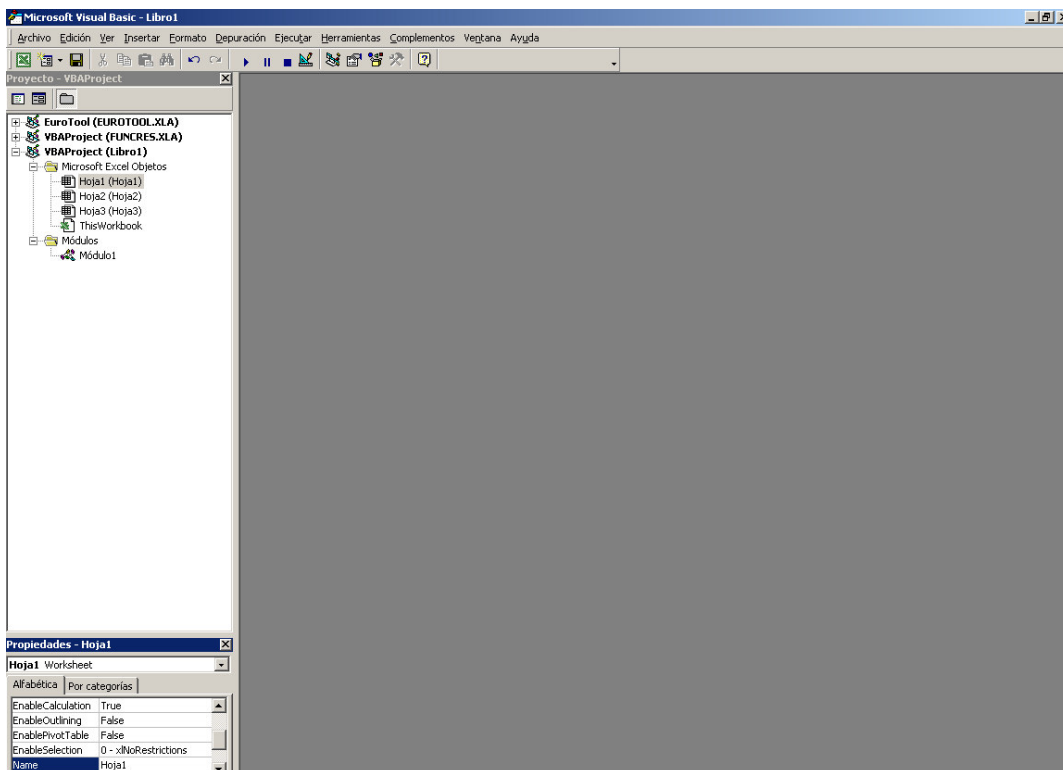


Los siguientes pasos pretenden activar las diferentes ventanas:

