

# MANUAL DE PROGRAMACIÓN EXCEL

---

*por Roger Perez - Aplica Excel Contable*

*(para versiones Excel del 2007 al 2013 inclusive)*

*Copyright © 2.009-2.013 Roger Perez (AEC)*

*El presente manual permitirá al usuario iniciarse en la programación VBA para Excel.*

# Tabla de Contenidos

	0
<b>Cap. I 1 - Introducción</b>	<b>7</b>
1 Bienvenid@s .....	7
2 Cómo utilizar este manual.....	8
<b>Cap. II 2- Cómo programar en Excel?</b>	<b>12</b>
1 Qué es una macro?.....	12
2 Breve introducción al Editor.....	15
3 Cómo crear una macro.....	15
4 Uso de la grabadora de macros.....	16
5 Privada o Pública .....	18
<b>Cap. III 3 - Referencias a objetos Excel</b>	<b>20</b>
1 Nombrando Libros y hojas.....	20
2 Nombrando Celdas y Rangos .....	20
3 Nombrando Objetos.....	21
<b>Cap. IV 4 - Eventos, Metodos y Propiedades</b>	<b>24</b>
1 Eventos .....	24
2 Eventos de Libros.....	25
3 Eventos de Hojas.....	26
4 Macro al cambio en celdas.....	26
5 Macro al seleccionar celda.....	27
6 Metodos .....	28
7 Propiedades.....	28
<b>Cap. V 5 - Ejecutando macros</b>	<b>30</b>
1 Dónde colocar las macros .....	30
2 Cómo ejecutar una macro.....	31
3 Ejecutar macro desde un botón .....	32
4 Ejecutar macro con atajo de teclado.....	33
5 Acerca de las macros Auto-Open.....	34
6 Acerca de las macros Auto-Close .....	35
7 Cómo probar una macro .....	36
8 Cómo controlar la ejecución de una macro .....	37
9 Conocer el valor que toman las variables.....	38
10 Cómo evitar que una instrucción se ejecute.....	39

11	Acceder a la Ayuda desde una línea de código.....	39
12	Salir de una rutina.....	40
<b>Cap. VI</b>	<b>6 - Seguridad en el proyecto</b>	<b>42</b>
1	Cómo proteger un proyecto.....	42
2	Evitar que las macros se vean desde el menú.....	43
3	Cómo interrumpir una macro.....	43
4	Habilitar o no las macros .....	44
<b>Cap. VII</b>	<b>7- Tratamiento de Variables</b>	<b>47</b>
1	Tipo de Variables .....	47
2	Duración de las variables.....	47
3	Determinar el tipo de variable.....	48
4	Convirtiendo variables .....	49
5	Limpiando variables .....	50
<b>Cap. VIII</b>	<b>8 - Trabajando con Cadenas</b>	<b>52</b>
1	Extraer partes de una cadena.....	52
2	Armando cadenas.....	53
3	Obtener el largo de una cadena.....	53
4	Introducir caracteres especiales .....	54
5	Detectar o encontrar texto en una cadena .....	55
6	Creando cadenas de largo fijo.....	55
7	Obtener la parte numerica de una cadena .....	56
<b>Cap. IX</b>	<b>9 - Trabajando con Libros</b>	<b>58</b>
1	Principales Metodos y Propiedades de Libros .....	58
2	Abrir un libro. Abrir libro con clave.....	58
3	Ejecutar macro al abrir un libro.....	59
4	Al abrir libro incrementar un contador .....	60
5	Seleccionar un libro.....	60
6	Activar otro libro distinto al actual.....	60
7	Obtener la ruta de un libro.....	61
8	Guardando Libros.....	61
	Guardar el libro activo.....	61
	Guardar un libro con otro nombre o formato.....	61
	Guardar un libro con clave .....	62
	Guardar un libro cuyo nombre será el valor de una variable.....	63
	Guardar un libro cuyo nombre serán datos concatenados.....	63
9	Cerrando Libros .....	64
	Cerrar todos los libros en uso.....	64
	Cerrar un solo libro .....	64
	Cerrar un libro SIN guardar los cambios.....	64

Cerrar un libro guardando los cambios .....	65
---	----

## Cap. X 10 - Trabajando con Hojas 67

1 Características del tratamiento de Hojas .....	67
2 Principales Metodos y Propiedades de Hojas .....	67
3 Activar o seleccionar otras hojas distintas a la actual .....	68
4 Seleccionar la hoja anterior o posterior a la activa .....	68
5 Devolver el nombre de la hoja en una variable .....	69
6 Seleccionar todas las hojas de un libro .....	69
7 Proteger una hoja .....	70
8 Desproteger una hoja .....	71
9 Vista previa de la hoja activa y de otras hojas .....	71
10 Imprimir hojas .....	71
11 Insertar hojas .....	72
12 Eliminar hojas .....	73
13 Copiar hojas .....	73
14 Ocultar hojas .....	74
15 Mostrar hoja oculta .....	75
16 Establecer area visible de una hoja .....	75

## Cap. XI 11 - Trabajando con Celdas y Rangos 77

1 General .....	77
2 Principales Metodos y Propiedades de Rangos .....	77
3 Selección de Celdas o Rangos .....	78
4 Selección de rango utilizando variables .....	79
5 Seleccionar celdas a cierta distancia de la celda activa .....	80
6 Seleccionar la región donde se encuentra la celda activa .....	80
7 Seleccionar hasta la última celda vacía - Fin de rango .....	81
8 Obtener primer fila libre .....	81
9 Obtener ultima columna con datos .....	82
10 Obtener la dirección de una celda y guardarla en variable .....	82
11 Borrar o Limpiar celdas o rangos .....	83
12 Eliminar celdas o rangos .....	83
13 Eliminar varias filas segun condicion .....	85
14 Insertar Filas .....	85
15 Eliminar Filas .....	85
16 Ocultar filas .....	86
17 Mostrar filas .....	86
18 Insertar Columnas .....	87
19 Eliminar columnas .....	87

20	Ocultar columnas.....	88
21	Mostrar Columnas.....	88
22	Capturar fecha y hora de carga de datos .....	89
23	Ordenar un rango.....	89
24	Detectar si la celda contiene formula.....	90
25	Resaltar la fila activa.....	90
26	Cambiar color de fuente a celdas con datos.....	91

## Cap. XII 12 - Bucles= Comandos Especiales 93

1	General .....	93
2	For Each .... Next.....	93
3	For ..... Next.....	94
4	While .... Wend.....	95
5	If.... Elself....Else.....	96
6	Do While .... Loop.....	97
7	Do Until.... Loop.....	97
8	Uso de SET .....	98
9	With....End With.....	99
10	Uso de Select Case.....	100

## Cap. XIII 13- Trabajando con fórmulas 103

1	Trabajando con fórmulas.....	103
2	Introducir fórmulas en celdas.....	103

## Cap. XIV 14 - Trabajando con Objetos Insertados en Hoja 105

1	Comentarios generales .....	105
2	Asignar rango a un Combobox.....	106
3	Enviar valor de un Combo a una celda.....	107
4	Enviar valores de Combo de 4 columnas a celdas.....	107
5	Buscar valor del Combo en base Devolver otros datos en textbox .....	108
6	Limpiar un combo.....	108
7	Funciones de validación .....	109
8	Asignar formato moneda a un TextBox.....	110
9	Validar campos fecha en Textbox.....	110
10	Validar campos numéricos en Textbox.....	111

## Cap. XV 15 - Controlando Mensajes de Excel 113

1	General .....	113
2	No mostrar avisos de alerta.....	113
3	No mostrar aviso, al guardar un archivo, de que el archivo ya existe:.....	114

4	No mostrar la ejecución de la macro o el movimiento de hojas	114
<b>Cap. XVI</b>	<b>16 - Controlando Errores</b>	<b>116</b>
1	On Error Resume Next	116
2	On Error GoTo	116
3	On Error GoTo 0	117
<b>Cap. XVII</b>	<b>17 - Uso de MsgBox e InputBox</b>	<b>119</b>
1	Construcción de MsgBox	119
2	Construcción de InputBox	120
3	Controlar que se ha ingresado un valor	121
<b>Cap. XVIII</b>	<b>18- Controlando Barras y Menú de Excel</b>	<b>124</b>
1	Ocultar Barras de Desplazamiento	124
2	Ocultar Pestañas de hoja	124
3	Ocultar Encabezados de filas y col.	125
4	Ocultar Líneas de División	125
5	Quitar barras frecuentes	125
6	Aclaración General	126
<b>Cap. XIX</b>	<b>19- Buscando-Comparando-Evaluando Datos</b>	<b>129</b>
1	Devolver en una celda el resultado de una búsqueda	129
2	Buscar un dato. Copiar la fila de todos los registros encontrados	130
3	Buscar cierto dato en un rango. Si se encuentra borrar la fila que lo contiene	131
4	Comparando cadenas	133
5	Extraer la parte numérica de una cadena	133
6	Eliminar filas si las celdas de cierta columna están vacías	134
7	Rellenar celdas vacías de un rango con cierto valor	134
<b>Cap. XX</b>	<b>20- Copiando Datos</b>	<b>136</b>
1	Copiar rango de datos de una hoja a la siguiente	136
2	Copiar cierta fila en otro libro. Conocer última fila con datos	136
3	Copiar solo valores. Pegado Especial	137
4	Copiar formato. Pegado Especial	137
5	Crear libro como copia de hoja	138



# Capítulo

A large gray circle with a white vertical bar in the center, resembling a stylized letter 'I' or a play button.

I



# 1 1 - Introducción

## 1.1 Bienvenid@s

### Manual: Programando MACROS en Excel

para versiones Excel: 2007  
al 2013 inclusive

Autor: Roger Perez  
(AEC)

La intención de este Manual es **guiar** al usuario de Excel a potenciar las planillas de cálculo con programación, y **prepararlo** para que, al final del estudio de este manual, sea un experto en **Programación en Excel**.

Los que nunca han trabajado con algún lenguaje de programación verán que **muy fácilmente** podrán 'personalizar' sus libros **adaptando o creando rutinas** con código **VBA** (Visual Basic para Aplicaciones)

El manual está dirigido a:

- 1-** los que **nunca han programado** (los primeros 7 capítulos en detalle y con imágenes contienen toda la teoría necesaria)
- 2-** aquellos que solo han **copiado rutinas** desconociendo el significado de cada instrucción (cada línea se explica en español)
- 3-** a los que **recorren foros** en busca de una rutina que luego, sin detalles, no les es posible adaptar a sus libros ( con *Notas* aclaratorias para poder adaptar las rutinas a otras situaciones)
- 4-** los que **ya han programado en VBA** (nuevos estilos en la programación, más, lo que ha cambiado en Excel 2007 - 2013)

**NOTA:** Las rutinas contenidas en este manual, fueron desarrolladas y probadas en las siguientes versiones: Office 2003, Office 2007, Office 2010 y Office 2013.



## 1.2 Cómo utilizar este manual

### ¿Cómo aprender a programar en Excel?

Como todo estudio, requiere del alumno un **seguimiento ordenado del manual**. De esa manera, al llegar a los capítulos más avanzados, esté en conocimiento del significado de los términos allí utilizados.

En el cap. siguiente un par de temas, **con imágenes**, para familiarizarnos con el entorno **Editor de Macros**.

Luego un par de capítulos detallando **conceptos** que serán utilizados a lo largo del manual:

**Referencias, Eventos, Métodos, Propiedades, Variables, Cadenas** y otros.

Otro **capítulo fundamental**: dónde colocar una macro, cómo ejecutarla, cómo hacer un seguimiento, uso de la Ayuda entre otros tems.

También un capítulo dedicado a la **seguridad** del proyecto.

Y a partir del capítulo 9 las **rutinas** de ejemplo ordenadas por temas: **Libros - Hojas - Celdas o rangos - Bucles - Fórmulas - Objetos insertados - Control de Mensajes - Control de Errores - Control de Barras y Menu - Uso de MsgBox e InputBox - Búsqueda, comparación y copiado** de datos.

Así, partiendo de las principales instrucciones básicas, y luego de **más de 100 rutinas**, se llegará a un gran dominio en programación de Macros para Excel.

Cada línea, cada comando, se encuentra explicado en español para que cada usuario pueda adaptar esa instrucción a su libro, hoja o referencia.

En muchos casos además, con **imágenes** que guiarán y mostrarán los resultados obtenidos una vez ejecutada la macro.



# Capítulo

---





## 2 2- Cómo programar en Excel?

### 2.1 Qué es una macro?

Al hablar de Excel, no podemos dejar de mencionar las '**macros**'.

Una **macro** o '**rutina**' es un conjunto de instrucciones en lenguaje de programación, que en el caso de Excel se conoce como **VBA** (Visual Basic for Applications).

Estas instrucciones nos permiten realizar ciertas tareas, que por ser repetitivas, nos valemos de una rutina para automatizarlas.

#### **Ejemplos:**

al abrir un libro, se incrementa un contador.

al abrir un libro se ocultan hojas o algunas barras

al ingresar datos en una columna, se completa el resto del registro con datos de otra hoja,

copiar datos de una hoja en otra, o en otro libro.

al cerrar un libro, se muestran barras que previamente fueron ocultas.

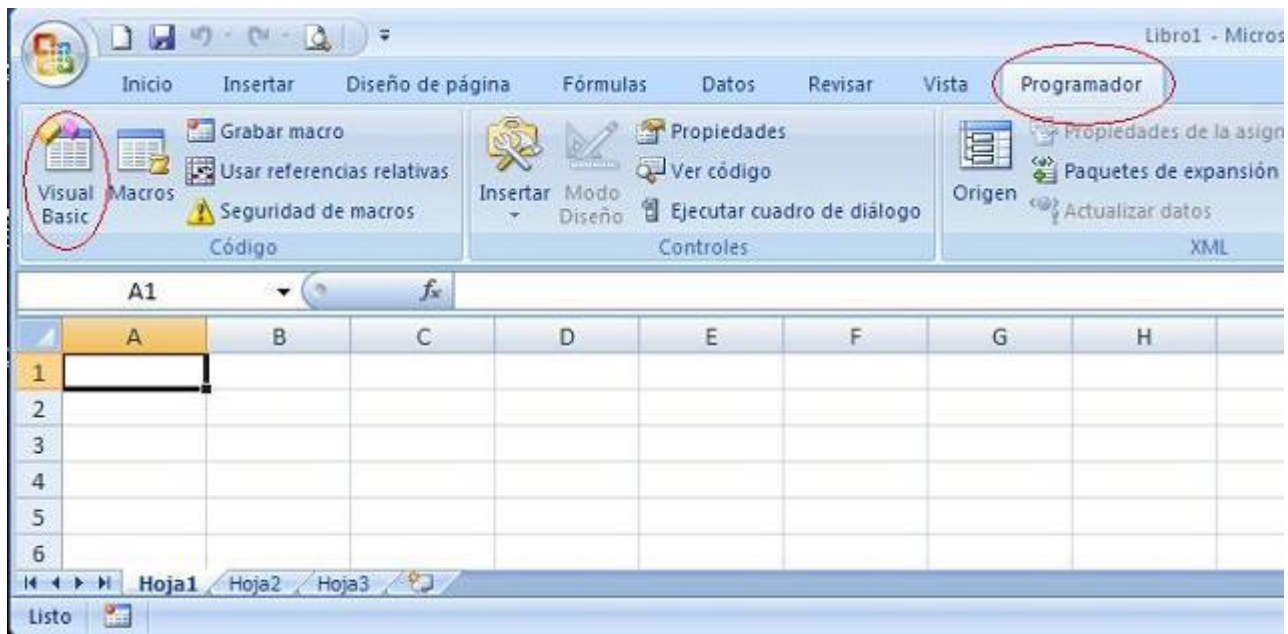
Para los que recién se inician en este tema, diré que para crear estas rutinas o 'macros' básicamente se necesitan los siguientes **elementos**:

1- un **espacio** donde escribir las instrucciones o rutinas que harán esas tareas: el **Editor de Macros** (lo veremos ampliamente en el capítulo siguiente).

A este espacio se accede, en versiones anteriores al 2007, desde menú **Herramientas, Macros, Editor** o con el atajo de teclado **Alt+F11**.

Una vez en ese 'espacio', podemos insertar módulos (donde escribiremos algunas rutinas) , formularios personales o Userforms, o escribir rutinas destinadas a cada objeto: hojas o libro.

**La siguiente imagen corresponde a la versión Excel 2007 (\* Ver Nota), donde se mantiene el mismo atajo de teclado (Alt+F11)**

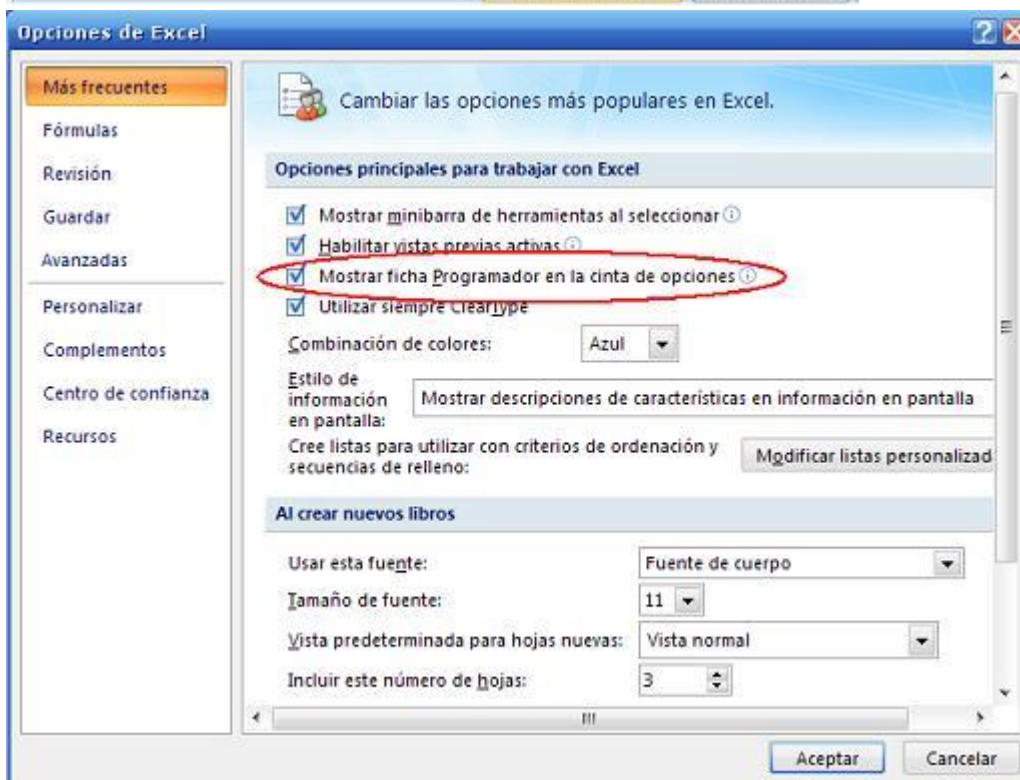
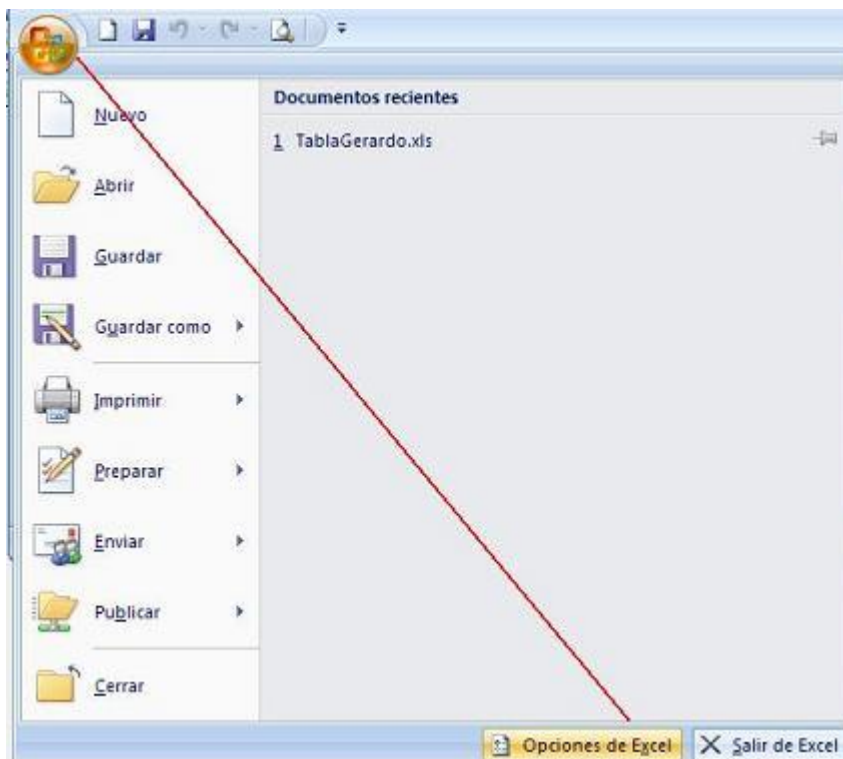


2- una **acción** que hará que la tarea programada se ejecute. A esto llamamos '**Eventos**' que inician una macro y los más habituales son:  
 abrir o cerrar un libro, entrar o salir de una hoja, cambios en celdas, selección de celdas, antes de imprimir o guardar, el 'clic' en un botón de comando, al presionar un atajo de teclado, y otros más.

3- un **lenguaje de programación**. En Excel utilizamos **VBA** (Visual Basic para Aplicaciones)

4- Ocasionalmente un **formulario** donde trabajar para luego volcar los resultados en las hojas: llamados **Userforms**.