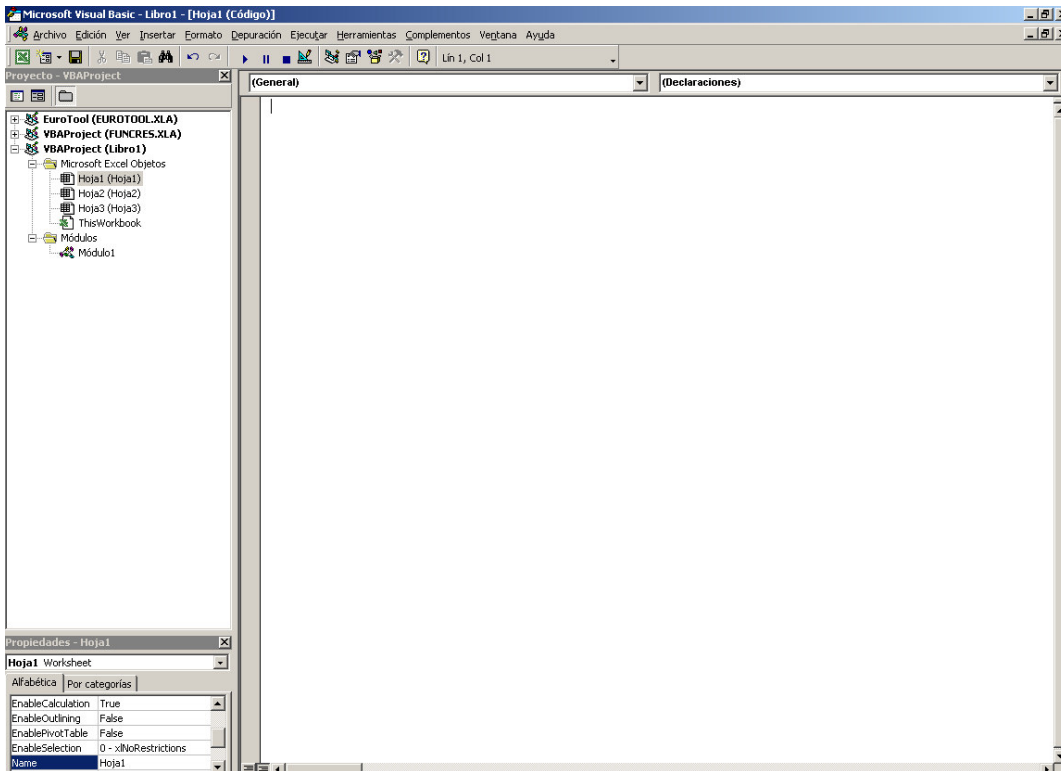
#### 1.- Ver > Explorador de objetos



#### 2.- Ver > Ventana de Propiedades



### 3.- Ver > Código



Ahora ya tienes configurado el entorno del editor de macros

#### 3.4.2.- Ver el código

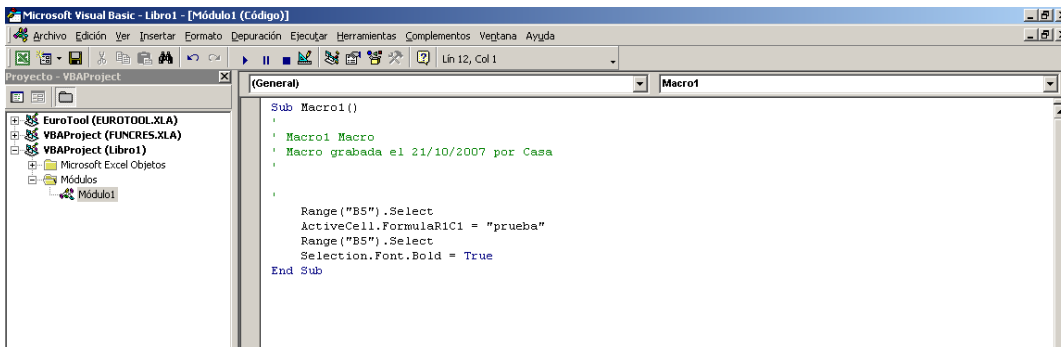
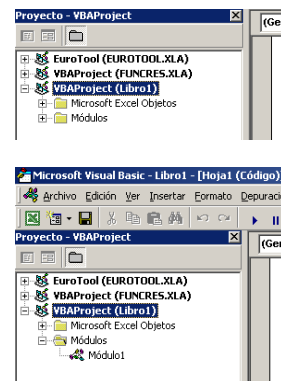
En el explorador de objetos verás que hay varios ítems, uno de ellos se llamará como el libro de trabajo que posees abierto.

Aprieta sobre el signo + y verás dos carpetas

Todas las macros que grabes se añadirán en la carpeta de Módulos

Despliega la carpeta de Módulos y verás que hay una línea que se llama Módulo1

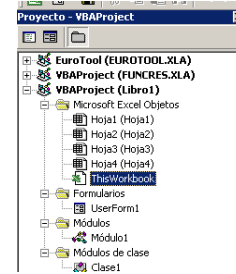
Al hacer doble click sobre Módulo1 verás el código en la derecha de la ventana:



### 3.4.3.- Estructura de Objetos

Un libro de trabajo (fichero de Excel) posee varios tipos de objetos:

- 1.- Carpeta de Microsoft Excel Objetos
  - Hojas de cálculo.
  - Thisworkbook
- 2.- Formularios; que también se pueden crear (ventanas)
- 3.- Módulos.
- 4.- Módulos de clases, se pueden crear objetos .