**Nota (\*)**: Si la ficha 'Programador' no aparece en la cinta de opciones, presionar el **botón de Office, Opciones de Excel**. En la ventana abierta, seleccionar del margen izquierdo la opción '**Más frecuentes**' y tildar la opción: '**Mostrar ficha Programador en cinta de opciones**'.



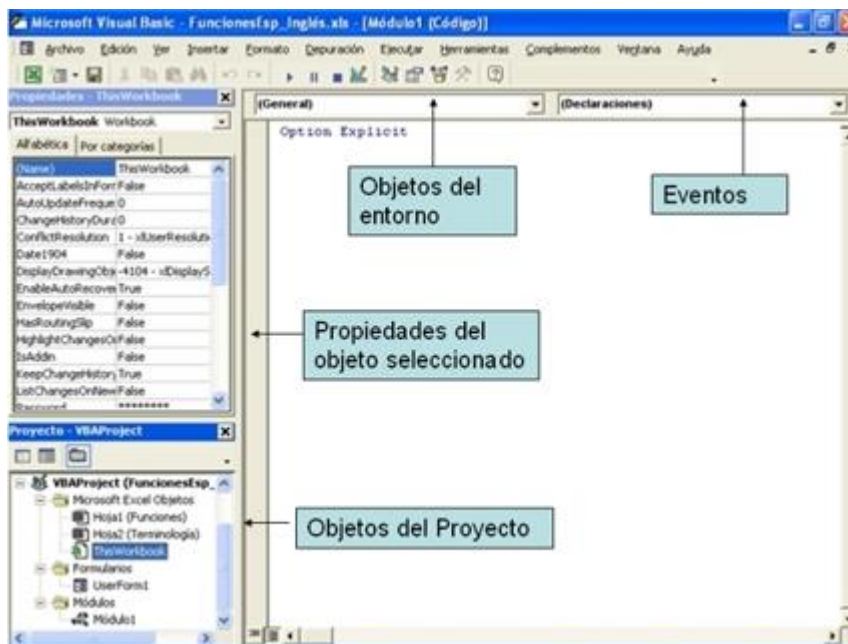
## 2.2 Breve introducción al Editor

### **El Editor de Macros:**

Con el atajo de teclado **Alt+F11** tenemos acceso a una nueva ventana, diferente a las hojas habituales de Excel.

Aquí en este espacio, trabajaremos con nuestras rutinas, programando nuestro libro o aplicación.

Veamos qué encontramos en la ventana del Editor:



Con **Alt+Q** regresamos a la hoja Excel. Cerramos el libro guardando los cambios. Cuando lo abramos nuevamente, por lo general se nos mostrará una ventana solicitando autorización para **'habilitar las macros'**.

Veremos en el tema **Seguridad** las distintas opciones al momento de abrir un **libro con macros**.

## 2.3 Cómo crear una macro

Hay 2 maneras de **crear una macro**:

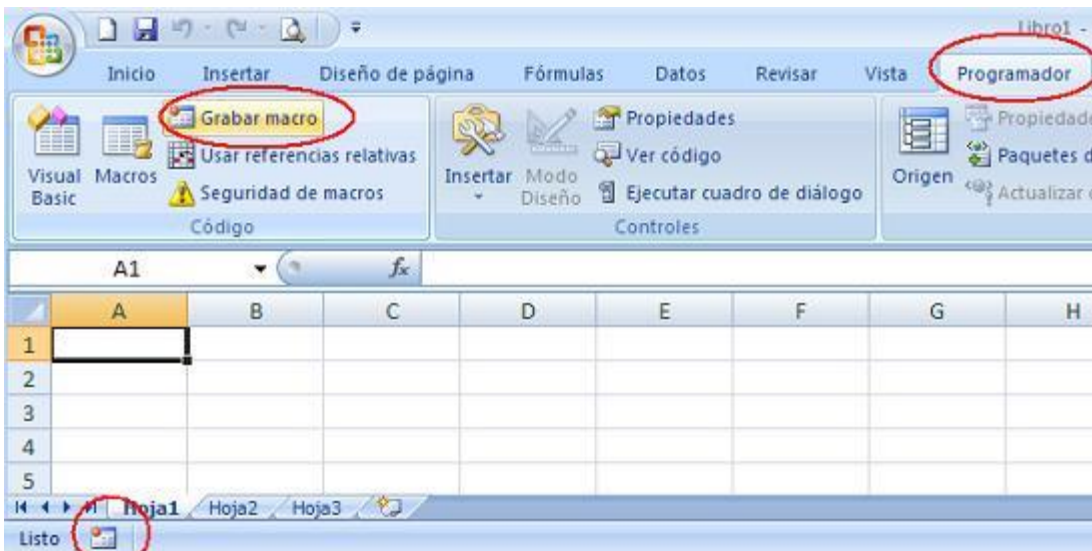
### **a) Utilizando la Grabadora de Macros.**

Desde la hoja Excel, en el menú **Herramientas, Macros, Grabar macro**.

En versión 2007, presionar el botón que se encuentra en la barra de estado (en la imagen, debajo de la pestaña Hoja1).

El mismo botón será para detener la grabadora.

También desde la ficha Programador se accede a la opción 'Grabar Macro'



**b) Escribiendo el código en el Editor:** El atajo para llegar al Editor son las teclas **Alt+F11**.

En temas siguientes veremos dónde colocar el código, dentro de este espacio, según la tarea que estemos por automatizar.

A partir del capítulo 8 se presentan más de 100 rutinas listas para copiar y pegar en este espacio.

## 2.4 Uso de la grabadora de macros

### Utilizando la grabadora de macros:

Al activar la grabadora, ya visto en el tema anterior, veremos la siguiente ventana:



**Nombre de macro:** utilizar nombres que nos ayuden a recordar qué tarea realiza la macro.

**Método abreviado:** ingresar una tecla o mayúscula tecla (\* Ver tema **Atajo de Teclado**)

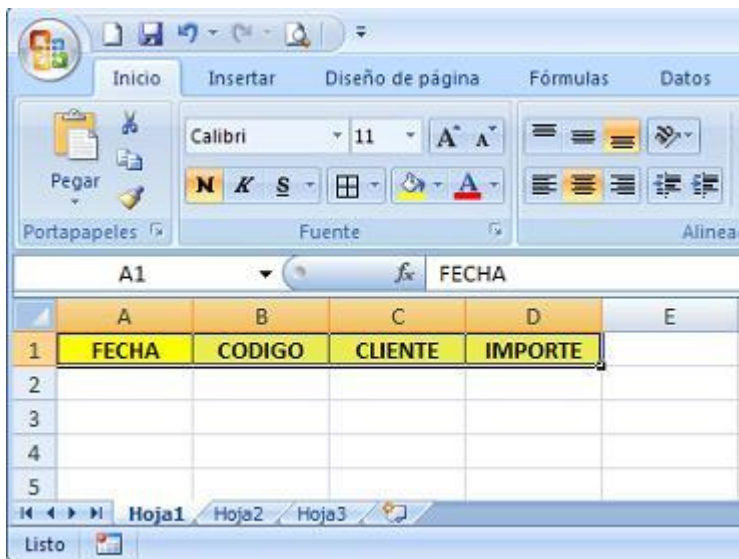
**Guardar macro en:** Si optamos por *Este libro* será de ejecución en el libro activo, si optamos por *Libro Personal* será de ejecución en todos los libros Excel.

**Descripción:** es recomendable ingresar algún detalle de lo que realiza la macro.

Al aceptar estamos ya en condiciones de ir generando el código. Para eso realizaremos todos los pasos que necesitamos automatizar y luego la detendremos.

### Ejercicio:

Abrir un libro nuevo de 3 hojas. Escribir y formatear las celdas de títulos según esta imagen:



**Activar la grabadora** (desde la barra o desde el botón en barra de estado, para versión 2007)

**Copiar los títulos** de la Hoja1 en las otras 2 hojas del libro, con el mismo texto y formato.

**Detener la grabadora** (desde la barra o desde el botón en barra de estado, para versión 2007)

Entramos al **Editor** (con Alt+F11) y allí veremos, en un **módulo**, la rutina "grabada". Todavía no explicaré el significado de cada línea... la analizaremos un poco más adelante, además de aprender cómo ejecutarla.

```
Sub Macro1()
'
' Macro1 Macro
'
    Range("A1:D1").Select
    Selection.Copy
    Sheets("Hoja2").Select
    Range("A1").Select
    ActiveSheet.Paste
    Sheets("Hoja3").Select
    ActiveSheet.Paste
    Sheets("Hoja1").Select
    Application.CutCopyMode = False
End Sub
```

## 2.5 Privada o Pública

Las rutinas que se colocan en un módulo, pueden ser **Públicas** o **Privadas**.

### Procedimientos Públicos:

Son aquellos que pueden ser accedidos desde cualquier otro módulo del proyecto. Empiezan con la frase **Public Sub** o directamente **Sub**

```
Sub miMacro()  
'instrucciones  
End Sub
```

### Procedimientos Privados:

Son aquellos que solo pueden ser accedidos desde el mismo módulo y se define como:

```
Private Sub miMacro()  
'instrucciones  
End Sub
```

También podemos definir **procedimientos estáticos**. Estos se utilizan cuando sus variables (\*) se conservan una vez completada su ejecución o finalizado el procedimiento. Se definen como:

```
Static Sub miMacro(variable1, variable2)  
'instrucciones  
End Sub
```

**Nota:** El tema **Variables** se encuentra desarrollado en el [cap. 7](#)

Para llamar a una subrutina o procedimiento, se utiliza la instrucción: **CALL (cuando es necesario pasarle los argumentos)** o directamente se ingresa el nombre de la subrutina.

### Ejemplo 1:

```
Sub miMacro()  
'instrucciones que asignan un valor a la variable 'valor1'  
valor1 = ....  
Call miMacro_Nva(valor1)  
End Sub
```

### Ejemplo 2:

```
Su miMacro()  
'instrucciones  
miMacro_Nva 'o también: Call miMacro_Nva (sin argumentos) (*)  
End Sub
```

**Atención:** Nótese que utilizando la instrucción **CALL** es más fácil de observar el llamado a subrutinas.



# Capítulo

---



## 3 3 - Referencias a objetos Excel

### 3.1 Nombrando Libros y hojas

Antes de avanzar con la ejecución de macros o rutinas, debemos saber cómo hacer referencia a los objetos de Excel en código VBA.

#### WORKBOOK: Libro de trabajo

ActiveWorkbook : Libro activo

Workbooks("Libro1.xls") : Llamada al libro de nombre Libro1  
**'ajustar la extensión tratándose de versión Excel 2007'**

Workbooks(2) : El segundo libro abierto

#### WORKSHEET: Hoja de trabajo

ActiveSheet : Hoja activa

Sheets("Enero") : Hoja de nombre 'Enero'

Sheets(3) : Número de hoja del libro según el *orden de las pestañas*

### 3.2 Nombrando Celdas y Rangos

La expresión '**Range**' sirve tanto para hacer referencia a **una celda** como a un conjunto o **rango de celdas**.

La palabra '**Cell**' indica una celda

#### RANGE : rango o celda

Range("A2") : la celda A2

Range("A5:B10"): el rango comprendido desde A5 hasta B10

Range("E:E") : columna E

Range("2:2") : fila 2

Range("A2,B5,C2:D10,F12") : se mencionan rangos discontinuos

#### CELL : celda

Activecell : la celda activa

Cells(2,1) : la celda A2 .

**Nota:** Observar que mientras en Range se introduce la celda en el orden Col,Fila, en Cells es a la inversa: **Cells(fila,col)**

**SELECTION: rango o celda seleccionadas al momento de ejecutar la rutina**

*Msgbox Selection.Address* 'nos devolverá la referencia de la selección, por ej: \$A\$2:\$B\$5

*Selection.Font.ColorIndex = 3* 'colorea el texto de las celdas seleccionadas de color rojo

**TARGET: celda**

Esta expresión la veremos en rutinas que evalúan los cambios o la selección de celdas.

En el **cap 4: Eventos**, *Evaluar cambios en celdas*, se explica el tema detalladamente.

**Ejemplo:**

*If Target.Address(false, false) = "A5" then*

Con esta instrucción estamos consultando si la celda activa corresponde a A5.

Si omitimos los argumentos '**false**' tanto para fila como columna, nos evaluará una referencia 'absoluta':

*If Target.Address = "\$A\$5"*

### 3.3 Nombrando Objetos

Los **objetos** insertados en una hoja Excel pueden ser:

- 1- de Dibujo
- 2- Gráfico
- 3- Imagen
- 4- Controles dibujados con la barra 'Cuadro de controles'
- 5- Controles dibujados con la barra 'Formularios'
- 6- Userforms (formularios desarrollados desde el Editor)

En general los objetos se denominan **Shapes** y sus nombres aparecen en el '**Cuadro de Nombres**', delante de la **barra de Fórmulas**.

**El número dependerá del nro de objetos insertados en la hoja.**



### Ejemplos:

`ActiveSheet.Shapes("5 CuadroTexto").Select` 'selecciona un cuadro de texto

`ActiveSheet.ChartObjects("3 Gráfico").Activate` 'selecciona un gráfico

`ActiveSheet.Shapes("2 Imagen").Select` 'selecciona una imagen

`ActiveSheet.Shapes("Combobox1").Select` o `ActiveSheet.Combobox1.Select`  
'selecciona un control del Cuadro de controles

`ActiveSheet.Shapes("List Box 1").Select` 'selecciona un control de la barra  
Formularios

`Userform1.Show` 'llamada a un formulario de nombre  
`Userform1`

**Nota:** En el **cap 14: Trabajando con Objetos** ..... continúa este tema



# Capítulo

# IV

## 4 - Eventos, Metodos y Propiedades

### 4.1 Eventos

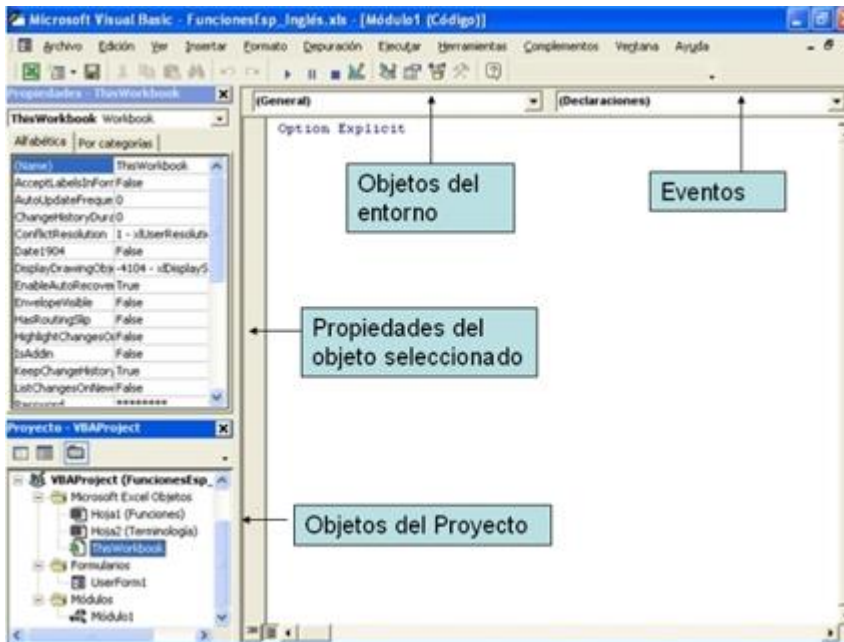
Hemos visto en el capítulo 2, tema 'Qué es una macro', que se necesitaba en principio de un espacio (el Editor) y una acción o **evento** que ejecutara una macro.

¿Pero qué es un **evento**? es una **acción que el usuario ejecuta**

Los **eventos** más comunes tienen que ver con los objetos **Libro** y **Hojas**, como ser: *abrir, guardar, imprimir o cerrar libros, activar o desactivar hojas*. Estas rutinas no se ejecutan manualmente, como las que colocamos en módulos, sino que se ejecutan al producirse el evento que las convoca.

Otros eventos pueden ser: *el click de un botón, al presionar un atajo de teclado* y otros que ya veremos en el curso de este manual.

Si volvemos a la imagen que se encuentra en el capítulo 2, **Breve introducción al Editor**, veremos los 2 cuadros desplegables: **Objetos y Eventos**.



Debemos seleccionar un **objeto** y luego optar por un **evento**. A continuación la primera y última instrucción de la macro se escribirán.

#### Ejemplos:

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)
'instrucciones que se ejecutarán antes de guardar el libro
End Sub
```

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
'instrucciones que se ejecutarán cada vez que se seleccione una celda
End Sub
```

**Atención:** también los **controles** colocados tanto en una hoja como en un Userform, tienen su lista de eventos, como **SetFocus** (al recibir el enfoque), **LostFocus** (al perder el enfoque) y otros que veremos en los ejemplos del cap 14: [Trabajando con Objetos...](#).

## 4.2 Eventos de Libros

A continuación los **Eventos de Libros**:

<b>Activate:</b>	se activa el libro
<b>AddInstall:</b>	ocurre al instalar el libro como complemento
<b>AddUninstall:</b>	ocurre al desinstalar el libro como complemento
<b>BeforeClose:</b>	antes de cerrar el libro
<b>BeforePrint:</b>	antes de imprimir
<b>BeforeSave:</b>	antes de guardar el libro
<b>Deactivate:</b>	al desactivar el libro
<b>NewSheet:</b>	al insertar nueva hoja
<b>Open:</b>	al abrir el libro
<b>SheetActivate:</b>	al activar una hoja
<b>SheetCalculate:</b>	al efectuar cálculos en hoja
<b>SheetChange:</b>	al efectuar cambios en hoja
<b>SheetDeactivate:</b>	al desactivar hoja
<b>SheetFollowHyperlink:</b>	al clic en un hipervínculo
<b>SheetPivotTableUpdate:</b>	ocurre después de actualizar la hoja del informe de tabla dinámica
<b>SheetSelectionChange:</b>	se produce cuando la selección cambia en una hoja
<b>WindowActivate:</b>	cuando se activa cualquier ventana del libro
<b>WindowDeactivate:</b>	cuando se desactiva cualquier ventana del libro
<b>WindowResize:</b>	cuando se cambia de tamaño cualquier ventana del libro
<b>PivotTableCloseConnection:</b>	Ocurre después de que un informe de tabla dinámica cierra la conexión con su origen de datos.
<b>PivotTableOpenConnection:</b>	Ocurre después de que un informe de tabla dinámica abre la conexión con su origen de datos
<b>SheetBeforeDoubleClick:</b>	Ocurre al hacer doble clic en una hoja de cálculo, antes de la acción predeterminada para el doble clic
<b>SheetBeforeRightClick:</b>	Ocurre al hacer clic con el botón secundario del <i>mouse</i> (ratón) en una hoja de cálculo, antes de la acción predeterminada.

### Nota:

Las rutinas colocadas en los **eventos** del libro (**Private Sub Workbook**) que hacen mención a hojas (**Sheet....**) se ejecutan para **todas** las hojas del libro.

## 4.3 Eventos de Hojas

### Principales **Eventos** de **Hojas**:

A diferencia de los últimos eventos de la lista anterior, un **evento** de **Hoja** solo se ejecuta para el objeto **Hoja** donde se encuentre.

Es decir, si seleccionamos de la lista de objetos del panel de la izquierda el objeto 'Hoja2', la macro que escribamos solo se ejecutará para esta hoja 2.

Los **eventos** de **Hojas** son:

**Activate:** al activar esta hoja  
**BeforeDoubleClick:** al presionar doble clic  
**BeforeRightClick:** al clic derecho en la hoja  
**Calculate:** al realizar cálculos en esta hoja  
**Change:** al introducir cambios en celdas de esta hoja  
**Deactivate:** al desactivar la hoja **FollowHyperlink:** al presionar un hipervínculo  
**PivotTableUpdate:** al actualizar una tabla dinámica en la hoja  
**SelectionChange:** al seleccionar una celda.

## 4.4 Macro al cambio en celdas

En el tema anterior vimos la lista de los principales eventos para hojas.

Pero dos eventos merecen un apartado especial, por ser la base de casi todas las rutinas:  
**Change y SelectionChange.**

### **Worksheet\_Change:**

Controla si la celda activa, a la que se denomina **Target**, ha **cambiado**  
(Ver tema **Target** en **cap. 3-[Nombrando Celdas](#)**.....)

Tareas como:

*Ejecutar rutina al ingresar valores en col, Colorear celdas según qué valor tomen, Controlar si lo ingresado es xxxx valor, etc necesitan una rutina que se colocará en el evento:*

**Worksheet\_Change** de la hoja activa.

Para controlar estos cambios podemos consultar si la celda activa (**Target**) tiene cierta dirección :

***If Target.Address(false, false) = "A5" Then (Ver \*)***

o si el cambio se produjo en cierta columna:

***If Target.Column = 3 Then 'si se ingresó cambios en celda de la col C***

O consultar si la celda activa se encuentra en cierto rango:

***If isEmpty (Intersect(Target, Range("miRango"))) Then***

Aquí estamos consultando si la intersección de la celda activa (**Target**) con un cierto rango está vacía (**Empty**)

En otras palabras si la comparación resulta verdadera (está vacía) significa que la celda activa no se encuentra en el rango

**Ejemplo:**

En el siguiente ejercicio, se ocultará la **fila** si en la col C se ingresa valor = OK

```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Column = 3 Then
'si el valor ingresado en col C es = "OK"
If Target.Value = "OK" Then
'se oculta la fila
Target.EntireRow.Hidden = True
End If
End If
End Sub
```

## 4.5 Macro al seleccionar celda

A diferencia del tema anterior, lo que aquí vemos es cómo evaluar qué celda ha sido seleccionada, y realizar una cierta acción.

**Worksheet\_SelectionChange**

A diferencia del evento anterior, este evento **captura la selección de una celda**.

Por ejemplo: para establecer mayúsculas en cada celda que ingresemos, colorear la celda activa, etc.

(Ver tema **Target** en **cap. 3-Nombrando Celdas.....**)

**Ejemplo 1:** La celda seleccionada cambia a fuente 15

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
Target.Font.Size = 14
End Sub
```

**Ejemplo 2:** Al seleccionar la celda A5 se solicita una clave

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
'al seleccionar celda A5 se solicita el ingreso de algún dato
If Target.Address(False, False) = "A5" Then
Dim pass
pass = InputBox("Ingresa password")
'otras instrucciones
'.... End
If End
Sub
```

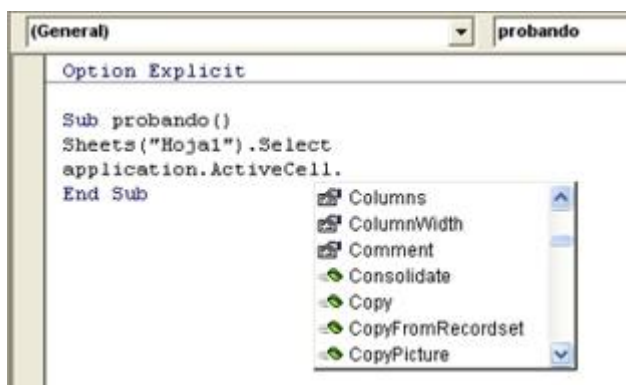
## 4.6 Metodos

Un **Método** es una **acción** que podemos realizar **sobre un objeto**.

Ejemplos de **métodos** serían: Agregar, Mover, Eliminar, Borrar, Copiar

Al escribir una instrucción veremos que se despliega la lista de **métodos y propiedades** (detalladas en el tema siguiente) propias para el objeto del que estamos haciendo referencia.

Podemos diferenciar en la imagen, en la lista desplegable, las **propiedades** (el icono con la manito) de los **métodos** (el icono verde)



En los capítulos correspondientes a **Libros, Hojas y Celdas-Rangos** veremos los principales **métodos** para cada uno de estos objetos.

## 4.7 Propiedades

Las **propiedades** son las **características de cada objeto**. Por ejemplo tamaño, color, con o sin protección, con o sin formato, etc.

Si por ejemplo, escribimos:

**Application.ActiveCell.** se mostrará una lista con las **propiedades** para una celda: entre otras, si tiene fórmula (HasFormula), el ancho (Height), si estará oculta (Hidden), la alineación horizontal y más.

*(Ver imagen en tema anterior)*

En los capítulos correspondientes a **Libros, Hojas y Celdas-Rangos** veremos las principales **propiedades** para cada uno de estos objetos.

# Capítulo

A large, solid gray circle containing a white, bold, sans-serif capital letter 'V' in the center.

V

## 5 5 - Ejecutando macros

### 5.1 Dónde colocar las macros

#### *Dónde escribir una macro?:*

En los ejercicios de los próximos capítulos, hago mención muchas veces, que las instrucciones pueden ser colocadas en cualquier punto de una rutina.

Por ejemplo *Abrir un segundo libro* es una **instrucción** que puede ser necesaria en algún punto de una **rutina** que realiza cálculos.

Se abrirá el segundo libro para volcar los resultados de estos cálculos.

#### **¿Pero dónde colocar esa rutina "principal"?**

Una vez dentro del Editor, básicamente las reglas son:

**1.** Si se trata de rutinas que se ejecutarán en ciertas **Hojas**, seleccionar con doble clic la **Hoja** correspondiente que se encuentra en el panel de la izquierda, entre los objetos del Proyecto. Escribir la rutina en ese espacio o ventana.

En este grupo de rutinas se encuentran las que se ejecutan al cambio o selección de celdas, al clic en botones insertados en la hoja, al activarse o desactivarse la hoja, etc.

#### **Ejemplos:**

```
Private Sub Worksheet_Change()  
    'instrucciones  
End Sub
```

```
Private Sub CommandButton1_Click()  
    'instrucciones  
End Sub
```

**2.** Si las instrucciones deben afectar a todo el **libro**, entonces seleccionar el objeto del proyecto denominado **ThisWorkbook** (o Este Libro según la versión).

Son las rutinas que contienen los eventos **Open** (al abrir el libro), **BeforeClose** (antes de cerrarlo), etc.

#### **Ejemplos:**

```
Private Sub Workbook_Open()  
    MsgBox "Bienvenidos" 'mensaje de bienvenida al abrir el libro  
End Sub
```

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
    MsgBox "Hasta la próxima- Que tengas un buen día"  
End Sub
```

3. Las rutinas que realizan procesos especiales y serán llamadas desde cualquier hoja, serán colocadas en un **módulo** (desde el Editor, menú Insertar, Módulo). Estas rutinas son del tipo: '*Cambiar a mayúsculas*', '*Colorear celdas*', '*Copiar datos*' o las asociadas a botones de la Barra Formularios (\*Ver '**Cómo ejecutar Macros**')

**Ejemplo:**

```
Sub ColorearFilas()  
'instrucciones  
End Sub
```

4. Las rutinas para controles de **Userforms**, deben ser colocadas en el formulario. Primero se inserta un Userform (desde el Editor, menú Insertar Userforms) y se dibujan los controles necesarios. Con doble clic sobre cada control (o con clic en el botón Código, en el panel de la izquierda) se pasa a la sección donde se copiará o escribirá la rutina para ese control o para el Userform.

(\* ver más detalles en capítulo **Userforms**)

**Ejemplo:**

```
Private Sub UserForm_Initialize()  
'inicializa los textbox con 1  
TextBox1.Value = 1  
TextBox2.Value = 1  
End Sub
```

5. Algunas instrucciones que se encuentran en los capítulos siguientes no comienzan por **Private Sub** ni por **Sub**. Están 'sueltas'.

Quedaron así porque pueden ser colocadas **en cualquier evento**, es decir que pueden ser incluidas en cualquier otra rutina y serán ejecutadas desde esa otra rutina principal.

**Ejemplo:**

La siguiente instrucción es a fines de mostrar cómo se selecciona una columna y será incluida en alguna rutina que necesite esta selección.

```
ActiveSheet.Range("C:C").Select
```

## 5.2 Cómo ejecutar una macro

**Ejecutar una macro:**

Si se encuentran en un Objeto Hoja o Libro (*ThisWorkbook*) se ejecutarán automáticamente al producirse el **evento** que las llama. Por ejemplo al activar una hoja, al abrir el libro, al cerrarlo, etc. (\* ver capítulo **Eventos, Métodos y Propiedades**)

Si las macros se encuentran en un Módulo, 'formalmente' se ejecutan desde el menú **Herramientas, Macros**, de la hoja Excel. Allí se seleccionará el nombre de la macro a ejecutar.

**En versión Excel2007, desde la ficha Programador - Botón Macros (1er grupo)**

Pero hay otras maneras de ejecutar una macro:

- 1- utilizando algún botón que al hacer clic sobre él se llama a la macro para su ejecución
- 2- utilizando un atajo de teclado, tal que al presionar las teclas asignadas se ejecute la rutina.

A continuación en detalle cada uno de estos métodos.

## 5.3 Ejecutar macro desde un botón

### 1. Utilizando botones

Desde el menú Ver de Excel, Barras de Herramientas, Cuadro de controles o Formularios, es posible insertar botones de comando. Una vez dibujados en la hoja, se le asignarán o escribirán las instrucciones a ejecutar al clic de esos botones.

**En versión Excel2007, desde la ficha Programador - Botón Insertar (2do grupo)**  
**(Ver imagen en cap 2: [Qué es una macro?](#))**

**a) Utilizando Cuadro de Controles:** una vez dibujado el *botón de comando* en la hoja, clic derecho sobre él y optar por '**Ver código**'. El control se pasará al Editor, en la hoja donde fue dibujado el botón, donde se escribirán las instrucciones a ejecutar entre estas 2 que ya aparecerán:

```
Private Sub CommandButton1_Click()  
End Sub
```

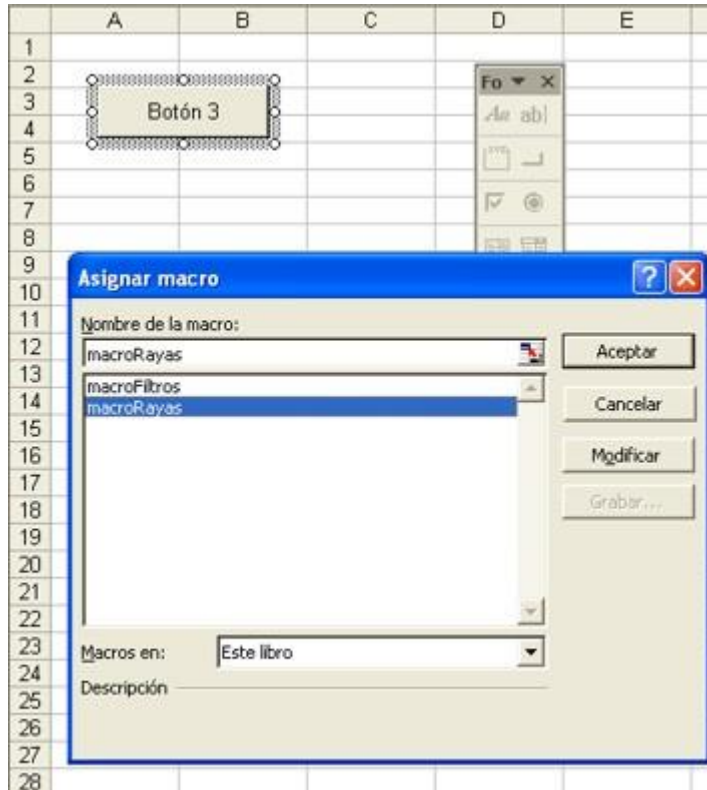
Para ejecutar la rutina, se debe desactivar el **Modo Diseño** del botón, desde la barra de herramientas **Cuadro de controles**.



**b) Utilizando la Barra Formularios:** al dibujar el botón en la hoja aparecerá la ventana '**Asignar Macro**', para asignar una ya creada y ubicada en un Módulo, o para escribirla en este momento, presionando el botón **Nuevo** (en la imagen aparece como '**Modificar**').

La macro se ubicará en un **Módulo**, y generalmente se llamará 'nombre\_del\_botón\_Al hacer click'. Recomiendo cambiar este nombre por uno más específico, como *ActualizaDatos*, *BorraFilas*, etc.

Predeterminadamente la macro se guardará en '*Todos los libros Abiertos*', pero salvo que sea eso lo deseado, es recomendable cambiar por la opción '**Este libro**'



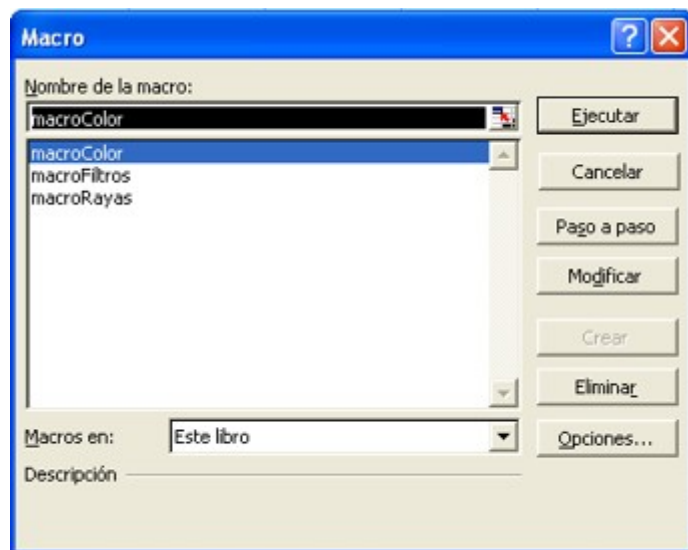
## 5.4 Ejecutar macro con atajo de teclado

### 2. Utilizando atajos de teclado:

Cuando ejecutaremos con frecuencia una rutina y no queremos colocar botones en la hoja, se puede recurrir a los atajos de teclado.

Para ello ir al menú **Herramientas, Macro, Macros**. Seleccionar la nuestra y con el botón **Opciones** podremos asignar una letra, que presionaremos a continuación de la tecla **Control** o **CTRL+Mayús**. si Excel ya tiene asignada esa letra.

Recomiendo utilizar las letras menos frecuentes en el entorno Windows, como K, M o N



### Ejercicio práctico:

Ir al Editor (Alt+F11), insertar un Módulo y copiar lo siguiente:

```
Sub macroColor()
'colorea la celda activa de color rojo
ActiveCell.Interior.ColorIndex = 3
End Sub
```

Ahora desde la hoja Excel, ir al menú **Herramientas, Macros**, seleccionar la que se llama: **macroColor**, presionar el botón **Opciones** e ingresar la letra 'n' (sin las comillas). Aceptar

**En versión Excel2007, desde la ficha Programador - Botón Macros (1er grupo)**

Probar pasando por distintas celdas y presionar las teclas **CTRL y n**

## 5.5 Acerca de las macros Auto-Open

Si deseamos que al abrir el libro se realicen ciertas acciones podemos colocar estas instrucciones en el evento **Open** del libro (1) o en una rutina denominada **Auto\_Open** (2)

1- En el primer caso, seleccionamos el objeto **ThisWorkbook** del Editor, y colocamos la rutina dentro de estas instrucciones:

```
Private Sub Workbook_Open()
MsgBox "Bienvenidos" 'mensaje de bienvenida al abrir el libro
'otras instrucciones
End Sub
```

2- Para el segundo caso tendremos en un módulo una rutina dentro de estas instrucciones:

```
Sub Auto_Open()  
    MsgBox "Bienvenidos" 'mensaje de bienvenida al abrir el libro  
    'otras instrucciones  
End Sub
```

### **ATENCIÓN:**

Hay que tener en cuenta estas diferencias:

- a- Si ambas rutinas se encuentran en el libro, primero se ejecutará la número 1 (**Private Sub Workbook\_Open()**)
- b- Si el libro es abierto a través de una macro desde otro libro, la rutina **Auto\_Open** **NO** será ejecutada.
- c- El tener presionada la tecla **ESC** al abrir un libro evita la ejecución de las macros de apertura.

## 5.6 Acerca de las macros Auto-Close

Así como hemos visto las rutinas Auto\_Open también podemos tener rutinas en módulos, denominadas **Auto\_Close**

Estas se ejecutarán al cerrarse el libro, de la misma manera que si las tuviéramos en el evento **BeforeClose** del libro.

### **Ejemplos:**

1- Seleccionamos el objeto **ThisWorkbook** del Editor, y colocamos la rutina dentro de estas instrucciones:

```
Private Sub Workbook_BeforeClose()  
    MsgBox "Hasta pronto" 'mensaje de salida al cerrar el libro  
    'otras instrucciones  
End Sub
```

2- Insertamos un módulo con una rutina dentro de estas instrucciones:

```
Sub Auto_Close()  
    MsgBox "Hasta pronto" 'mensaje de salida al cerrar el libro  
    'otras instrucciones  
End Sub
```

### ATENCIÓN:

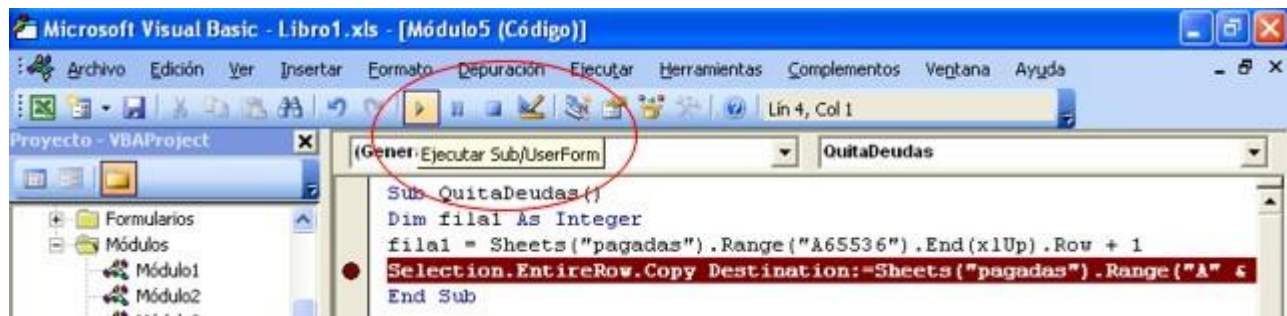
Si el libro se cierra desde otro libro, hay que llamar a la macro Auto\_close con esta instrucción para que se ejecute:

Application.Run "'Libro1.xls'!Auto\_Close" 'ajustar la extensión tratándose de versión Excel 2007

Donde **Libro1** será el nombre del libro que se trata de cerrar y cuenta con una rutina Auto\_Close

## 5.7 Cómo probar una macro

Para **probar una macro**, desde la ventana del Editor, presionar el botón **Ejecutar**.

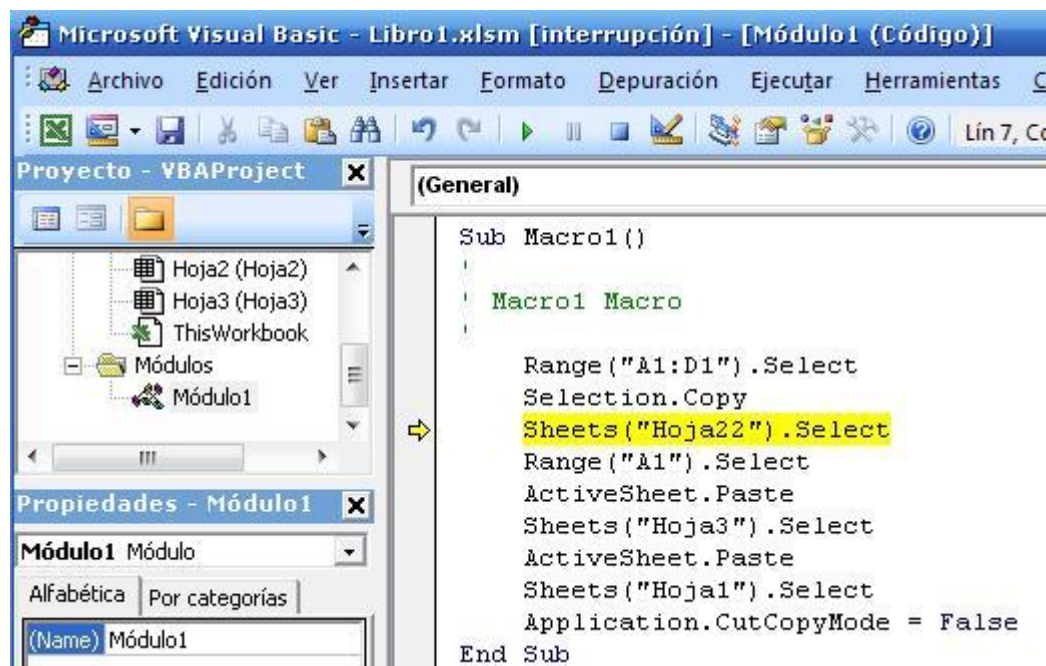


Si se presenta un **error**, se mostrará una ventana con opción de **Finalizar o Depurar**.



Presionando el botón **Depurar** se volverá al Editor donde se mostrará de amarillo la línea donde se produjo el error.

En el ejemplo se trata de un error por no existir la *Hoja22*, lo correcto es *Hoja2*. Se deberá corregir el error y presionar nuevamente el botón que ahora se mostrará como **Ejecutar/Continuar**



En cambio si se presiona el botón **Restablecer** (el del cuadrado) finaliza la ejecución de la rutina, al igual que si se hubiese presionado el botón **'Finalizar'** del cuadro 'mensaje de error'.

**Nota:** Si el proyecto se encuentra **protegido**, no aparecerá la opción de **Depurar**.

## 5.8 Cómo controlar la ejecución de una macro

Se puede **controlar la ejecución de una macro** colocando **puntos de interrupción** en las líneas a controlar. Haga clic en el margen izquierdo de la línea.

**Atención:** la línea donde se interrumpe aún **no ha sido ejecutada**, por lo que en un cálculo las variables aún no han tomado el valor que dicha línea le asigne.

**Ejemplo:**



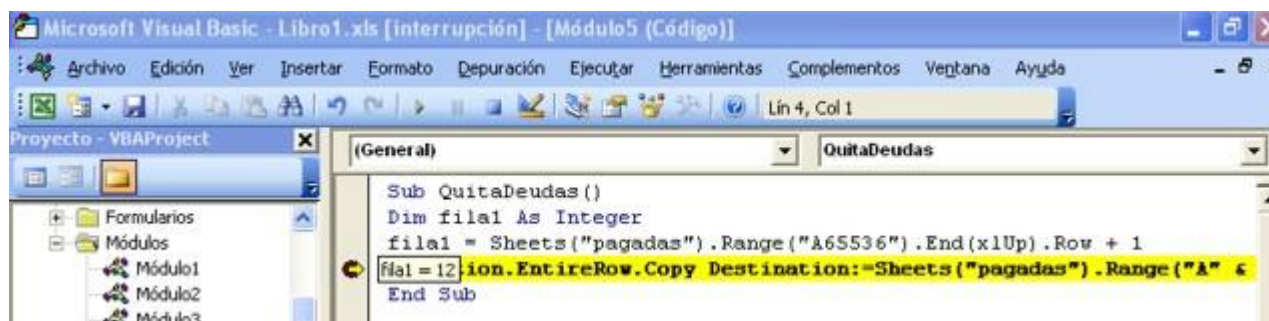
En la primer interrupción la variable *Unacelda* presentará un valor vacío porque esa línea aún no se ejecutó.

## 5.9 Conocer el valor que toman las variables

Colocando *puntos de interrupción*, se puede acceder a la rutina y observar el **valor que han tomado las variables, al pasar el cursor por encima**.

En la imagen observamos el valor que presenta la variable *fila1* en este punto. (= 12)  
Se ha colocado el punto de interrupción en la línea siguiente, una vez que se ha ejecutado la línea donde se asigna valor a la variable 'fila1'.

Este procedimiento es sumamente útil a la hora de ir controlando los valores que toman nuestras variables y corregir posibles errores.



## 5.10 Cómo evitar que una instrucción se ejecute

Para **evitar que una instrucción se ejecute**, colocar una **comilla simple** (') delante de una línea para que sea ignorada.

Esto es útil en casos de dudas acerca de alguna instrucción. Utilice este método en lugar de borrar la línea.

Cuando tenga su proyecto listo y funcionando, podrá borrar todas las líneas innecesarias.

***También se utiliza para dejar comentarios. A lo largo de este manual verá que se 'explican' las instrucciones utilizadas***

**Ejemplo:** se ignora el control de error porque se trata justamente de probar si presenta o no error la línea siguiente.

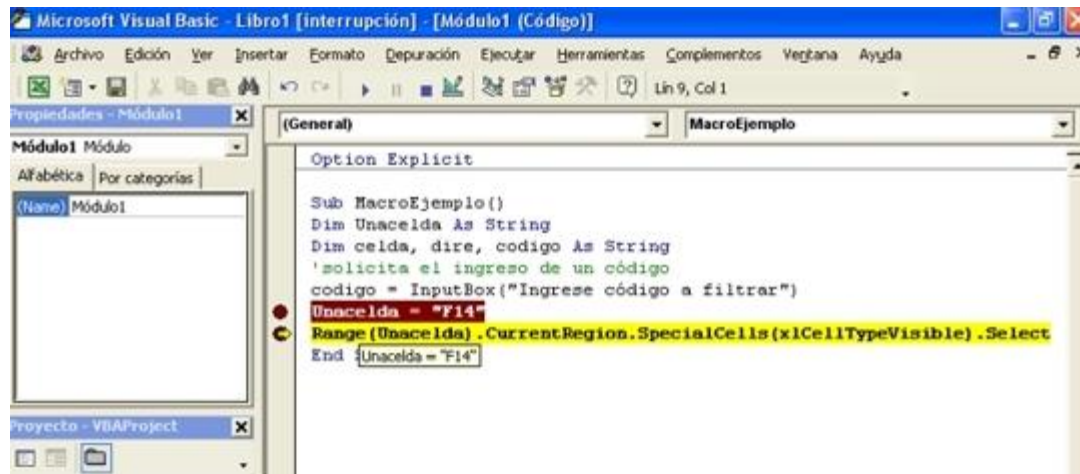
```
Sub Cancela()  
'On Error Resume Next  'esta línea no se ejecutará  
Unload UserForm1  
End Sub
```

**Nota:** posiblemente haya visto líneas precedidas por las letras REM. La comilla simple es una manera más rápida y reemplaza estas letras.

## 5.11 Acceder a la Ayuda desde una línea de código

Para obtener más información acerca del código utilizado, posicionar el cursor en la palabra y presionar la **tecla F1** para llamar a la Ayuda de Visual Basic.

**Ejemplo:** para ver la manera de ingresar los argumentos para el *InputBox* del ejemplo de la imagen, posicionar el cursor en dicha palabra, es decir hacer clic sobre la palabra, y presionar **F1**, para abrir la Ayuda de Visual Basic en este tema.



## 5.12 Salir de una rutina

Hasta ahora hemos visto que las rutinas, macros o procedimientos, se escriben entre 2 líneas, tanto las Públicas como Privadas:

***Sub Nombre\_rutina()***

***End Sub***

***Private Sub Workbooks\_Open()***

***End Sub***

Pero en ocasiones, se desea '**salir**' del procedimiento, y no continuar con el resto de instrucciones. En esos casos se recurre a la instrucción:

***Exit Sub***

**Ejemplo:**

***Sub miMacro()***

***'controla si se cumple alguna condición***

***If textbox1 = "" then***

***'no se ingresó dato en el control, entonces finaliza el proceso***

***Exit Sub***

***End If***

***'continúa el proceso de los datos del textbox***

***End Sub***



# Capítulo

VI

## 6 6 - Seguridad en el proyecto

### 6.1 Cómo proteger un proyecto

#### *Proteger el proyecto VBA:*

Cuando un libro o aplicación contenga macros, será conveniente **proteger** el proyecto, es decir proteger el acceso al Editor de Macros.

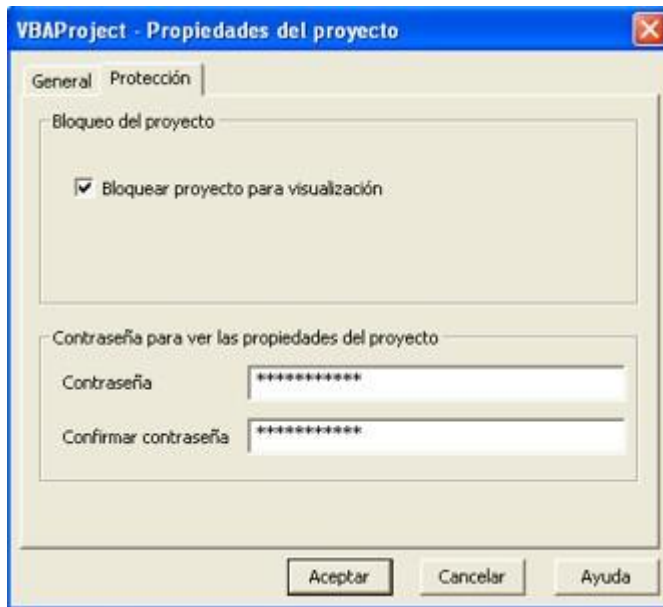
Para esto, dentro del **Editor**, pestaña **Herramientas**, optar por '**Propiedades de VBA Project**'

En la ventana que se nos presenta podemos:

1- asignar un nombre al proyecto actual e introducir una breve descripción. Si no asignamos un nombre siempre aparecerá como VBAProject (nombre\_del\_libro)



2- en la segunda pestaña podemos proteger el proyecto tildando la opción '**Bloquear proyecto...**' e ingresar una contraseña.



## 6.2 Evitar que las macros se vean desde el menú

Siguiendo con el tema '**Protección**' otro aspecto a tener en cuenta es **evitar que las macros se vean** desde el menú **Herramientas**, Macros de la hoja Excel, o desde la ficha **Programador** en versión Excel 2007.

Esto se evita colocando al inicio de cada módulo la instrucción:

### Option Private Module

Otro método es colocar delante de cada declaración '**Sub**' la palabra '**Private**'.

```
Private Sub MiMacro()  
    'instrucciones  
End Sub
```

Este es un método *individual*, solo será oculta la macro denominada 'MiMacro'. Si el módulo contiene otras rutinas, estas sí se verán desde la hoja Excel.

## 6.3 Cómo interrumpir una macro

Para **detener la ejecución** de una macro se debe presionar la tecla Escape (**ESC**).

Entonces si queremos **impedir** que se **interrumpa la ejecución** de las mismas, debemos **desactivar** el uso de la tecla **ESC**.

Para esto colocaremos una instrucción en el evento **Open** del libro.

**Ejercicio:**

Entrar al Editor, seleccionar el objeto ThisWorkbook (o Este Libro) y allí copiar esta rutina:

```
Private Sub Workbook_Open()  
Application.EnableCancelKey = xlDisabled  
End Sub
```

Al final de tu macro que no debe ser interrumpida, o en el evento **BeforeClose** del libro, agregar esta otra línea para volver la tecla a su estado normal:

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
Application.EnableCancelKey = xlInterrupt  
End Sub
```

## 6.4 Habilitar o no las macros

En versiones anteriores a Excel 2007 la opción **Seguridad** del menú **Herramientas, Macros** presenta 3 opciones:

1- Al abrir un libro con macros se habilitan sin aviso. No es un método recomendado si recibirá libros ajenos, que no son de su autoría.

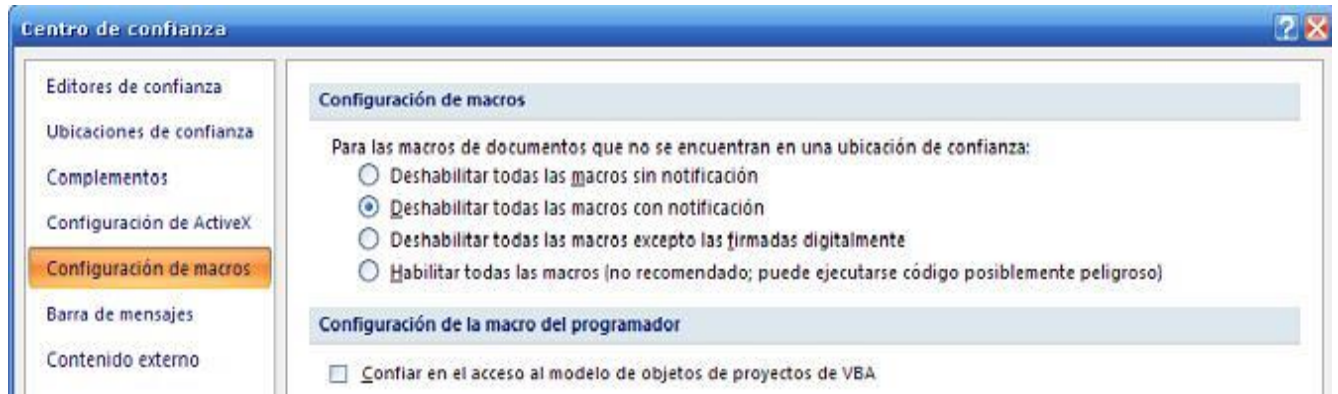
2- Al abrir un libro con macros recibirá una ventana notificando su contenido y permitiendo que ud decida si las habilita o no.

Este es el método más recomendado. Cuando se trate de libros que desconozca su contenido, puede 'Deshabilitar las macros', revisarlo y luego optar por abrirlo nuevamente 'Habilitando las macros' si lo considera necesario.

3- Al abrir un libro 'no se habilitan las macros' sin aviso previo. No podrá ejecutar ninguna rutina del libro. Si presiona algún botón con rutina asociada recibirá un aviso de que las macros fueron inhabilitadas.

**En versión Excel 2007:**

Desde la ficha **Programador**, 1er grupo (**Código**) presionamos el botón '**Seguridad de macros**', desplegándose la siguiente ventana:



Allí observamos **4 opciones**, siendo la **predeterminada la nro 2** coincidente con la opción 2 en versiones anteriores.

La novedad en esta versión es la opción nro 3 (Deshabilitar las macros excepto las firmadas digitalmente).

En este caso, si el libro contiene una firma digital que ya ha sido colocado en la categoría de 'editores confiables' se habilitará sin la aparición del cuadro de advertencia.



# Capítulo

VII

## 7 7- Tratamiento de Variables

### 7.1 Tipo de Variables

Se recomienda **declarar las variables** aunque no sean obligatorias, no solo por una cuestión de orden sino porque de esta manera le indicamos a nuestro programa qué tipo de datos contendrá cada variable y no dejamos que se le asigne el tipo predeterminado (Variant) que utiliza mayores recursos en bytes como verán en la tabla adjunta.

Desde el **Editor**, menú **Herramientas, Opciones, Editor**, podemos dejar tildada la opción de **"Requerir declaración de variables"**.

Al hacerlo veremos que cada módulo comienza con la instrucción:

#### Option Explicit

Esto significa que si una variable no fue declarada el programa dará error al ejecutarlo, obligándonos a declararla.

Los distintos tipos de variables que podemos utilizar son:

Tipo	Bytes	Descripción	Comentario
Byte	1	0-255	Integrales positivos
Boolean	1	True/False	Valores discretos
Integer	2	-32.768 hasta + ...	Números enteros
Long (long int.)	4	-2.147.483.648 hasta + ...	Números enteros
Single	4	-3,402823 E38 hasta + ...	Decimales
Double	8	-1,79769313486232 E308 hasta + ...	Decimales
Currency	8	15 díg. + 4 decimales	Número, 4 dec.
Date	8	1-ene-100 hasta 31-dic-9999	Fechas
Object	4	referencia a objetos	Ej. "Workbook"
String	10+	carácteres Ascii (texto)	Texto
String (long. fija)	1+	carácteres Ascii, longitud predef.	Texto
Variant	16+	cualquier tipo de datos	Cubre la mayoría

### 7.2 Duración de las variables

Las variables pueden ser: **Locales o Públicas**

#### Variables Locales:

Las variables **Locales** son las que se declaran dentro de un procedimiento y sus valores sólo pueden ser utilizados en éste. Para declararlas se utiliza la sentencia **Dim**, generalmente al inicio del procedimiento, aunque también pueden ser declaradas en otros puntos del mismo.

**Ejemplo:**

```
Sub Macro1()  
Dim valor1 as Integer, valor2 as Integer  
Dim cadena1 as String  
  
'otras instrucciones  
End Sub
```

**Variables Públicas:**

Son las que estarán disponibles para todos los procesos, sus valores pueden ser utilizados en cualquier módulo.

Se declaran como **Public**. Recomendando utilizar un módulo especialmente destinado a la declaración de estas variables lo que facilitará su ubicación.

Si se declararán en un módulo utilizado para otro procedimiento deberán ser las primeras instrucciones

**Ejemplo:**

```
Option Explicit  
Public minro as Byte  
  
Sub Macro2()  
'instrucciones  
End Sub
```

**Constantes:**

A diferencia de las variables que modifican sus valores durante la ejecución de un proceso, las **Constantes** mantienen su valor. También pueden ser **Locales o Públicas**

**Ejemplos:**

```
Const dia as Integer  
  
'o si fuese pública:  
Public Const cadena1 as String
```

## 7.3 Determinar el tipo de variable

La función **TypeName** aplicada sobre el valor de una celda o variable devuelve de qué tipo de variable se trata.

Los valores a devolver son: Error, String, Double, etc

**Ejemplo1:** obtenemos en un cuadro de mensaje el tipo de contenido que presenta la celda activa.

```
Sub TipoVariable1()  
MsgBox TypeName(ActiveCell.Value)  
End Sub
```

**Ejemplo 2:** obtenemos en un cuadro de mensaje el tipo de variable que se ha ingresada por InputBox

```
Sub TipoVariable2()  
Dim valor As Variant  
valor= Application.InputBox("Ingrese clave")  
MsgBox TypeName(valor)  
End Sub
```

En este último ejemplo el valor a devolver será **Boolean** si se optó por el botón **Cancelar** del InputBox, o **String** si se ha ingresado un dato.

## 7.4 Convirtiendo variables

En muchas ocasiones deberemos establecer el **tipo de variables** con que se debe guardar un dato obtenido a través de un InputBox o de un Textbox.

Esto se logra haciendo uso de ciertas **funciones de conversión** que trae Visual Basic. Las principales son:

**CInt:** devuelve número entero

Si valor = 125,30, ActiveCell.Value = **CInt**(valor) devolverá 125

**CDec:** devuelve valor decimal

**cDbl:** devuelve valor double

**CSng:** devuelve valor single

**CStr:** devuelve el mismo intervalo que **Double** para valores numéricos. El mismo intervalo que **String** para valores no numéricos

**CDate:** reconoce formatos de fecha que se ajusten a la configuración regional de su sistema:

```
valor = Application.InputBox("Ingrese fecha")  
ActiveCell.Value = CDate(valor)
```

**Trim:** devuelve una cadena sin espacios. Otras variantes son: **LTrim** (sin espacios a la izquierda) y **RTrim** (sin espacios a la derecha)

```
Trim("  Hola  ") 'devuelve: "Hola"
```

**Val:** devuelve los números de una cadena, ignorando los espacios

```
Val(" 1615 198 Calle N.E.") 'devuelve 1615198
```

**UCase:** convierte en mayúsculas el contenido de una celda o variable.

```
'primero convertimos a mayúsculas para realizar la comparación  
texto1 = UCase(Range("A1").Value)
```

**LCase**: convierte a minúsculas el contenido de una celda o variable.

Si el argumento de la función **CHR()** son números del 0 al 31 devolverá los códigos ASCII estándar no imprimibles.

Por ejemplo, **Chr(10)** devuelve un carácter de avance de línea o salto de carro en los MsgBox.

## 7.5 Limpiando variables

Para evitar errores y para un mejor aprovechamiento de los recursos, es ideal limpiar o **liberar las variables** utilizadas.

### **Algunos ejemplos:**

Líneas que serán colocadas al finalizar el proceso que utiliza esas variables.

'inicializar o dejar en 0 la variable utilizada  
**miValor = 0**

'limpiar una matriz o Array  
**Erase miArray()**

'liberar una variable inicializada con Set  
**Set busca = Nothing**



# Capítulo

VIII

## 8 8 - Trabajando con Cadenas

### 8.1 Extraer partes de una cadena

Para **extraer partes de una cadena**, ya sea el contenido de una celda, una variable o el resultado de un **InputBox**, contamos con las siguientes funciones:

**Left:** devuelve el contenido de una cadena comenzando desde la izquierda a partir de la posición indicada en el segundo argumento.

**Right:** devuelve lo que se encuentra a la derecha de la cadena

**Len:** devuelve el largo de una cadena

**Mid:** devuelve lo que se encuentra a partir de cierta posición y del largo establecido

**InStr:** devuelve la posición inicial de una cadena en otra

**Ejemplos:** si la celda A1 contiene el valor 'ABC567DEF'

**Left**(Range("A1").Value, 3) 'devuelve 'ABC'

**Mid**(Range("A1").Value, 3) 'devuelve 'C567DEF'

**Mid**(Range("A1").Value, 4, 2) 'devuelve "56", pero como un texto (string)

Para convertirlo a número utilizar la función **Val**

(\*ver más sobre [Conversión de variables](#) en el cap 7: **Tratamiento de Variables**)

**Right**(Range("A1").Value, 2) 'devuelve 'EF'

También podemos utilizar la función **InStr** conjuntamente con **Mid** para obtener una cadena a partir de cierto caracter.

**Ejemplo:** Se trata de obtener el apellido sabiendo que se ubica después del caracter 'espacio'

**Sub variables() Dim**

**esp As integer Dim**

**apellido As String Dim**

**cadena As String**

**cadena = "Juan Perez"**

**'se obtiene la ubicación del espacio**

**esp = InStr("Juan Perez", " ")**

**'se obtiene la cadena a partir de la posición del espacio**

**apellido = Mid(cadena, esp)**

**'se muestra el resultado**

**MsgBox apellido**

**End Sub**

## 8.2 Armando cadenas

Para **armar cadenas**, ya sea en una variable, una celda o un mensaje, utilizamos el caracter '&'

### Ejemplos:

1- Si B10 contiene el valor 'ABC', la instrucción:

**Range("B10").value = Range("B10").value & "-"** 'devuelve: 'ABC-'

2- Si la variable miValor contiene la cadena: 'Enero', la siguiente rutina devolverá, siguiendo el ejemplo anterior el resultado: Enero/ABC-, tal como lo muestra el cuadro de mensaje:

```
miValor = "Enero"  
miValor = miValor & "/" & Range("B10")  
msgbox miValor
```

3- Ahora mostramos en un cuadro de mensaje el contenido de la variable miValor a la que concatenaremos la fecha actual

**MsgBox miValor & Date()** 'devolverá: 'Enero/ABC-20/01/08'

Siguiendo estos ejemplos se pueden armar todo tipo de cadenas.

## 8.3 Obtener el largo de una cadena

Es probable que en ciertos casos tengamos que saber el largo de una cadena o el total de caracteres de un rango de celdas.

En esos casos hacemos uso de la **función Len** (largo)

**Ejemplo 1:** devolver en una celda el largo del contenido de otra

```
Sub largoCadena()  
Dim cantCarac as integer  
cantCarac = Len(Range("A5"))  
'devolvemos en una celda el total de caracteres  
Range("B5") = cantCarac  
End Sub
```

**Ejemplo 2:** devolver en un mensaje el total de caracteres de un rango que previamente se habrá seleccionado

```
Sub CuentaCaract()  
Dim celda As Range  
Dim I As Integer  
'se recorre cada celda del rango  
For Each celda In Selection '(* Nota)  
I = I + Len(celda.Value)
```

```
Next  
MsgBox "Se encontraron " & I & "caracteres y espacios en la selección"  
End Sub
```

**Nota:** el tema **For Each...Next** se halla explicado en el cap **12:** [Bucles-Comandos especiales](#).

## 8.4 Introducir caracteres especiales

La **función CHR()** permite introducir caracteres especiales en cuadros de mensajes o al introducir valores en celdas.