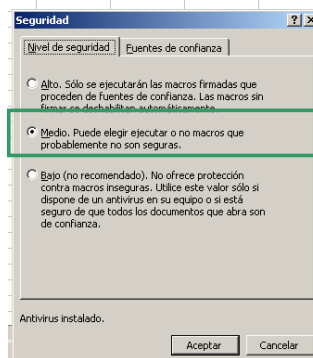
Entre todos estos tipos de objetos, en este manual nos centraremos en las hojas de cálculo, thisworkbook y módulos.

Tanto los formularios como los módulos de clases son herramientas propias de Visual Basic, no exclusivas de Excel; es decir, si lees un manual de Visual Basic que explique como realizar formularios, todo lo que aprendas te servirá para aplicarlo al realizar formularios en Excel

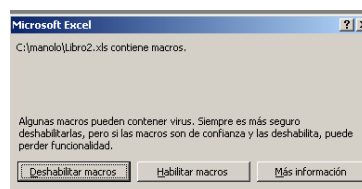
### 3.5.- Nivel de seguridad

Las macros pueden modificar el contenido de nuestro ordenador, con lo que se aconseja estar “protegido” contra archivos que contengan macros y nos lleguen a nuestro ordenador, ya que se puede crear código malintencionado como pueden ser los virus, con este formato. Cuidado.

Desde Herramientas > Macros > Seguridad... se puede configurar el nivel de seguridad; aconsejamos el modo Medio.



Con este nivel de seguridad, al abrir un fichero de Excel con macros, se mostrará la siguiente pantalla, que nos permite Habilitar Macros o Deshabilitar Macros en función del origen de Excel (si te fías o no).



## 4.- Inicio de Programación en Excel

El código que se has creado en tu primera macro era el siguiente:

1. *Sub Macro1()*
2. *'*
3. *' Macro1 Macro*
4. *' Macro grabada el 21/10/2007 por Casa'*
5. *Range("B5").Select*
6. *ActiveCell.FormulaR1C1 = "prueba"*
7. *Range("B5").Select*
8. *Selection.Font.Bold = True*
9. *End Sub*

### 4.1.- Insertar controles

Las macros se ejecutan cuando se produce un evento. Lo más habitual consiste en crear un botón y asignar el evento click sobre dicho botón a una macro. Para hacer esto debes:

Pulsar el botón de Cuadro de Controles



Se mostrará la herramienta de controles; al añadir cualquier control, pasaremos al modo diseño, que se señala (y se quita) con esta opción:



En este caso, seleccionamos la opción de añadir un botón pulsando:



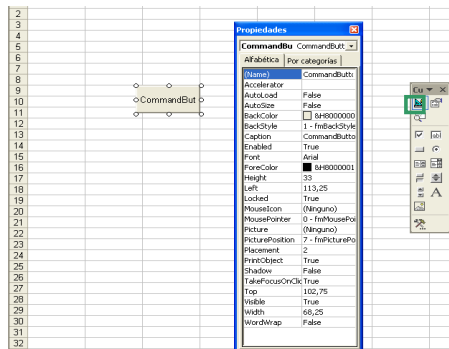
Entonces entraremos en modo diseño; para ver las propiedades del botón (o de cualquier otro control que hubiéramos seleccionado) hay que pulsar dicha opción.



Una vez insertado el botón (y en modo diseño) has de hacer doble clic sobre el mismo, y accederás al editor de Visual Basic, con lo que ya puedes poner el código que desees. En el caso de querer modificar atributos del botón (y/o del resto de controles) has de acceder en Excel (no desde el editor de visual basic), estar en modo diseño, seleccionar el control y pulsar el botón de Propiedades.