**Ejemplo 1: CHR(13)** para introducir texto en diferentes líneas

```
MsgBox "Bienvenidos " & Chr(13) & "al curso VBA"
```

(también puede utilizarse **Chr(10)** )

**Ejemplo 2:** para introducir comillas dobles en un texto, se usa **Chr(34)**

```
MsgBox "El nombre del capítulo es " & _  
Chr(34) & "Trabajar con Cadenas" & Chr(34)
```

**Ejemplo 3:** para introducir un espacio de tabulación se utiliza **Chr(9)**

```
MsgBox "Este libro se llama " & Chr(13) & Chr(9) & ActiveWorkbook.Name
```

**Ejemplo 4:** para escribir en 2 líneas dentro de una celda (como presionando Alt+Enter), utilizamos **Chr(10)**

```
ActiveCell.Value = "Código de" & Chr(10) & "Proveedor"
```

**Ejemplo 5:** la siguiente rutina devuelve en una columna de la Hoja 1, la lista de símbolos que se pueden obtener a partir de **Chr(33)**. Los caracteres del 1 al 32 son caracteres 'no imprimibles' Esta lista corresponde a los caracteres ASCII

```
Sub listarCaracteres()  
Dim I As Integer  
'colocamos título a la columna A y B  
Worksheets(1).Cells(1, 1).Value = "Chr #"  
Worksheets(1).Cells(1, 2).Value = "Símbolo"  
'recorremos el rango de caracteres Ascii
```

```

For I = 33 To 255
    Worksheets(1).Cells(I - 30, 1).Value = I
    Worksheets(1).Cells(I - 30, 2).Value = Chr(I)
Next
'se establece alineación horizontal = centrar
Worksheets(1).Columns("A:B").HorizontalAlignment = xlCenter
End Sub

```

## 8.5 Detectar o encontrar texto en una cadena

Si necesitamos saber si cierto texto se encuentra en una cadena o en una celda utilizaremos la **función InStr**, ya que esta función nos devuelve la posición del texto buscado. Si el valor devuelto es **> 0** significará que el texto **ha sido encontrado** en la cadena.

**Ejemplo:** se desea saber si la celda A1 contiene la palabra "JUAN"

```

Sub buscaTexto()
    Dim texto1 As String
    'primero convertimos a mayúsculas para realizar la comparación
    texto1 = UCase(Range("A1").Value)
    'si la ubicación de la palabra "JUAN" es mayor a 0 ha sido encontrado
    If InStr(texto1, "JUAN") > 0 Then
        'instrucciones si el texto fue encontrado
        MsgBox "Encontrado"
    Else
        'instrucciones si el texto NO fue encontrado
        MsgBox "No Encontrado"
    End If
End Sub

```

## 8.6 Creando cadenas de largo fijo

Para crear **cadenas de largo fijo** como **asteriscos** u otro caracter podemos utilizar instrucciones como estas:

**Ejemplos:**

```

miAster = String(10, "*")    ' devolverá en la variable 10 asteriscos

Range("B2") = String(5,35)  ' devolverá 5 numerales # . 35 corresponde al caracter ASCII

```

## 8.7 Obtener la parte numerica de una cadena

Podemos utilizar la función **VAL** para obtener la parte numérica de una cadena.

### **Ejemplo1:**

```
valor1 = "5401 - ABC"  
valor2 = Val(valor1) 'devolverá 5401
```

**Ejemplo2:** si la celda A3 = \$120, para obtener el valor sin el signo, haremos:

```
valor1 = Range("A3")  
If Left(valor1, 1) = "$" Then  
    'tomamos el valor de la variable a partir de la posición 2  
    valor2 = Val(Mid(valor1, 2))  
Else  
    valor2 = Val(valor1)  
End If
```

**Nota:** utilizar **Cdbl** en lugar de **Val** en casos de aplicaciones internacionales, donde exista la posibilidad de utilizar diferentes separadores decimales.



# Capítulo

IX

## 9 9 - Trabajando con Libros

### 9.1 Principales Metodos y Propiedades de Libros

(Ver descripción e imagen en cap 4: [Eventos, Métodos y Propiedades](#))

#### Métodos más comunes

<b>Add</b>	Para Crear un nuevo Libro de Trabajo
<b>Open</b>	Para Abrir un Libro de Trabajo
<b>Close</b>	Para Cerrar un Libro de Trabajo
<b>Save</b>	Para Guardar un Libro de Trabajo Existente ( <b>Guardar</b> del menú Archivo)
<b>SaveAs</b>	Para Guardar un Libro de Trabajo Existente ( <b>Guardar Como</b> del menú Archivo)

#### Propiedades más comunes

<b>ActiveSheet</b>	Hoja activa del Libro de trabajo
<b>Name</b>	Nombre del Libro de Trabajo
<b>Password</b>	Devuelve o establece la contraseña para abrir el libro
<b>Path</b>	Ruta en la que se encuentra el Libro de Trabajo
<b>Saved</b>	Estableciendo esta propiedad en 'True' se cerrará sin guardar los cambios
<b>Worksheets</b>	Conjunto de Hojas que componen el Libro de Trabajo

### 9.2 Abrir un libro. Abrir libro con clave

La instrucción **Open** puede ser colocada en cualquier parte de una rutina, en el momento que se necesita trabajar con un segundo libro:

***Application.Workbooks.Open "ruta y nombre de libro"***

#### **Ejemplo 1:**

***Application.Workbooks.Open "C:\Mis documentos\Libro1.xls"***

#### **Ejemplo 2:**

***Application.Workbooks.Open ThisWorkbook.Path & "\Libro1.xls"***

En este ejemplo se utilizó la ruta del libro activo, con la expresión: **ThisWorkBook.Path**

#### **Ejemplo 3:**

Si el libro que necesitamos abrir está protegido con clave deberemos indicarla en la instrucción de apertura, con la expresión **Password**

***Application.Workbooks.Open "C:\Mis documentos\Libro1.xls", Password:= "miclave"***

**En versión Excel 2007:**

Tener en cuenta las distintas 'extensiones' de los archivos en esta nueva versión:

**xlsx**: libro sin macros, **xlsm**: libro habilitado para macros, **xltx**: plantilla, **xltm**: plantilla con macros, y otras.

**Ejemplo 4:**

*Application.Workbooks.Open "C:\Documents and Settings\All Users\Documentos\LibroNuevo.xlsm"*

**Nota:** Al abrir un libro, éste pasa a ser el libro Activo

## 9.3 Ejecutar macro al abrir un libro

Para que una rutina se ejecute al abrir el libro, deberá colocarse en el objeto: ThisWorkbook (del panel de la izquierda), en el evento **Open**:

**Ejemplo 1:**

```
Private Sub Workbook_Open ()  
    'este ejemplo establece que siempre que se abra el libro se seleccione la hoja 2  
    Sheets("Hoja2").Select  
End Sub
```

**Ejemplo 2:**

```
Private Sub Workbook_Open ()  
    'este ejemplo llama a una rutina que se encuentra en un módulo  
    Nuevamacro  
End Sub
```

En un *módulo* del proyecto actual se deberá tener entonces esta macro:

```
Sub Nuevamacro()  
    'instrucciones que se ejecutarán al abrir el libro  
End Sub
```

**Nota:** será lo mismo llamar a la macro solo por su nombre (Nuevamacro, como en el ejemplo) o con la instrucción **Call**:

```
Private Sub Workbook_Open ()  
    'este ejemplo llama a una rutina que se encuentra en un módulo  
    Call Nuevamacro  
End Sub
```

## 9.4 Al abrir libro incrementar un contador

Si desea llevar un contador que indique el número de aperturas que tuvo su libro, debe agregar la siguiente rutina en el objeto ThisWorkbook del Editor de macros:

```
Private Sub Workbook_Open()  
    'el contador se estableció en la Hoja1, celda AZ1  
    Sheets("Hoja1").Range("AZ1").Value = _  
    Sheets("Hoja1").Range("AZ1") + 1  
End Sub
```

**Atención:** deberá asegurarse que se guardan los cambios al cerrar el libro para que refleje el cambio en la celda AZ1 (\*Ver "**Guardando Libros**" en este mismo capítulo)

## 9.5 Seleccionar un libro

El método **Select** es el utilizado para seleccionar

Si tenemos más de un libro abierto, para seleccionarlos usaremos alguno de estos ejemplos:

**Ejemplo 1:** `Workbooks("Libro1.xls").Select` 'ajustar la extensión tratándose de versión Excel 2007

**Ejemplo 2:** `Workbooks(2).Select`

## 9.6 Activar otro libro distinto al actual

La expresión **Activate** activa la primer ventana del libro mencionado, pero no ejecuta ninguna macro Auto\_Open del mismo.

**Ejemplo 1:** `Workbooks("Libro1.xls") Activate` 'ajustar la extensión tratándose de versión Excel 2007

**Ejemplo 2:** `Workbooks(2).Activate`

## 9.7 Obtener la ruta de un libro

La ruta de un libro abierto, se obtiene con la expresión **Path**

### **Ejemplo:**

```
Sub BuscoRuta()  
'se define la variable que guardará la ruta del archivo  
Dim miRuta As String  
  
'se guarda la ruta en la variable  
miRuta = ActiveWorkbook.Path  
  
'muestra la cadena obtenida  
MsgBox miRuta  
  
End Sub
```

## 9.8 Guardando Libros

### 9.8.1 Guardar el libro activo

Los libros se pueden **guardar** en cualquier momento, tanto en rutinas ubicadas en módulos o dentro de los eventos de hojas o libro (ThisWorkbook).

A continuación algunos ejemplos del método **Save**:

La siguiente instrucción es el equivalente a presionar el botón **Guardar** de la barra de Excel.

```
ActiveWorkBook.Save
```

**Ejemplo:** guardar el libro antes de ejecutar la impresión

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)  
ActiveWorkbook.Save  
End Sub
```

### 9.8.2 Guardar un libro con otro nombre o formato

La siguiente instrucción (**SaveAs**) es el equivalente a la opción **Guardar como** del menú Archivo de Excel:

```
ActiveWorkbook.SaveAs Filename:="C:\Mi doc\Libro1.xls", FileFormat: =xlNormal,  
Password:="clave", ReadOnlyRecommended: =False
```

Los **argumentos** son:

**Filename:** ruta y nombre con que se guardará el archivo

**FileFormat:** formato del libro (ver Ayuda de Excel para otros formatos)

**Password:** clave del libro

**RedOnlyRecommended:** si se guarda como libro de solo lectura, optar por True en lugar de False

Si se omite algún argumento, escribir la coma, como en el siguiente ejemplo:

```
ActiveWorkbook.SaveAs Filename:="C:\Mis doc\Libro.xls",,Password:="clave"
```

### En versión Excel 2007:

Tener en cuenta las distintas 'extensiones' de los archivos en esta nueva versión:

**xlsx:** libro sin macros, **xlsm:** libro habilitado para macros, **xltx:** plantilla, **xltm:** plantilla con macros, y otras.

### Ejemplo: Libro guardado con formato 'binario'.

```
ActiveWorkbook.SaveAs Filename:=_  
"C:\Documents and Settings\All Users\Libro11.xlsm", FileFormat:=xlExcel12,  
CreateBackup:=False
```

## 9.8.3 Guardar un libro con clave

Con la siguiente instrucción el libro se guardará con el mismo nombre actual pero con clave

Nuevamente recordar que el valor de la clave puede ser el contenido de una celda o el de una variable.

```
ActiveWorkbook.SaveAs Password:="miclave"
```

**Nota:** para abrirlo se necesitará agregar la misma clave

```
Workbooks.Open "ruta y nombre", Password:="miclave"
```

(\*Ver el tema "**Abrir Libros**" en este mismo capítulo)

### 9.8.4 Guardar un libro cuyo nombre será el valor de una variable

La siguiente instrucción guarda el libro (en la misma ubicación) pero con nombre igual al contenido de la celda A2 de la hoja activa.

*ActiveWorkbook.**SaveAs** Filename:=Range("A2").Value*

### 9.8.5 Guardar un libro cuyo nombre serán datos concatenados

Utilizaremos una variable para "armar" el nombre del archivo. En el ejemplo se ingresa la ruta al directorio y como nombre el contenido de una celda con las 3 primeras letras del mes de la fecha actual.

**Ejemplo:** Guardar el libro con nombre: Facturas-Ene (o el mes que corresponda al actual)

Recuerden que se puede guardar un libro en cualquier punto de una macro, o en cualquier evento del libro.

Pero ahora, para probarlo, insertaremos un módulo y crearemos una rutina 'GuardaLibro'

En celda A2 guardamos el valor: Facturas

Ajustar en la rutina la ruta que aparece como : C:\Mis documentos\ por la habitual, por ej:  
C:\Documents and Settings\All Users\Documentos\

*Sub GuardaLibro()*

*'aquí colocaremos las instrucciones para armar el nombre del libro*

*'definimos una variable*

*Dim miNombre As String*

*'en la variable unimos el nombre de la carpeta, el contenido de la celda A2 de la hoja activa y 3 caracteres del mes actual*

*miNombre = "C:\Documents and Settings\All Users\Documentos\" & Range("A2") & "-" & Format(Now, "mmm") & ".xls"*

*'guardamos el libro con nombre igual a la variable*

*ActiveWorkbook.**SaveAs** miNombre*

*End Sub*

#### **Nota:**

#### **En versión Excel 2007:**

Tener en cuenta las distintas 'extensiones' de los archivos en esta nueva versión:

**xlsx**: libro sin macros, **xlsb**: libro habilitado para macros, **xltx**: plantilla, **xltm**: plantilla con macros, y otras.

## 9.9 Cerrando Libros

### 9.9.1 Cerrar todos los libros en uso

Al igual que Abrir o Guardar libros, las instrucciones para **cerrar** libros, pueden ser ejecutadas en cualquier punto de nuestras macros (por ejemplo ante cierto resultado o el ingreso de una clave inválida).

A continuación algunos ejemplos del método **Close**:

Si deseamos cerrar **todos los libros** abiertos, la instrucción será:

***Workbooks.Close***

Si hubo cambios en los libros, Excel mostrará los mensajes y cuadros de diálogo oportunos para preguntar si desea guardar estos cambios.

**Atención:** Al **cerrar** un libro desde Visual Basic no se ejecuta ninguna de las macros **Auto\_Close** del libro. Use el método **RunAutoMacros** para ejecutar las macros de cierre automático.

### 9.9.2 Cerrar un solo libro

Para cerrar un solo libro, cuando tenemos más de uno abierto, haremos:

**Ejemplo 1:** *Workbooks("Libro1.xls").Close*

**Ejemplo 2:** *Workbooks(2).Close*

Si el libro presenta cambios, Excel mostrará el mensaje preguntando si desea guardar los mismos.

### 9.9.3 Cerrar un libro SIN guardar los cambios

A continuación algunas maneras de cerrar un libro, sin guardar los cambios:

**Ejemplo 1:** *Workbooks("Libro1.xls").Close False* 'ajustar la extensión tratándose de versión Excel 2007

**Ejemplo 2:** `ActiveWorkbook.Close False`

**Ejemplo 3:** `Workbooks(1).Close SaveChanges:=False`

Otro manera es utilizando el método **Saved**:

```
'primero indicamos que ha sido guardado (aunque no lo hayamos guardado)
ThisWorkbook.Saved = True
'luego lo cerramos
ThisWorkbook.Close
```

#### 9.9.4 Cerrar un libro guardando los cambios

La siguiente instrucción cierra el libro guardando los cambios, sin consultar.

```
ActiveWorkbook.Close True
```



# Capítulo

A large, solid gray circle is positioned to the right of the word 'Capítulo'. Inside the circle is a large, white, bold letter 'X'.

## 10 10 - Trabajando con Hojas

### 10.1 Características del tratamiento de Hojas

Las instrucciones para el **manejo de Hojas**, pueden ser incluídas en cualquier punto de nuestras rutinas, ya sea que las tengamos en módulos o en ciertos eventos como por ejemplo en el evento **Open** del libro.

**Ejemplo 1:**

```
Private Sub Workbook_Open()  
    Sheets(2).Select  
End Sub
```

A continuación encontrarán ejemplos de los principales métodos y propiedades más comunes de las hojas.

**Ejemplo 2:**

Si una rutina debe ser ejecutada en **todas las hojas** de un libro, se colocarán en el objeto ThisWorkbook (o EsteLibro), como por ejemplo:

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)  
    'instrucciones que se ejecutarán cada vez que se active cualquier hoja del libro  
End Sub
```

**Ejemplo 3:**

Si en cambio tenemos una rutina que debe controlar lo que se ingrese **en cierta hoja**, se colocará en el **objeto Hoja** que corresponda.

```
Private Sub Worksheet_Change(ByVal Target As Range)  
    'instrucciones....  
End Sub
```

**Nota:** Ver más detalles en cap 4: **Eventos**, [Evaluar cambios en hojas](#).

### 10.2 Principales Metodos y Propiedades de Hojas

(Ver descripción e imagen en **cap 4:** [Eventos, Métodos y Propiedades](#))

#### Métodos más comunes

<b>Add</b>	Para Crear una nueva Hoja
<b>Copy</b>	Para Copiar una Hoja
<b>Delete</b>	Para Eliminar una Hoja
<b>FillAcrossSheets</b>	Para copiar un rango en el mismo lugar en varias hojas
<b>Move</b>	Para Mover una Hoja

<b>Print</b> (Out o Preview)	Para imprimir una Hoja o hacer Vista Preliminar
<b>Protect</b>	Para proteger una Hoja
<b>Select</b>	Para Seleccionar una Hoja

### Propiedades más comunes

<b>ActiveSheet</b>	Hoja activa del Libro de trabajo
<b>AutoFilter</b>	Indica si el filtrado está activado y 'Nothing' si está desactivado.
<b>AutoFilterMode</b>	Devuelve 'True' si las flechas desplegables de Filtro Automático están visibles.
<b>Cells</b>	Celda de la Hoja
<b>CodeName</b>	Devuelve el nombre de código del objeto
<b>Columns</b>	Para hacer referencia a Columnas
<b>Comments</b>	Devuelve un conjunto que hace referencia a los Comentarios
<b>Hyperlinks</b>	Devuelve un conjunto que hace referencia a los hipervínculos de la hoja
<b>Index</b>	Índice de la Hoja
<b>Name</b>	Nombre de la Hoja
<b>Next</b>	Hoja siguiente según el orden de las pestañas
<b>PageSetup</b>	Para establecer la configuración de página
<b>Previous</b>	Hoja anterior según el orden de las pestañas
<b>Range</b>	Rango de celdas o celda de la Hoja
<b>Rows</b>	Para hacer referencia a Filas
<b>ScrollArea</b>	Establece el rango en que está permitido el desplazamiento
<b>Shapes</b>	Hace referencia a los objetos dibujados o insertados en hoja
<b>Visible</b>	Indica con 'True' si la hoja está actualmente visible.

## 10.3 Activar o seleccionar otras hojas distintas a la actual

Hay distintas maneras de mencionar una hoja. En este caso utilizamos una instrucción por nombre y la segunda por su número de índice

***Sheets("Hoja2").Activate*** 'activa la hoja cuyo nombre es Hoja2

***Sheets(2).Select*** 'selecciona la hoja cuyo índice es 2

## 10.4 Seleccionar la hoja anterior o posterior a la activa

Para posicionarnos en la hoja anterior utilizamos la expresión "**Previous**"

***ActiveSheet.Previous.Select***

**Atención:** no se trata de la hoja donde estuvimos con anterioridad, sino la que se encuentra en la **pestaña anterior**.

Para posicionarnos en la hoja siguiente utilizamos "**Next**"

```
ActiveSheet.Next.Select
```

**Atención:** se trata de la hoja que se encuentra en la **pestaña siguiente**.

**Importante:** Estas instrucciones darán **error** si tratamos de seleccionar la **anterior** a la primer pestaña o la **siguiente** estando ya en la última.  
Ver en el [cap.16](#) cómo evitar este tipo de errores.

## 10.5 Devolver el nombre de la hoja en una variable

Siguiendo el ejemplo anterior, guardamos el nombre de la hoja anterior en una variable.

**Ejemplo:**

```
Sub HojaAnterior()  
Dim cadena1 As String  
  
'la variable guarda el nombre de la hoja anterior  
cadena1 = ActiveSheet.Previous.Name  
  
'se muestra el nombre en un mensaje  
MsgBox cadena1  
  
End Sub
```

**Importante:** Estas instrucciones darán **error** si tratamos de obtener el nombre de la **anterior** a la primer pestaña o la **siguiente** estando ya en la última.  
Ver en el [cap.16](#) cómo evitar este tipo de errores.

## 10.6 Seleccionar todas las hojas de un libro

Para seleccionar todas las hojas del libro activo, hacemos referencia al conjunto **Sheets**.

```
ActiveWorkbook.Sheets.Select
```

**Ejemplo:**

Seleccionar todas las hojas y asignarle color de fuente azul a todas.

```
Sub TodasAzules()  
ActiveWorkbook.Sheets.Select  
Cells.Select  
Selection.Font.ColorIndex = 5  
End Sub
```

## 10.7 Proteger una hoja

Aquí se presentan 3 ejemplos de cómo proteger una hoja utilizando el método **Protect**

```
ActiveSheet.Protect Password:="contraseña"  
'proteger con contraseña
```

```
Sheets("Hoja3").Protect  
'proteger sin contraseña
```

```
Sheets(2).Protect Password:="contraseña", DrawingObjects:=True, Contents:=True,  
Scenarios:=True
```

**Nota:** Este último ejemplo protege con las 3 condiciones tildadas que nos presenta la opción Proteger del menú Herramientas (hasta **versión Excel 2000**)

### Para versiones 2002-2003-2007:

El siguiente ejemplo protege la hoja dejando tildadas algunas opciones. En este caso se permite: seleccionar celdas desbloqueadas, dar formato a columnas e insertar filas.

```
ActiveSheet.Protect DrawingObjects:=True, Contents:=True, Scenarios:=True _  
, AllowFormattingColumns:=True, AllowInsertingRows:=True  
ActiveSheet.EnableSelection = xlUnlockedCells
```

**IMPORTANTE:** Para obtener la sintaxis apropiada a cada caso, realizar la **protección de la hoja** con la **grabadora encendida**. Al detenerla se encontrará en un módulo, la instrucción generada.