En el caso concreto del botón, se mostraría la siguiente pantalla de propiedades



## 4.2.- Acceder a las celdas

Para leer o escribir en una celda de la hoja en la que nos encontramos se utiliza la función Cells (fila, columna)

En el siguiente ejemplo leeremos los datos de la primera columna, hasta encontrar el símbolo \*\*\*, que nos indicará "parar de leer"; la información que se va leyendo se acumula en una variable que después se graba en la celda B1; al aparecer los \*\*\*, se finaliza esta acumulación y el último dato, es el acumulado total.

```

1. Private Sub CommandButton1_Click()
2. Dim fila As Long, acumulado As Double
3.   fila = 1           'indicar la primera fila
4.   acumulado = 0     'valor donde se acumula el resultado
5.   While Cells(fila, 1) <> "***"
6.       'acumular el valor
7.       acumulado = acumulado + Cells(fila, 1)
8.       fila = fila + 1   'incrementar la fila
9.   Wend
10.  cells(1,2)= acumulado 'escribir el resultado en B2(fila=1, columna=2)
11. End Sub

```

Ahora queremos borrar las líneas que no tengan valores numéricos. La primera duda que surge es...¿cómo se borra una fila en Excel?

Existen 2 opciones para resolverlo:

- Hacer un manual con todos los objetos y métodos (manual de más de 200 páginas)
- Enseñarte cómo descubrirlo... en 4 líneas.

Opto por la segunda opción, y los pasos para seguir serán:

- 1.- Pulsa el botón de grabación
- 2.- Borra alguna línea en Excel
- 3.- Para la grabación de la macro
- 4.- Ver el código

En mi ejemplo borré dos líneas, y el código que se genera es el siguiente:

```

1. Sub Macro2()
2. '
3. ' Macro2 Macro
4. ' Macro grabada el 21/10/2007 por Casa
5. '
6. '
7.   Rows("17:17").Select
8.   Selection.Delete Shift:=xlUp
9.   Rows("24:24").Select
10.  Selection.Delete Shift:=xlUp
11. End Sub

```

Después de haber visto es, nuestra macro queda de la siguiente forma:

```

1. Private Sub CommandButton1_Click()
2. Dim fila As Long, acumulado As Double
3.   fila = 1           'indicar la primera fila
4.   acumulado= 0      'valor donde se acumula el resultado
5.   While Cells(fila, 1) <> ""
6.     If IsNumeric(Cells(fila, 1)) And Cells(fila, 1) <> "" Then
7.       'acumular el valor
8.       acumulado = acumulado + Cells(fila, 1)
9.       fila = fila + 1 'incrementar la fila
10.    else 'borrar la fila
11.      Rows(fila & ":" & fila).Select
12.      Selection.Delete Shift:=xlUp
13.    end if
14.  Wend
15.  Cells(1,2)= acumulado 'escribir el resultado en B2(fila=1, columna=2)
16. End Sub

```

#### 4.3.- Objeto Sheet y Workbook

Te habrás fijado que accedes a la celda de la hoja de cálculo del fichero de excel que tienes activo.

La hoja de cálculo se denomina Sheet y el fichero de Excel es el Workbook.

Imagínate que deseas leer la celda A1 de la hoja que se llama "hoja2"; para hacer esto puedes poner:

```
Sheets("hoja2").Cells(1,1)
```

Si el libro de trabajo se llamará "libro2.xls", pondrías:

```
Workbooks("libro2.xls").Sheets("hoja2").Cells(1, 1)
```

Esta instrucción resulta útil cuando se trabaja con varias hojas de cálculo o varios ficheros de Excel.

Atención, ya que algunas funciones sólo se pueden ejecutar cuando la hoja está activa. Por ejemplo, no se puede borrar una línea de una hoja que no esta activa. Para ello habría que utilizar previamente el método select para seleccionar la hoja; es decir:

```
Sheets("hoja2").Select
```

La idea es que Sheet, Worbook y Cells son objetos que tienen propiedades y métodos, algunos compartidos, y otros propios y diferentes.