## 10.8 Desproteger una hoja

Para desproteger una hoja con o sin contraseñas aquí algunos ejemplos de instrucciones con el método **Unprotect** :

*ActiveSheet.**UnProtect** Password:="contraseña"* 'desproteger con contraseña

*Sheets("Hoja3").**UnProtect*** 'desproteger sin contraseña

## 10.9 Vista previa de la hoja activa y de otras hojas

La instrucción para una Vista Previa es "**PrintPreview**"

*ActiveSheet.**PrintPreview*** 'hoja activa

*Sheets("Hoja3").**PrintPreview*** 'hojas no activas

**Atención:** si no se cuenta con impresoras instaladas, la instrucción dará error. Para evitarlo anteponer una instrucción de control (**On Error**)  
(\*Véase el tema en el cap 16 "**Controlando Errores**"):

*On Error Resume Next*  
*ActiveSheet.**PrintPreview*** 'hoja activa

## 10.10 Imprimir hojas

Se utiliza el método **PrintOut** con los siguientes argumentos:

*ActiveSheet.**PrintOut**(From, To, Copies, Preview, ActivePrinter, PrintToFile, Collate, PrToFileName)*

**From Variant** opcional. El número de la página en que se empezará a imprimir. Si este argumento se omite, la impresión comenzará por el principio.

**To Variant** opcional. El número de la última página a imprimir. Si este argumento se omite, la impresión finalizará en la última página.

**Copies Variant** opcional. El número de copias que se imprimirán. Si este argumento se omite, se imprimirá una sola copia.

**Preview Variant** opcional. Si es **True**, Microsoft Excel activará la vista preliminar antes de imprimir el objeto. **False** (o se omite) para imprimir el objeto inmediatamente.

**ActivePrinter Variant** opcional. Establece el nombre de la impresora activa.

**PrintToFile Variant** opcional. Si es **True**, se imprimirá en un archivo. Si no se especifica **PrToFileName**, Microsoft Excel solicitará al usuario que introduzca el nombre del archivo de salida.

**Collate Variant** opcional. **True** para intercalar múltiples copias.

**PrToFileName Variant** opcional. Si **PrintToFile** se establece en **True**, el argumento especificará el nombre del archivo al que desea imprimir.

**Ejemplo 1:** imprimir todas las hojas de un libro

*On Error Resume Next*

*ActiveWorkbook.PrintOut From:=1, To:=3, copies:=1, collate:=True*

**Ejemplo 2:** imprimir una hoja con 2 copias

*'controlando posible error al no tener impresoras instaladas*

*On Error Resume Next*

*Sheets(2).PrintOut Copies:=2, Collate:=True*

**Ejemplo 3:** imprimir solo las celdas seleccionadas

*On Error Resume Next*

*ActiveWindow.Selection.PrintOut copies:=2, collate:=True 'imprime dos copias*

## 10.11 Insertar hojas

La instrucción básica para insertar hojas es la siguiente:

*Workbooks(1).Sheets.Add*

Con esta instrucción se insertará una hoja delante de la hoja activa. Otras opciones son:

### **1- Insertar una hoja después de la última hoja del libro activo**

*ActiveWorkbook.Sheets.Add After:=Worksheets(Worksheets.Count)*

### **2- Insertar una hoja delante de la última hoja del libro**

*ActiveWorkbook.Sheets.Add Before:=Worksheets(Worksheets.Count)*

### **3- Insertar una hoja al inicio (delante de todas)**

*ActiveWorkbook.Sheets.Add Before:=Worksheets(1)*

## 10.12 Eliminar hojas

El método a utilizar para eliminar hojas es: **Delete**

```
ActiveWorkbook.Sheets(5).Delete
```

```
Sheets("Hoja5").Delete
```

**Atención:** las 2 instrucciones no eliminan la misma hoja: la primera eliminará la hoja número 5 en el orden en que se encuentran las pestañas en el libro, en cambio la segunda eliminará aquella con nombre = Hoja5

## 10.13 Copiar hojas

La instrucción básica para copiar hojas es:

```
ActiveSheet.Copy
```

Lo que le falta a esta instrucción es decirle a dónde la copiaremos: en una nueva hoja, en otro libro o crearemos un libro con solo esa hoja.

**Ejemplo1:** se duplica la hoja activa, copiándola a continuación: de la última.

```
Sub copiaHoja1()  
ActiveSheet.Copy After:=Sheets(Sheets.Count)  
End Sub
```

**Nota:** la hoja creada tendrá el mismo nombre que la original, con un subíndice = 2  
Si la hoja activa se llama Enero, la copia se llamará Enero(2)

**Ejemplo 2:** se copia una hoja de un libro en un libro nuevo, a continuación del total de hojas:

```
Sub copiaHoja2()  
'se abre un nuevo libro  
Workbooks.Add  
'se activa el anterior, es decir el libro con datos a copiar  
ActiveWindow.ActivatePrevious  
'se copia la Hoja6 en el nuevo libro, a continuación del total de hojas  
Sheets("Hoja6").Copy After:=Workbooks(2).Sheets(Workbooks(2).Sheets.Count)  
'se cierra el libro nuevo (opcional)  
'ActiveWorkbook.Close True  
'o se activa nuevamente el libro original  
ActiveWindow.ActivatePrevious  
End Sub
```

**Nota:** Utilice la pestaña **Buscar** de este manual para encontrar más ejemplos de copiado de hojas en los otros capítulos.

## 10.14 Ocultar hojas

Hay 2 maneras de ocultar una hoja: con la propiedad **Visible** con valor **False** o con valor **xlVeryHidden**

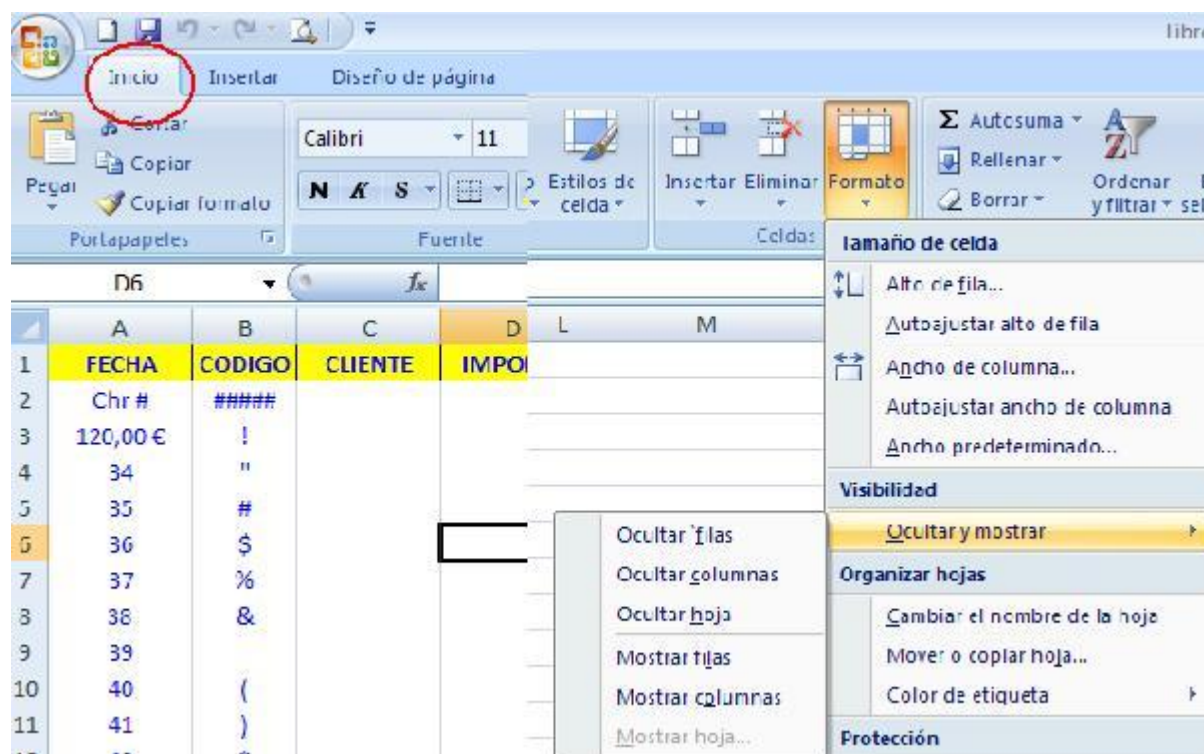
### Ejemplo 1:

`ActiveSheet.Visible = False` 'oculta la hoja activa

`Sheets("Hoja2").Visible = False` 'oculta la hoja de nombre 'Hoja2'

**Nota:** con estas instrucciones las hojas pueden mostrarse con la opción del menú **Formato, Hojas, Mostrar**

**En versión Excel 2007:** Para mostrar una hoja, desde la opción **Insertar** de la cinta de opciones, grupo **Formato**, en **Visibilidad** aparecen las opciones de **Ocultar y mostrar**



### Ejemplo 2:

`ActiveSheet.Visible = xlVeryHidden` 'oculta la hoja activa

`Sheets("Hoja2").Visible = xlVeryHidden` 'oculta la hoja de nombre 'Hoja2'

**Nota:** estas hojas solo podrán volver a verse utilizando una instrucción en macro colocando nuevamente la propiedad Visible en **True** (\*Ver tema siguiente)

## 10.15 Mostrar hoja oculta

Según hemos visto en el tema anterior, este método muestra cualquier hoja que fue ocultada, tanto con su propiedad **Visible** en **False** como en **XIVeryHidden**

```
Sheets("Hoja2").Visible = True
```

## 10.16 Establecer area visible de una hoja

Aquí lo que pretendemos es establecer el área visible de una hoja, es decir que no se pueda mover el cursor más allá de las celdas establecidas.

Colocaremos la instrucción en el objeto Hoja del Editor, que corresponda a la hoja que tratamos de personalizar, en el evento Activate. De esta manera cada vez que seleccionamos esa hoja se establece el área de acceso con la propiedad **ScrollArea**.

```
Private Sub Worksheet_Activate()  
ActiveSheet.ScrollArea = "A1:O30"  
End Sub
```

En este ejemplo solo se podrán seleccionar las celdas en el rango que va de A1 hasta O30.

**Nota:** También puede establecerse por única vez en el evento Open del libro, de la siguiente manera:

```
Private Sub Workbook_Open()  
Sheets(1).ScrollArea = "A1:O30"  
End Sub
```

**IMPORTANTE:** Para **restablecer** el área visible de una hoja quitando la restricción que establecimos en el punto anterior, colocaremos esta instrucción en el evento **Deactivate** de la hoja o en un módulo si lo queremos hacer de manera manual.

```
Private Sub Worksheet_Deactivate()  
ActiveSheet.ScrollArea = ""  
End Sub
```



# Capítulo

XI

## 11 11 - Trabajando con Celdas y Rangos

### 11.1 General

La hoja de Excel se compone de **Celdas o Rangos**, es decir un conjunto de celdas.

Existen muchas maneras de seleccionar, en una macro, una celda o rango de celdas.

Dentro de este capítulo encontrarán las formas más usuales para hacer referencia a un conjunto de celdas.

#### **Por ejemplo:**

`Range("B:B")`, `Range("A1:C10")`, `Cells(10,2)`, etc.

Luego en los capítulos siguientes, se utilizan indistintamente estas instrucciones que se detallan en los primeros puntos de este capítulo.

### 11.2 Principales Metodos y Propiedades de Rangos

Un rango es un conjunto (de 1 o más elementos) de celdas, es decir que puede ser tanto una sola celda como la hoja completa.

(ver descripción e imagen en cap 4: [Eventos, Métodos y Propiedades](#))

#### Métodos más comunes

<b>Activate</b>	Activa el Rango seleccionado
<b>AddComent</b>	Agrega un comentario al rango
<b>AdvancedFilter</b>	Aplica Filtro Avanzado
<b>AutoFill</b>	Rellena las celdas del rango especificado
<b>AutoFilter</b>	Aplica Autofiltro
<b>AutoFit</b>	Autoajusta filas o columnas
<b>Calculate</b>	Recalcula
<b>Clear</b>	Borra los valores del Rango (con las variantes del menú Edición, Borrar)
<b>Copy</b>	Copia los datos del Rango
<b>Delete</b>	Elimina el Rango
<b>Find</b>	Busca información en un rango devolviendo la primer celda donde lo encuentre
<b>Insert</b>	Inserta celdas
<b>PasteSpecial</b>	Pegado especial (Valores, Fórmulas, Formato, etc...).
<b>Select</b>	Para seleccionar un Rango
<b>Sort</b>	Para ordenar un rango en base a un criterio determinado

### Propiedades más comunes

<b>Address</b>	Indica la dirección del rango seleccionado.
<b>Areas</b>	Número de Rangos Seleccionados
<b>Column</b>	Posición de columna en la que empieza el Rango
<b>Columns</b>	Columnas que componen un rango
<b>Count</b>	Cuenta el número de objetos de un conjunto.
<b>CurrentRegion</b>	La región actual es un rango limitado por cualquier combinación de filas y columnas vacías. Equivale a presionar <b>Shift+Control+*</b> sobre una celda
<b>End</b>	Representa la celda situada al final de la región que contiene el rango fuente.
<b>EntireColumn</b>	Representa toda la columna a la cual pertenece un rango
<b>EntireRow</b>	Representa toda la fila a la cual pertenece un rango
<b>Font</b>	Representa la fuente del Rango
<b>Formula</b>	Fórmula de las celdas de un rango
<b>HasFormula</b>	Verdadero si la celda o rango tiene una fórmula, falso en caso contrario
<b>Offset</b>	Permite desplazarse en forma relativa con respecto a una celda o un rango
<b>Resize</b>	Permite redefinir el tamaño de un rango
<b>Row</b>	Posición de fila en la que empieza el Rango
<b>Rows</b>	Filas que componen un rango
<b>Text</b>	Texto contenido en las celdas de un rango
<b>Value</b>	Valor contenido en las celdas de un rango

En las hojas siguientes encontrarán ejemplos del uso de cada uno de estos temas.

## 11.3 Selección de Celdas o Rangos

Estos son ejemplos de cómo seleccionar celdas o rangos en VBA:

```

Range("B7").Select      'selecciona la celda B7

Range("A:A, D:F").Select  'selecciona las columnas A, D, E y F

Range("2:2, 4:7").Select  'selecciona las filas 2 y desde la 4 hasta la 7.

Range("A4:A10, D10, B5:B20").Select  'selecciona rangos discontinuos

Cells(10,5).Select      'selecciona la celda E10.

miRgo = "A2:B5"
Range(miRgo).Select      'selecciona el rango asignado en la variable

```

**Atención:** Nótese que en **Cells** los argumentos se ordenan por fila, columna, mientras que en **Range** se escribe primero la columna luego la fila.

## 11.4 Selección de rango utilizando variables

**Ejemplo 1:** la fila a seleccionar es el valor de cierta celda

```
Sub seleccion1()  
Dim filalibre As Integer  
'en la variable se guarda el contenido de la celda B3, que será un número de fila, ej: 5  
filalibre = Range("B3")  
  
'selecciona la celda cuya fila será el valor que contiene la celda B3  
Range("A" & filalibre).Select  
End Sub
```

**Ejemplo 2:** la última fila del rango a seleccionar es el valor de cierta celda

```
Sub seleccion2()  
Dim mirango As String  
Dim filalibre As Integer  
filalibre = Range("A1").Value  
  
'suponiendo que el contenido de A1 sea 10, se establece como rango desde B2 hasta E10  
mirango = "B2" & ":" & "E" & filalibre  
Range(mirango).Select  
End Sub
```

**Ejemplo 3:** la última columna del rango a seleccionar es el valor de cierta celda

```
Sub seleccion3()  
Dim columna1 As Integer  
columna1 = Range("F5").Value  
  
'suponiendo que el contenido de F5 sea 10, se establece como rango desde la columna 2 hasta la 10  
Range(Cells(2, 2), Cells(5, columna1)).Select  
End Sub
```

**Ejemplo 4:** la última columna del rango a seleccionar es el número de columna de la celda actual

```
Sub seleccion4()  
Dim mirango As String  
Dim columna1 As Integer  
columna1 = ActiveCell.Column  
  
'suponiendo que la columna actual sea 10, se establece como rango desde B2 hasta E10  
mirango = "B2" & ":" & "E" & columna1  
Range(mirango).Select  
End Sub
```

## 11.5 Seleccionar celdas a cierta distancia de la celda activa

La expresión '**Offset(fila, col)**' significa un **desplazamiento**, a tantas filas y tantas columnas, a partir de la celda activa.

**Activecell.Offset(1,0)** : nos posiciona en la misma columna (desplazamiento = 0) pero 1 fila hacia abajo (desplazamiento = 1) con respecto a la celda activa.

**Ejemplo 1:** Partiendo de A2 y con un desplazamiento de 3 filas y 2 columnas, la nueva celda seleccionada será C5

*Cells(2,1).Offset(3,2).Select*

**Ejemplo 2:** selecciona la celda D3, es decir 2 filas hacia abajo y 3 columnas a la derecha de la celda activa

*Sheets(1).Range("A1").Offset(2,3).Select*

**Ejemplo 3:** selecciona la celda que se encuentra 10 filas por encima y 1 columna a la izquierda de la celda activa.

*ActiveCell.Offset(-10,-1).Select*

## 11.6 Seleccionar la región donde se encuentra la celda activa

Si por ejemplo tenemos una lista que va desde B2 hasta H20, la instrucción siguiente selecciona el **rango completo**

*Range("B2").CurrentRegion.Select*

Una región comprende celdas aledañas hasta encontrar celdas vacías, tanto hacia la derecha como hacia abajo.

	A	B	C	D	E	F	G	H
1								
2		enero	10	20	30	40		enero
3		febrero	15	25	35	45		febrero
4		marzo	20	30	40	50		marzo
5		abril	25	35	45	55		abril
6		mayo	30	40	50	60		mayo
7		junio	35	45	55	65		junio
8								
9		Totales	135	195	255	315		
10								

## 11.7 Seleccionar hasta la última celda vacía -Fin de rango

Estos ejemplos **seleccionan** las celdas hacia abajo, derecha, izquierda o arriba hasta la última celda con datos dentro del rango.

**`Range("B2", Range("B2").End(xlDown)).Select`**

*'selecciona desde B2 hacia abajo*

**`Range("B2", Range("B2").End(xlToRight)).Select`**

*'selecciona desde B2 hacia la derecha*

**`Range("D2", Range("D2").End(xlToLeft)).Select`**

*'selecciona desde D2 hacia la izquierda*

**`Range("B2", Range("B20").End(xlUp)).Select`**

*'selecciona desde B20 hacia arriba, hasta la B2*

**Nota:** Las instrucciones anteriores presentan un problema cuando el rango está vacío. Por ejemplo, para el primer caso, si la celda A3 está vacía (en una tabla que todavía no se comenzó a cargar), se seleccionará toda la columna. Para evitar este error, se deberán agregar las siguientes líneas contemplando esta situación con una instrucción **IF**:

**`If Range("A2").Offset(1, 0) <> "" Then`**

*'si la celda A3 no está vacía se seleccionará hasta la última con datos*

**`Range("A2", Range("A2").End(xlDown)).Select`** *'selecciona desde A2 hacia abajo*

**`Else`**

*'se seleccionará solamente la celda inicial*

**`Range("A2").Select`**

**`End If`**

Otra opción para evitar este posible error, es utilizar la instrucción:

**`Range("B4", Range("B65536").End(xlUp)).Select`**

**Atención: en la versión Excel2007 podemos utilizar:**

**`Range("B4", Range("B1048576").End(xlUp)).Select`**

## 11.8 Obtener primer fila libre

Podemos obtener la **primer fila libre**, según una columna, con esta instrucción:

*`miFila = Range("A65536").End(xlUp).Row + 1`*

Con esta instrucción se recorre la **col A** desde de la última fila según versiones anteriores a **Excel 2007**

**Atención: en la versión Excel2007 podemos utilizar:**

***Range("A1048576").End(xlUp).Row + 1***

## 11.9 Obtener ultima columna con datos

De la misma manera que obtenemos la primer fila libre (o última fila con datos) también podemos obtener la **última columna** con datos de cierta fila.

**Ejemplo:** Obtener la última columna con datos en fila 10

```
Sub ultimacol()
Dim Col As Integer
'se recorre la fila 10 comenzando desde la última columna de la hoja (*)
Col = ActiveSheet.Range("IV10").End(xlToLeft).Column
'se muestra el nro en un cuadro de mensaje
MsgBox Col
End Sub
```

**Nota:** sumar 1 a la variable para obtener la primer columna libre

**Atención (\*) :** La referencia **IV** es la última columna en versiones anteriores a **Excel 2007**

**En versión Excel 2007, se puede utilizar la última col, es decir: XFD**

## 11.10 Obtener la dirección de una celda y guardarla en variable

La expresión **'Address'** nos devuelve la **dirección** de cierta celda

```
Dim MiDire As String
MiDire= ActiveCell.Address           'guarda la dirección absoluta como $B$5

MiDire= ActiveCell.Address(false, false)  'guarda la dirección relativa como B5
```

## 11.11 Borrar o Limpiar celdas o rangos

Utilizaré las distintas formas de selección vistas anteriormente para estos ejemplos que nos permiten **borrar celdas o rangos**, mediante el método **Clear** en todas sus variantes:

### Ejemplo 1:

```
Range("B5:E10").Select
Selection.Clear 'borra todo, contenido y formatos
```

### Ejemplo 2:

```
Dim MiRango as string
MiRango= "A2:A10"
'borra solo contenido, al igual que la tecla Supr
Range(MiRango).ClearContents
```

### Ejemplo 3:

```
Range("A2:A5, C2, F3").ClearFormats
'borra solo formatos de las celdas A2 hasta A5, C2 y F3
```

### Ejemplo 4:

```
Range("A1").CurrentRegion.ClearComments
'borra solo comentarios en la región A1
```

## 11.12 Eliminar celdas o rangos

Las siguientes instrucciones utilizando el método **Delete**, nos permiten **eliminar celdas o rangos**, con las opciones que encontramos desde el menú Edición, Eliminar.....