Si deseas añadir, por ejemplo, una hoja, sería tan sencillo como:

```
Sheets.Add
```

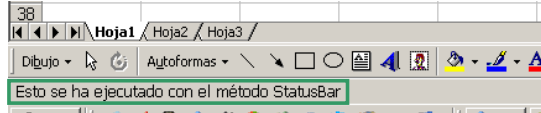
## 4.4.- Objeto Application

### 4.4.1.- StatusBar

El objeto Application posee gran cantidad de métodos y propiedades. Una de las más útiles y menos utilizadas es StatusBar. Con este método podemos escribir en la barra de status, como veremos ahora.

Prueba la siguiente línea de código:

*Application.StatusBar = "Esto se ha ejecutado con el método StatusBar"*



### 4.4.2.- ScreenUpdating

Otra propiedad muy utilizada para mejorar el rendimiento de las macros es ScreenUpdating. Si se pone como false. No actualiza (no refresca) a la vista los datos mientras se van cambiando (por acción de una macro). Hay que acordarse de volver a ponerlo como True antes de que se acabe la ejecución de la macro:

*Application.ScreenUpdating = False*

“macro que requiere mucho rendimiento (que cambia datos, etc...)”

*Application.ScreenUpdating = True*

### 4.4.3.- MousePointer

Esta propiedad (que se puede aplicar a más objetos) sirve para cambiar el formato del cursor.

## 4.5.- Función Active

La función ActiveCell, ActiveSheet y ActiveWorkbook retorna la celda, la hoja o el libro que se tiene activo, prueba el siguiente código:

*MsgBox ActiveSheet.Name*

## 4.6.- Escribir Fórmulas

Prueba de hacer una suma y mira la macro de código que te genera:

*ActiveCell.FormulaR1C1 = "=SUM(R[-4]C[-4]:R[2]C[-4])"*

Viendo el código (que raro ¿eh?) deducimos lo siguiente:

1. El nombre de la función está traducido al inglés; aunque hayamos puesto “suma” el sistema lo traduce automáticamente.
2. ¡Atención! Al escribir la fórmula lo primero que se pone es el =
3. Esta manera de escribir la formula es relativa a donde se ha puesto la celda. En este caso, lo que había puesto en la celda E5 es lo siguiente:

=SUMA(A1:A7). Para entender un poco más la formula R[-4]C[-4]:R[2]C[-4] te doy una pista; R viene de Rows (filas) y C de Columns (columnas) y recuerdo que lo he escrito tiene su origen en la celda E5.

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					

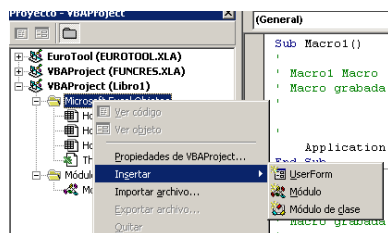
Entiendo que esto es algo complicado; yo prefiero escribir lo siguiente:

*ActiveCell.Formula = "=SUM(A1:A7)"*

Fíjate bien, que en este otro caso no he utilizado la propiedad **FormulaR1C1**, sino la propiedad **Formula**.

#### 4.7.- Insertar Objetos

Para crear un nuevo módulo, formulario o módulo de clase se debe pulsar, dentro del editor de macros, el botón derecho sobre el explorador de proyectos e ir a la opción de Insertar.



#### 4.8.- Mejorar rendimiento

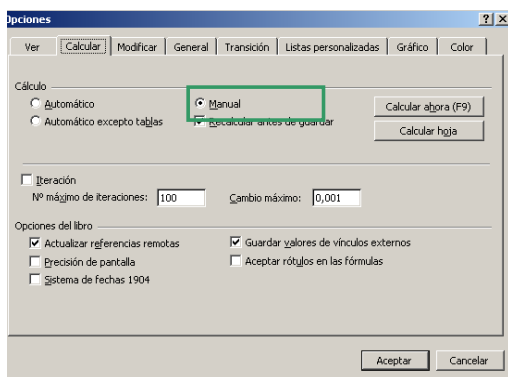
Puede darse el caso de que su macro cambie muchos datos y que estos datos afecten a fórmulas. Si posees una gran cantidad de datos, puede ser que se ralentice tu sistema. Para evitarlo, existen varios trucos:

1.- Desactivar el cálculo automático (equivale a: Herramientas > Opciones..., pestaña Calcular y marcar Manual)

El código asociado es el siguiente:  
*Application.Calculation = xlManual*

Recuerda que una vez acabada la macro, debes volver a marcar como automático (si no, no se refrescaría el resultado final correctamente).

2.- La instrucción DoEvents retorna el control al ordenador. Imagínate que haces una macro donde, dentro de un



bucle, se consumen muchos recursos (o tiempo), y por este motivo el controlador de tareas no permite cambiar de aplicación. Si dentro del bucle pones la instrucción DoEvents, esto te permitirá continuar trabajando en otras ventanas.

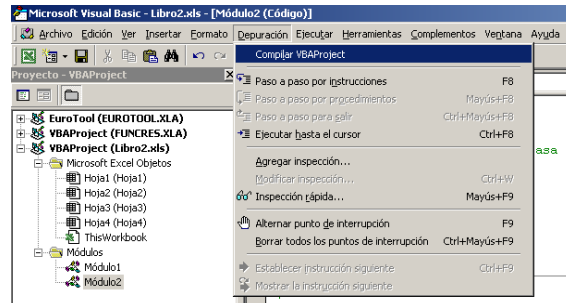
3.- ScreenUpdating (¿lo recuerdas? ¿no?); pon esta propiedad de Application a False cuando vayas a ejecutar una macro que consuma muchos recursos.

## 5.-Detalles técnicos

### 5.1.- Compilar

Dentro del editor de Visual Basic tienes la opción de Depuración>Compilar VBAProject

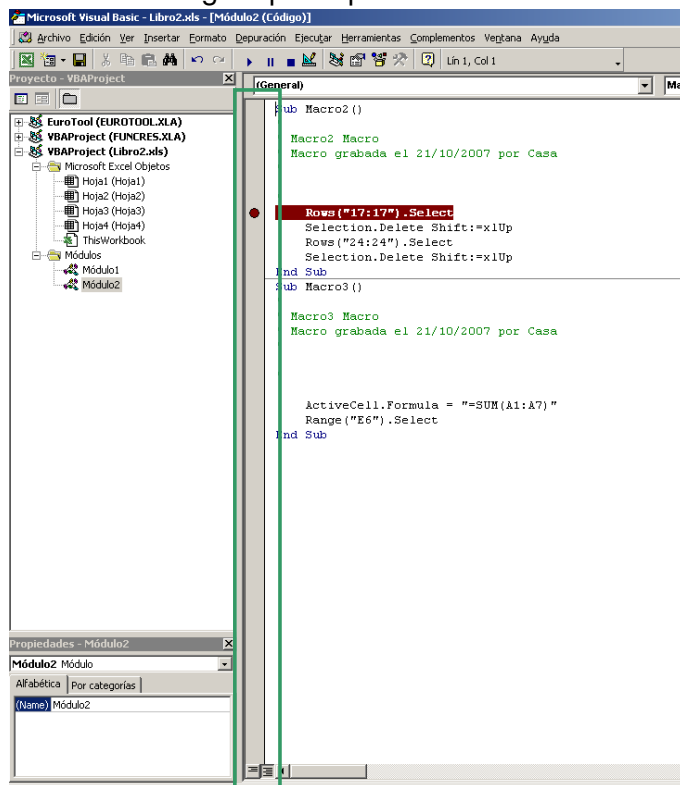
Esta compilación NO generará ningún tipo de fichero ejecutable, simplemente detectará los posibles errores sintácticos que haya en el código.



### 5.2.- Depurar y Breakpoints

La opción de depurar se refiere a ejecutar el código "paso a paso", para poder detectar errores de programación. Desde el menú de Depuración existen varias opciones; como por ejemplo el breakpoint.

Para insertar un breakpoint o punto de parada tenemos que hacer un click sobre la barra gris que separa las ventanas del código



Los breakpoints se marcan con una redonda de color rojo en la barra gris y con la línea de código resaltada en el mismo color rojo.