**Ejemplo 1:** eliminar celdas, desplazando las demás hacia arriba

La siguiente rutina ejecutada sobre la primer tabla dará como resultado, la siguiente.

```
Sub HaciaArriba()
' se selecciona el rango (en el ejemplo las celdas coloreadas)
Range("B6:D6").Select
' se elimina la selección desplazando las demás hacia arriba
Selection.Delete Shift:=xlUp
End Sub
```

	A	B	C	D	E	F	G	H
1								
2	enero	10	20	30	40	50	60	
3	febrero	20	30	40	50	60	70	
4	marzo	30	40	50	60	70	80	
5	abril	40	50	60	70	80	90	
6	mayo	50	60	70	80	90	100	
7	junio	60	70	80	90	100	110	
8	julio	70	80	90	100	110	120	
9								
10								

	A	B	C	D	E	F	G	H
1								
2	enero	10	20	30	40	50	60	
3	febrero	20	30	40	50	60	70	
4	marzo	30	40	50	60	70	80	
5	abril	40	50	60	70	80	90	
6	mayo	60	70	80	80	90	100	
7	junio	70	80	90	90	100	110	
8	julio				100	110	120	
9								
10								

**Ejemplo 2:** eliminar celdas desplazando las demás hacia la izquierda  
Observar en la segunda tabla el resultado de la rutina:

*Sub HaciaIzquierda()*

*'se selecciona el rango (en el ejemplo las celdas coloreadas)*

*Range("C4:C6").Select*

*'se elimina la selección desplazando las demás hacia la izquierda*

*Selection.Delete Shift:=xlToLeft*

*End Sub*

	A	B	C	D	E	F	G	H
1								
2	enero	10	20	30	40	50	60	
3	febrero	20	30	40	50	60	70	
4	marzo	30	40	50	60	70	80	
5	abril	40	50	60	70	80	90	
6	mayo	50	60	70	80	90	100	
7	junio	60	70	80	90	100	110	
8	julio	70	80	90	100	110	120	
9								
10								

	A	B	C	D	E	F	G	H
1								
2	enero	10	20	30	40	50	60	
3	febrero	20	30	40	50	60	70	
4	marzo	30	50	60	70	80		
5	abril	40	60	70	80	90		
6	mayo	50	70	80	90	100		
7	junio	60	70	80	90	100	110	
8	julio	70	80	90	100	110	120	
9								
10								

## 11.13 Eliminar varias filas segun condicion

Con la siguiente rutina se pretende **eliminar una cantidad x de filas** si se cumple cierta **condición**.

En el ejemplo se compara si las celdas C32 y D32 son = 0. En caso afirmativo se eliminan 7 filas (**Observar la selección utilizando Offset(-6,0) )**

```
Sub macroElimina7()
If Range("C32").Value = 0 And Range("D32").Value = 0 Then
Range(Range("C32"), Range("C32").Offset(-6, 0)).EntireRow.Select
Selection.Delete
Range("C32").Select
End If
End Sub
```

## 11.14 Insertar Filas

A continuación algunos ejemplos de cómo **insertar filas** en una hoja, utilizando las distintas formas de selección vistas anteriormente. El método es **Insert**

```
ActiveSheet.Range("A15").EntireRow.Insert
' en la hoja activa, inserta 1 fila por encima de A15
```

```
Sheets("Hoja3").Range("A10").EntireRow.Insert
' inserta en la Hoja3, una fila por encima de A10
```

```
Selection.EntireRow.Insert
' inserta 1 fila por encima de la celda seleccionada
```

```
ActiveSheet.Range("3:3").EntireRow.Insert
' inserta por encima de la fila 3, en hoja activa
```

## 11.15 Eliminar Filas

Distintos ejemplos para **eliminar filas**, utilizando las distintas formas de selección vistas anteriormente y el método **Delete**

```
ActiveSheet.Range("C15").EntireRow.Delete
' elimina la fila 15 de la hoja activa
```

```
Sheets("Hoja2").Range("H10").EntireRow.Delete  
'elimina la fila 10 de la Hoja 2
```

```
Selection.EntireRow.Delete  
'elimina la fila que se encuentra por encima de la selección actual
```

```
ActiveSheet.Range("10:10").EntireRow.Delete  
'elimina la fila 10 de hoja activa
```

## 11.16 Ocultar filas

Distintas maneras de **ocultar filas**, utilizando las distintas formas de selección vistas anteriormente y el método **Hidden**

```
ActiveCell.EntireRow.Hidden=True  
'oculta la fila de la celda activa
```

```
Rows(14).Hidden = True  
'oculta la fila 14
```

```
Cells(24, 1).EntireRow.Hidden = True  
'oculta la fila 24
```

## 11.17 Mostrar filas

La expresión '**Hidden**' significa **Oculto**, por eso al colocarla como **False** obtenemos el estado contrario, es decir '**mostramos**' la fila.

```
Rows(14).Hidden = False      'muestra la fila 14
```

## 11.18 Insertar Columnas

Varios ejemplos para **insertar columnas**, utilizando las distintas formas de selección vistas anteriormente y el método **Insert**

```
ActiveSheet.Range("A15").EntireColumn.Insert
```

'inserta 1 columna por delante de la columna A

```
Sheets("Hoja2").Range("D10").EntireColumn.Insert
```

'inserta 1 columna por delante de D en Hoja2

```
Selection.EntireColumn.Insert
```

'inserta 1 columna por delante de la celda seleccionada

```
ActiveSheet.Range("C:E").EntireColumn.Insert
```

'inserta 3 columnas por delante de C en hoja activa

## 11.19 Eliminar columnas

Varios ejemplos para **eliminar columnas**, utilizando las distintas formas de selección vistas anteriormente. El método es **Delete**

```
ActiveSheet.Range("A15").EntireColumn.Delete
```

'elimina la col A en hoja activa

```
Sheets("Hoja2").Range("D10").EntireColumn.Delete
```

'elimina la col D en Hoja2

```
Selection.EntireColumn.Delete
```

'elimina la columna activa

```
ActiveSheet.Range("C:E").EntireColumn.Delete
```

'elimina las col C, D y E en hoja activa

## 11.20 Ocultar columnas

Distintas maneras de **ocultar columnas**, utilizando el método **Hidden** y las distintas formas de selección vistas anteriormente:

```
ActiveCell.EntireColumn.Hidden=True  
'oculta la columna de la celda activa
```

```
Cells(24,3).EntireColumn.Hidden = True  
'oculta la columna 3
```

```
Range("D:E").EntireColumn.Hidden = True  
'oculta las columnas D y E
```

```
Columns(8).Hidden = True  
'oculta la columna H
```

**Atención:** con la última instrucción tomará la col H (8) .Aunque haya otras ocultas, igual se cuentan al momento de calcular cuál es la columna 8

## 11.21 Mostrar Columnas

Varios ejemplos para **mostrar columnas** ocultas, utilizando las distintas formas de selección vistas anteriormente:

```
Cells(24,3).EntireColumn.Hidden = False  
'muestra la columna 3
```

```
Range("D:F").EntireColumn.Hidden = False  
'muestra las columnas D, E y F
```

```
Columns(8).Hidden = False  
'muestra la columna H
```

**Atención:** con la última instrucción tomará la col H (8) .Aunque haya otras ocultas, igual se cuentan al momento de calcular cuál es la columna 8

## 11.22 Capturar fecha y hora de carga de datos

Suponiendo que se ingresarán datos en la columna A, se desea **guardar la fecha y hora de carga** en las columnas C y D respectivamente.

Para ello utilizaremos la siguiente rutina, que será colocada en el objeto **Hoja** donde se ingresarán los datos.

**Nota:** Este ejemplo sirve para cualquier columna de datos, evaluando que fecha y hora se colocarán a 2 y 3 col de distancia. **Ajustar los nros en variables colDato y colFecha**

```
Private Sub Worksheet_Change(ByVal Target As Range)
Dim colDato As Integer, colFecha As Integer, col As Integer

'guarda el número de col donde se ingresan los datos y la fecha
colDato = 1
colFecha = 3

'guarda la distancia que habrá entre las 2 columnas
col = colFecha - colDato

'controla si el dato ingresado fue en la columna 1
If Target.Column = colDato Then

'coloca la fecha en 2 col a la derecha del dato ingresado, es decir en la col 3
Target.Offset(0, col).Value = Date

'coloca la hora en 1 col más a la derecha de la fecha
Target.Offset(0, col + 1).Value = Time()
End If

End Sub
```

## 11.23 Ordenar un rango

### Uso de Sort:

**Ejemplo 1-** Esta rutina **ordena** una tabla, de una o más columnas, por la columna A

```
Sub Ordena1()
'se trabaja sobre la col A desde la fila 2
ActiveSheet.Range("A2").Select

'se ordena el rango por la col 1, en forma ascendente
Selection.Sort key1:=Range("A2"), order1:=xlAscending, Header:=xlYes, ordercustom:=1,
MatchCase:=False, Orientation:=xlTopToBottom
End Sub
```

**Ejemplo 2-** La siguiente rutina **ordena** una tabla, por 2 criterios, Key1 y Key2

```
Sub Ordena2()  
'se trabaja sobre el rango L2:N11  
ActiveSheet.Range("L2").Select  
  
'se ordena el rango por la col L, luego por la col M, en forma ascendente  
Range("L2:N11").Sort Key1:=Range("L2"), Order1:=xlAscending, Key2:=Range( _  
    "M2"), Order2:=xlAscending, Header:=xlGuess, OrderCustom:=1, MatchCase _  
    :=False, Orientation:=xlTopToBottom  
End Sub
```

## 11.24 Detectar si la celda contiene formula

Si se desea conocer si una **celda tiene fórmula** o valores, utilizar la propiedad: **HasFormula** que devolverá Verdadero o Falso

**Ejemplo:**

```
Sub TieneForm()  
If Range("A10").HasFormula Then MsgBox "Con Formula"  
End Sub
```

## 11.25 Resaltar la fila activa

En este ejemplo podemos ir **resaltando la fila** de la celda activa, quitando el color a la anterior que se ha coloreado.

La rutina se coloca en el objeto HOJA donde se hará la carga de datos.

```
Public fila As Integer  
  
Private Sub Worksheet_SelectionChange(ByVal Target As Range)  
'se controla si fila aún no tiene valor por ser la primera vez que se colorea 1 selección  
On Error Resume Next  
Cells(fila, 1).EntireRow.Interior.ColorIndex = xlColorIndexNone  
Target.EntireRow.Interior.ColorIndex = 3  
fila = Target.Row  
End Sub
```



## 11.26 Cambiar color de fuente a celdas con datos

Este ejemplo se encuentra desarrollado en el capítulo siguiente (***Instrucciones o comandos especiales***), tema: ***For Each...Next***



Roger Perez - Presenta

CURSO DE PROGRAMACION EN VBA

# Capítulo



XII

## 12 12 - Bucles= Comandos Especiales

### 12.1 General

En Visual Basic para Excel, contamos con algunas instrucciones especiales, que nos permiten realizar acciones generalmente *repetitivas*, llamadas **bucles**.

**Por ejemplo:**

- 1- repetir una acción hasta que se llegue a cierto valor
- 2- ejecutar instrucciones para todos los objetos que cumplan cierta condición.
- 3- ejecutar instrucciones mientras una variable no supere una condición
- ....

Se encontrará más ejemplos de estas instrucciones en los capítulos siguientes.

### 12.2 For Each .... Next

La expresión '**For Each Next**' permite recorrer todas las ocurrencias que componen un objeto.

**Sintaxis:**

<b>For Each</b> elemento <b>In</b> grupo	<i>'por cada elemento del grupo</i>
[instrucciones]	<i>'se ejecutan las instrucciones</i>
<b>[Exit For]</b>	<i>'opción de salir del bucle</i>
[instrucciones]	<i>'opción de ejecutar otras instrucciones</i>
<b>Next</b> [elemento]	<i>'se repite el ciclo para el elemento siguiente</i>

**Por ejemplo:** Recorrer todas las hojas de un libro (hoja/Worksheets), recorrer las celdas de un rango (celda/Range) o los libros abiertos (libro/Workbooks)

**Ejemplo:**

**Recorrer un rango** que previamente se habrá seleccionado. Pasar a color de fuente azul si la celda contiene datos.

```
Sub RecorreCeldas()  
Dim celda As Range  
Dim rango As Range
```

```
'se toma el rango seleccionado previamente  
Set rango = Selection
```

```
'por cada celda en el rango  
For Each celda In rango
```

```
'si la celda está vacía
If celda.Value = "" Then

'se coloca color de fuente automático
celda.Font.ColorIndex = xlAutomatic

'si la celda tiene datos
Else

'se coloca la fuente de color azul
celda.Font.ColorIndex = 5
End If

'se repite el bucle, con la siguiente celda del rango
Next

End Sub
```

**Nota:** con la pestaña 'Buscar' de este manual, podrán encontrar otros ejemplos del uso de este comando.

## 12.3 For ..... Next

La instrucción **For...Next** nos permite repetir unas instrucciones tantas veces como lo indiquemos en una variable denominada 'contador'

### Sintaxis

```
For contador = principio To fin [Step incremento]
[instrucciones]
Exit For
[instrucciones]
Next [contador]
```

Por ejemplo para copiar desde la fila 5 hasta la 10 haremos:

```
Dim i As Byte
For i = 5 To 10
```

También podemos indicar si deseamos realizar saltos. En el ejemplo anterior si quisiéramos copiar solo las filas impares tendríamos que agregar la expresión **Step**

```
Dim i As Byte
For i = 5 To 10 Step 2
```

Esto hará que copie las filas 5,7 y 9. En caso de no colocar *Step* se asume como valor predeterminado 1.

A continuación la rutina completa para **copiar filas impares** en otra hoja

```
Sub CopiaFilasImpares()
Dim i As Byte
Dim filalibre As Integer
'esta variable guarda la primer fila donde se copiarán los datos
filalibre = 20

'copia desde la fila 5 hasta la 10 en saltos de 2
For i = 5 To 10 Step 2
    Worksheets("Hoja1").Cells(i, 1).EntireRow.Copy
    Destination:=Worksheets("Hoja2").Cells(filalibre, 1)
    filalibre = filalibre + 1
Next
End Sub
```

## 12.4 While .... Wend

La instrucción **While...Wend** ejecuta una serie de instrucciones mientras una condición dada sea Verdadera

### Sintaxis:

```
While condición
[instrucciones]
Wend
```

### **Ejemplo:** Encontrar última celda con datos

```
Sub UltimoDato()
i = 1
'se recorre la col 1 desde la celda i=1 hasta que se encuentre una celda vacía
While ActiveSheet.Cells(i, 1).Value <> ""

'si no está vacía se pasa a la fila siguiente
ActiveSheet.Cells(i, 1).Select

'se incrementa el contador
i = i + 1

'se repite el bucle
Wend

'mensaje indicando la fila de la última celda
MsgBox i-1
End Sub
```

**Nota:** Este ejemplo es para mostrar el uso del bucle While-Wend. Para encontrar última celda con datos hay métodos más breves en el **cap. 11 'Obtener primer fila libre'**

## 12.5 If.... Elseif....Else....

La instrucción **If....Then....Else** ejecuta condicionalmente un grupo de instrucciones, dependiendo del valor de una expresión.

### Sintaxis:

```
If condición Then
[instrucciones]
ElseIf condición-n Then
[instrucciones_elseif] ...
Else
[instrucciones_else]
End If
```

**Ejemplo:** colocar valores en celdas según contenido de la celda activa

```
Sub comandos()
A = ActiveCell.Value
B = ActiveCell.Offset(0, 1)
C = ActiveCell.Offset(0, 2)
If A < 10 Then
A = A + 1
B = B + A
C = C + B
ElseIf A = 10 Then
B = A
C = B
Else
A = A - 1
End If
MsgBox A & B & C
End Sub
```

**Atención:** Para no extendernos en tantas líneas se pueden escribir todas las instrucciones de una misma condición separadas por dos puntos (:)

```
If A < 10 Then
A = A + 1 : B = B + A : C = C + B
ElseIf.....
```

## 12.6 Do While .... Loop

La instrucción **Do.....Loop** repite un bloque de instrucciones cuando una condición es Verdadera o hasta que una condición se convierta en Verdadera.

Hay 2 maneras de utilizar este bucle: con **While** (mientras sea verdadero) o **Until** (hasta que sea verdadero)

### Sintaxis:

```
Do [{While | Until} condición]
[instrucciones]
[Exit Do]
[instrucciones]
Loop
```

O bien, puede utilizarse esta sintaxis:

```
Do
[instrucciones]
[Exit Do]
[instrucciones]
Loop [{While | Until} condición]
```

**Ejemplo:** obligar al ingreso de un nombre en un InputBox

```
Sub IngreseClave()
Do
miNombre = InputBox("Por favor, ingrese su nombre:", "DATOS PERSONALES")

'se repite el bucle 'mientras' el contenido del InputBox sea 0
Loop While Len(miNombre) = 0 'obliga a ingresar dato en este inputbox
End Sub
```

## 12.7 Do Until.... Loop

Hemos visto la sintaxis de este **bucle** en el tema anterior.

Veremos aquí 2 ejemplos más:

### Ejemplo 1:

Se recorre una col hasta encontrar un dato y se devuelve su dirección

```
Sub BuscaTexto()
Dim i As Integer
i = 1
```

*'se comienza el ciclo desde la celda C1, hasta encontrar el texto*

**Do Until** Cells(i, 3) = "Vendido"

*i = i + 1*

**Loop**

*'se devuelve el número de fila del texto encontrado*

MsgBox "El texto 'Vendido' se encontró en la fila " & i

End Sub

### **Ejemplo 2:**

Se recorre una lista **desde la primer celda vacía** y se coloca el texto '**Anulado**'.

El bucle se repite hasta que se encuentra una celda con datos (*not IsEmpty*)

*Sub RellenarVacias()*

Range("F3").Select

*'se asume que la primer celda será vacía, si no lo es finaliza la rutina*

If ActiveCell.Value <> "" Then Exit Sub

*'comienza el bucle*

**Do**

*'se coloca el texto en la celda activa*

ActiveCell.Value = "Anulado"

*'se pasa a la siguiente fila*

ActiveCell.Offset(1, 0).Select

*'se repite hasta encontrar una celda no vacía*

**Loop Until Not IsEmpty**(ActiveCell)

End Sub

## 12.8 Uso de SET

La instrucción **SET** asigna una referencia de objeto a una variable o propiedad. Se utiliza para crear un nuevo objeto.

**Ejemplo:** creamos un objeto resultado de una búsqueda.

En este ejercicio se trata de encontrar en rango C3:C65500 de Hoja3, el valor que guarda la celda B2 de la Hoja2. Si el valor es encontrado indicará en qué fila se encuentra.

*Sub buscando()*

Dim quebusco As String

Dim busca As object

*'defino el valor a buscar*

quebusco = Sheets("Hoja2").Range("B2")

*'la búsqueda se realiza sobre la columna C de la Hoja3*

```
Set busca = Sheets("Hoja3").Range("C5:C65500").Find(quebusco, LookIn:=xlValues,
Lookat:=xlWhole)
```

```
'consultamos si la búsqueda ha sido exitosa,
'la traducción sería: si el objeto 'busca' No es vacío
If Not busca Is Nothing Then
```

```
'muestra un mensaje indicando la fila del dato encontrado
MsgBox "Valor encontrado en fila " & busca.Row"
```

```
else
'es decir no se encontró el valor
MsgBox "NO se encontró valor"
```

```
End If
```

```
'se libera el objeto
Set busca = Nothing
```

```
End Sub
```

### Otros ejemplos:

```
Dim miRango as Range
Set miRango= Columns("C:C") 'columna C
Set miRango= Rows(10)      'fila 10
```

## 12.9 With....End With

La instrucción **With** ejecuta una serie de instrucciones sobre un único objeto o sobre un tipo definido por el usuario.

### Sintaxis:

```
With objeto
[instrucciones]
End With
```

**Ejemplo 1 :** el objeto es un rango de una hoja

```
.....
Range("B2:D10").Select
With Selection
    .Font.Bold = True           'formato negrita
    .Font.Italic = True        'formato cursiva
End With
.....
'la rutina continúa
```

**Ejemplo 2:** el objeto se ha creado con la instrucción SET

```
.....  
Set wb = ActiveWorkbook 'se crea un nuevo libro  
With wb  
.SaveAs ThisWorkbook.Path & "\" & "LibroNvo.xls" 'ajustar la extensión tratándose de  
versión Excel 2007  
.Close  
End With  
.....  
'la rutina continúa
```

**Nota:** Nótese el uso de **punto** reemplazando al **objeto** en las instrucciones que se encuentran entre **With** y **End With**

**Atención:** Ver más ejemplos desde la pestaña '**Buscar**' de este manual.

## 12.10 Uso de Select Case

Un método para optimizar la escritura de código y por ende hacerlo más comprensible, es el uso de la instrucción **Select Case** en lugar de **If... else**, ejecutando uno de varios grupos de instrucciones, dependiendo del valor de una expresión.

**Ejemplo 1:** primero veremos un bucle utilizando **If... else** para determinar, sobre un rango seleccionado, el color de celda según su valor.

```
Sub ColorCelda1()  
For Each celda In Selection  
If celda.Value < 0 Then  
celda.Font.ColorIndex = 3  
ElseIf celda.Value < 20 Then  
celda.Font.ColorIndex = 10  
ElseIf celda.Value < 100 Then  
celda.Font.ColorIndex = 7  
Else  
celda.Font.ColorIndex = 5  
End If  
Next  
End Sub
```

Lo mismo podemos obtener utilizando la instrucción **Select Case** :

```
Sub ColorCelda2()  
For Each celda In Selection  
Select Case celda.Value  
Case < 0  
celda.Font.ColorIndex = 3  
Case 0 To 19  
celda.Font.ColorIndex = 10  
Case 20 To 99  
celda.Font.ColorIndex = 7  
Case 100 To 199  
celda.Font.ColorIndex = 5  
Case Else  
celda.Font.ColorIndex = 5  
End Select  
Next  
End Sub
```

```
Case 0 To 19
celda.Font.ColorIndex = 10
Case 20 To 99
celda.Font.ColorIndex = 7
Case >= 100
celda.Font.ColorIndex = 5
End Select
Next
End Sub
```

Otros ejemplos para definir rangos:

```
Sub uso_de_select()
Select Case Activecell.value
Case "A", "C", "F"
'instrucciones si el valor de la celda es = A, C y F

Case "5" to "7"
'instrucciones si el valor de la celda va de 5 a 7 inclusive

Case Is = "Anulado"
'instrucciones si el valor de la celda es = 'Anulado'

'ver Ayuda VB para otros ejemplos

Case Else
'instrucciones para otros valores
End Select
End Sub
```

# Capítulo

A large, solid gray circle containing the Roman numeral 'XIII' in white, bold, sans-serif capital letters.

**XIII**

## 13 13- Trabajando con fórmulas

### 13.1 Trabajando con fórmulas

Para incluir fórmulas en celdas utilizaremos la propiedad '**Fórmula**' o '**FórmulaR1C1**'

La diferencia entre ambas es que la primera utiliza la notación A1 (letra de columna, número de fila). La segunda en cambio utiliza la notación R1C1 (fila (row) número, columna (col) número)

**Ejemplo:** si queremos realizar en la celda activa esta operación: A5+B5 podemos escribir la instrucción de estas 2 maneras:

```
ActiveCell.Formula = "= A5+B5"
```

```
ActiveCell.FormulaR1C1 = "= R5C1 + R5C2"
```

Para colocar un texto en la fórmula se lo colocará entre 2 juegos de comillas dobles.

**Ejemplo:**

```
Worksheets("Hoja1").Range("B1").Formula = "(A1 & ""OK"")"
```

(los paréntesis no son necesarios, se incluyen para diferenciar las comillas propias del texto)

### 13.2 Introducir fórmulas en celdas

Si la versión de Excel es en español y así aparecen las funciones en la barra de fórmulas, para colocarlas en el Editor es necesario conocer las versiones en inglés de las funciones a utilizar. Por eso lo mejor es aplicar la función en una celda con la grabadora de macros encendida. Al detenerla se encontrará la instrucción correcta en un módulo.

(\* Ver cap 2 '**Cómo programar en Excel? - Cómo crear una macro**'), y luego ajustar las referencias.

**Ejemplo 1:** Restar un rango de otro

```
Sub formula1()  
Range("K10").Formula = "=+SUM(A1:J10)-SUM(K11:K12)"  
End Sub
```

**Ejemplo 2:** Devolver valores con BuscarV

```
Sub formula2  
ActiveCell.Formula = "=+VLOOKUP(A2,Hoja1!A2:B25,2,FALSE)"  
End Sub
```

**IMPORTANTE:** la lista completa de funciones en Español e Inglés la encontrarán en el Manual: '**Excelencias**' de la misma autora, que se adjunta con el manual **400MacrosPlus**



# Capítulo

XIV

## 14 14 - Trabajando con Objetos Insertados en Hoja

### 14.1 Comentarios generales

Por **Objeto Insertado** hago referencia a los objetos de la **Barra de Herramientas Cuadro de Controles o Formulario**, como así también a los que se insertan con la **Barra de Dibujo** (Autoformas, Imagen, etc).

Estos objetos se denominan **Shapes o Pictures** para las imágenes. También veremos algunas rutinas para controles ubicados en un **Userform**.

#### En versión Excel 2007:

Desde ficha 'Insertar' de la **Cinta de Opciones** se pueden insertar objetos como Dibujos, Cuadros de texto, imágenes, etc.

Desde ficha '**Programador**', grupo **Controles**, botón **Insertar**, permite trabajar con controles de la barra Formulario o ActiveX



Las siguientes rutinas, son generales para cualquier control (**ComboBox, Listbox, Textbox**). Solo se deberá reemplazar la expresión 'ComboBox' por 'ListBox' o el que corresponda y respetar las propiedades de cada control.

**Importante:** debemos tener en cuenta *qué barra* utilizamos para *dibujar el control*

(En general en los siguientes ejercicios he utilizado la barra de herramientas: **Cuadro de controles**)

A continuación algunos ejemplos con aclaración del tipo de control:

*ActiveSheet.ComboBox1.Visible = True 'Cuadro de controles*

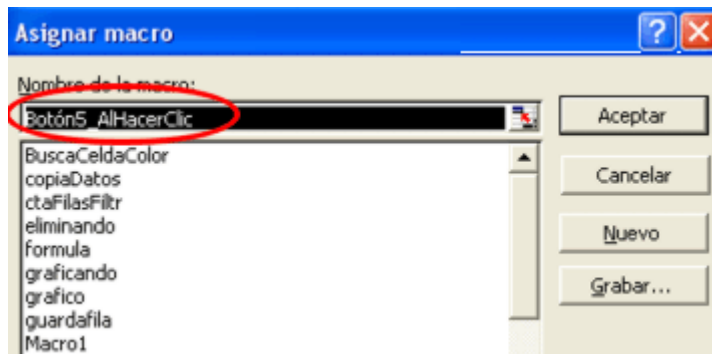
```
ActiveSheet.ListBox1.Visible = True      'Cuadro de controles
```

```
ActiveSheet.DropDowns("Lista desplegable 9").Visible=True      'Barra Formularios
```

```
ActiveSheet.Shapes("Cuadro de lista 8").Visible = False      'Barra Formularios
```

**Nota:** para conocer el nombre del objeto dibujado con la barra Formulario, seleccionar el mismo con clic derecho. A la derecha de la barra de fórmulas, aparecerá en el **Cuadro de nombres**, el nombre del control.

También haciendo clic derecho sobre el control, opción '*Asignar Macro*' se verá en la ventana emergente el nombre del control. Luego cancelar esta ventana.



## 14.2 Asignar rango a un Combobox

Aquí consideremos que en la Hoja 3 tenemos un **combobox** (dibujado con la barra '**Cuadro de controles**'), al que se le **asigna** por macro el **rango de valores**

```
Sub AsignaRango()  
'se selecciona el objeto  
Sheets(3).Combobox1.Select  
  
'se asigna el rango  
With Selection  
  .ListFillRange = "D1:D40"  
End With  
  
End Sub
```

### 14.3 Enviar valor de un Combo a una celda

Al insertar el **Combobox** (de la barra **Cuadro de controles**) en la hoja, y una vez asignada su propiedad **ListFillRange** (de donde tomará los datos), seleccionarlo con click derecho, opción *Ver código*.

En el Editor escribir estas instrucciones:

```
Private Sub ComboBox1_Change()  
Range("G3") = ComboBox1.Value  
End Sub
```

**Nota:** esta rutina es útil a la hora de **enviar datos del combo, con formato** (en el ejemplo: formato *mayúsculas*)

```
Private Sub ComboBox1_Change()  
'envía la opción seleccionada con formato Mayúsculas  
Range("G3") = UCase(ComboBox1.Value)  
End Sub
```

### 14.4 Enviar valores de Combo de 4 columnas a celdas

En este ejemplo tenemos un **Combobox** de 4 columnas.

Al seleccionar un valor en él, se **vuelcan estos valores a una fila de la hoja**.

Las instrucciones pueden ser para el evento **Change** del combo o también para el evento **Click**

```
Private Sub ComboBox1_Change()  
Dim fila As Integer, filalibre As Integer  
'filalibre es donde quieras volcar los datos  
filalibre = 22  
  
'guarda la fila seleccionada del combo  
fila = ComboBox1.ListIndex '(* ver Nota)  
'coloca en celdas el valor de cada col  
Cells(filalibre, 1).Value = ComboBox1.List(fila, 0)  
Cells(filalibre, 5).Value = ComboBox1.List(fila, 1)  
Cells(filalibre, 6).Value = ComboBox1.List(fila, 2)  
Cells(filalibre, 7).Value = ComboBox1.List(fila, 3)  
  
End Sub
```

**Nota:** la instrucción **ListIndex** devuelve el número de posición del registro seleccionado

## 14.5 Buscar valor del Combo en base Devolver otros datos en textbox

Esta rutina es una de las principales tareas que requerimos al utilizar **Combos**. Lo que se pretende es que se busque en una base el valor seleccionado en el combo y que devuelva (en celdas o en cuadros de texto) los otros campos del registro encontrado.

Al dibujar el control en la hoja, seleccionarlo con botón derecho y optar por 'Ver código' o 'Asignar Macro' según con qué barra se lo dibujó.

(\* Ver tema **Cómo ejecutar una macro**, del capítulo 5)

En este ejemplo, se busca en la **col A** de la **hoja activa** el valor seleccionado en un **combobox** y se devuelven los otros campos del registro encontrado en otros controles **Textbox** que se habrán dibujado en la hoja.

```
Private Sub ComboBox1_Change()  
Dim valor1 As String, rango As String  
Dim dato  
'guarda en la variable el valor actual del combo  
valor1 = ComboBox1.Value  
'define el rango donde se buscará  
rango = "A1:A30"  
'definir el nombre de hoja, sino será la hoja activa  
With ActiveSheet.Range(rango)  
Set dato = .Find(valor1, LookIn:=xlValues, Lookat:=xlWhole)  
If Not dato Is Nothing Then  
'devuelve en los textbox los datos de las 2 columnas siguientes  
TextBox1 = dato.Offset(0, 1).Value  
TextBox2 = dato.Offset(0, 2).Value  
End If  
End With  
End Sub
```

## 14.6 Limpiar un combo

Para borrar el contenido de un combo se utiliza la instrucción **Clear**

En el siguiente ejemplo se limpia el **combo** insertado en la hoja con la herramienta **Cuadro de controles**, del menú Ver

```
Sub limpiaCombo()  
ActiveSheet.ComboBox1.Clear  
End Sub
```

## 14.7 Funciones de validación

Los valores que se ingresen en controles **Textbox** son de formato **String**.

Si deseamos *validar* que estos datos sean *fechas, números, texto*, etc debemos utilizar ciertas **funciones de comprobación**.

**IsEmpty:** esta función nos permite evaluar si el contenido del control está vacío (en ese caso devolverá True) o no.

**IsNumeric:** devuelve un valor True si la expresión o el contenido del Textbox es un valor numérico, de lo contrario devolverá False

**Nota:** la expresión **Not** evalúa la condición opuesta a la función.

### Ejemplo:

```
If not IsEmpty (Activecell) Then
    MsgBox "La celda no está vacía"

    If not IsNumeric (Activecell.value) Then
        MsgBox "El dato no es numérico"
    else
        MsgBox "El dato es numérico"
    end if

else
    MsgBox "La celda está vacía = Empty"
End If
```

### Otras funciones:

**IsDate(expresión):** evalúa si la expresión corresponde a un formato fecha

**IsError(expresión):** evalúa si expresión devuelve algún tipo de error.

**IsArray(expresión):** comprueba si expresión corresponde a una matriz o no

**IsNull(expresión) :** evalúa si expresión contiene un valor nulo por datos no válidos.

**Nothing:** evalúa si una variable ha sido declarada o no.

### Ejemplo:

```
Sub evalua_variable()
    Dim valor as String

    If valor is Nothing Then
        MsgBox "La variable no ha sido declarada"
    else
        valor = Activecell.address
    End if
End Sub
```

## 14.8 Asignar formato moneda a un TextBox

Otro caso que requiere formato son los controles **Textbox** que reciben valores del tipo **moneda**.

Para asignar este formato, tendremos un control TextBox1 y esta rutina que se ejecuta al perder el foco o salir del control:

```
Private Sub TextBox1_LostFocus()  
    TextBox1.Text = Format(TextBox1.Text, "$ #.##0,00")  
End Sub
```

**Nota:** Para un control Textbox en un **Userform**, utilizar la siguiente rutina.

```
Private Sub TextBox2_Exit(ByVal Cancel As MSForms.ReturnBoolean)  
    TextBox2.Text = Format(TextBox2.Text, "$ #.##0,00")  
End Sub
```

## 14.9 Validar campos fecha en Textbox

Para evaluar si lo ingresado en un control **TextBox** corresponde a **fecha**, utilizaremos esta rutina asociada al control.

De esta manera, el usuario no podrá salir del control hasta ingresar una fecha o dejar el campo en blanco.

```
Private Sub TextBox1_LostFocus()  
    If IsDate(TextBox1.Text) = False And Len(TextBox1.Text) <> 0 Then  
        MsgBox "Debe ingresar una fecha"  
        TextBox1.Text = ""  
        TextBox1.Activate  
    End If  
End Sub
```

**Nota:** Para un control Textbox en un **Userform**, utilizar la siguiente rutina.

```
Private Sub TextBox1_Exit(ByVal Cancel As MSForms.ReturnBoolean)  
    If IsDate(TextBox1.Text) = False And Len(TextBox1.Text) <> 0 Then  
        MsgBox "Debe ingresar una fecha"  
        TextBox1.Text = ""  
        Cancel = True  
    End If  
End Sub
```

## 14.10 Validar campos numéricos en Textbox

Para evaluar si lo ingresado en un control **TextBox** corresponde a **número**, utilizaremos esta rutina asociada al control.

De esta manera, el usuario no podrá salir del control hasta ingresar un valor numérico o dejar el campo en blanco.

```
Private Sub TextBox1_LostFocus()  
    If IsNumeric(TextBox1.Text) = False And Len(TextBox1.Text) <> 0 Then  
        MsgBox "Debe ingresar un campo numérico"  
        TextBox1.Text = ""  
        TextBox1.Activate  
    End If  
End Sub
```

**Nota:** Para un control Textbox en un **Userform**, utilizar la siguiente rutina.

```
Private Sub TextBox2_Exit(ByVal Cancel As MSForms.ReturnBoolean)  
    If IsNumeric(TextBox2.Text) = False And Len(TextBox2.Text) <> 0 Then  
        MsgBox "Debe ingresar un campo numérico"  
        TextBox2.Text = ""  
        Cancel = True  
    End If  
End Sub
```



# Capítulo

XV

## 15 15 - Controlando Mensajes de Excel

### 15.1 General

Es usual que al ejecutar una rutina queremos evitar que aparezcan los **mensajes** propios de Excel, como el aviso de que el libro contiene vínculos, o ya se encuentra un libro de igual nombre al momento de guardarlo.

Las **instrucciones de control** pueden ser colocadas en cualquier rutina, antes de realizar una acción como Guardar, Eliminar hoja, o Abrir libros, y se ejecuta el **valor predeterminado** del cuadro de mensaje que envía Excel (*Ver más detalles en tema siguiente*)

**Por ejemplo:** al intentar eliminar una hoja Excel envía un mensaje de alerta con 2 opciones: Eliminar o Cancelar, siendo el predeterminado Eliminar. Esta es la opción que se ejecuta al agregar una instrucción que controla este tipo de mensajes

Otros ejemplos en los temas siguientes:

### 15.2 No mostrar avisos de alerta

Si se desean evitar los mensajes de alerta que Excel pueda enviar al ejecutar una macro, utilizar la propiedad **DisplayAlerts** con valor **False**

**Application.DisplayAlerts = False**

hará que no se muestre el aviso y si necesita una respuesta por parte del usuario Excel tomará su **valor predeterminado (Ver Nota)**

Por ejemplo, al eliminar una hoja el aviso presenta 2 opciones: Aceptar (predeterminado) y Cancelar. Utilizando esta propiedad se evita el aviso y se toma como respuesta Aceptar.

La rutina completa es la siguiente:

```
Sub eliminaHoja()
'evita que se exhiba el mensaje de alerta
Application.DisplayAlerts = False
'elimina la hoja
Sheets(3).Delete
'devolver a la propiedad su valor predeterminado
Application.DisplayAlerts = True
End Sub
```

**Nota:** *Ver excepción en tema siguiente*

### 15.3 No mostrar aviso, al guardar un archivo, de que el archivo ya existe:

Este ejemplo se utiliza en el objeto ***ThisWorkbook***.

Antes de cerrar el libro, se lo guardará como Libro1. Si ya existe un libro con ese nombre en la carpeta asignada, Excel lo sobrescribirá **sin** mostrar el mensaje de alerta

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
' se inhabilita la opción de mostrar el cuadro de Alerta
Application.DisplayAlerts = False
```

```
' se guarda el libro
ActiveWorkbook.SaveAs "C:\Mis documentos\Libro1.xls" 'ajustar la extensión
tratándose de versión Excel 2007
```

```
' se vuelve la aplicación al estado normal
Application.DisplayAlerts = True
```

```
End Sub
```

#### IMPORTANTE:

Cuando se utiliza el método **SaveAs** para sobrescribir los libros de un archivo existente, el valor predeterminado del mensaje "Sobrescribir" es "No"; sin embargo, cuando la propiedad **DisplayAlerts** se establece como **False**, Excel selecciona la respuesta "Sí".

### 15.4 No mostrar la ejecución de la macro o el movimiento de hojas:

Otro problema común al trabajar con varias hojas, es el vaivén o parpadeo al entrar y salir de las distintas hojas o de Formularios. Para evitar esto utilizamos la propiedad **ScreenUpdating** con valor **False**

```
Application.ScreenUpdating = False
```

```
'volverla a True al finalizar la macro
```



# Capítulo

XVI

## 16 16 - Controlando Errores

### 16.1 On Error Resume Next

Un buen programa debe controlar los posibles errores imprevistos que pueden aparecer al ejecutarse el mismo, y que permitan seguir o cancelar el proceso normalmente.

Por ejemplo, antes de la instrucción **Print** (imprimir) se debe agregar una instrucción que controle el error que puede producirse si el sistema no encuentra una impresora instalada.

Las principales instrucciones son las que veremos en este capítulo.

La sentencia **On Error Resume Next** permite ignorar un error y avanzar a la siguiente instrucción en la ejecución de un procedimiento.

#### **Ejemplo:**

```
Sub miMacro()  
  On Error Resume Next  
  Instrucción 1  
  Instrucción 2  
End Sub
```

El incluir la sentencia **On Error Resume Next** antes de la instrucción 1 implica que si se produce algún error el programa lo ignorará y continuará con la instrucción siguiente.

**Nota:** Es importante agregar esta sentencia antes de las instrucciones de impresión, ya que esas son con frecuencia las acciones que originan errores en las macros.

### 16.2 On Error GoTo ....

La sentencia **On Error GoTo.....** se utiliza para ir a una parte específica del procedimiento cuando se produce un error.

#### **Ejemplo:**

```
Sub miMacro1()  
  On Error GoTo final  
  Instrucción 1  
  Instrucción 2  
Exit Sub
```

```
final:  
[otras instrucciones]  
End Sub
```

En este caso, se utiliza la sentencia **On Error GoTo final**, lo que implica que si se produce un error, el procedimiento continuará ejecutándose a partir de la sentencia **final:** (no olvidar aquí los 2 puntos)

La instrucción **Exit Sub** permite salir del procedimiento en caso de no presentarse ningún error, para que no se ejecute lo que sigue a partir de la sentencia **final:**

También es posible que al presentarse un error solo queremos que la macro finalice. En ese caso no habrá instrucciones luego de la llamada y tampoco se necesitará entonces la expresión **Exit Sub**

```
Sub miMacro2()  
On Error GoTo final  
Instrucción 1  
Instrucción 2  
  
final:  
End Sub
```

## 16.3 On Error GoTo 0

La sentencia **On Error GoTo 0** se utiliza para volver al modo normal de manejo de errores.

### **Ejemplo:**

```
Sub miMacro3()  
On Error Resume Next  
Instrucción 1  
Instrucción 2  
On Error GoTo 0  
End Sub
```

En este caso una vez ejecutadas las instrucciones del código, se vuelve al modo normal de manejo de errores.



# Capítulo

XVII

## 17 17 - Uso de MsgBox e InputBox

### 17.1 Construcción de MsgBox

Los **MsgBox** crean una interfaz entre el usuario y el programa. Se utilizan para:

- 1- **Mostrar información**
- 2- **Mostrar información con datos del proceso**
- 3- **Recibir información del usuario**

#### Sintaxis para los casos 1 y 2:

***Msgbox "Mensaje", Botones/icons, "Título"***

**Mensaje:** Cualquier texto.

Para saltar a la próxima línea usamos el carácter **vbCrLf** o **chr(10)**

#### **Ejemplo:**

"Bienvenidos" & chr(10) & "al mundo de la programación"

**Botones:** Se pueden añadir cualquiera de estos cuatro botones (si se omiten Excel asume vbOkOnly por defecto).

- **vbOkOnly** 'predeterminado
- vbOkCancel
- vbYesNoCancel
- vbAbortRetryIgnore

**Iconos:** Se puede elegir entre los siguientes.

- vbCritical - vbQuestion - vbExclamation - vbInformation

**Título:** Cualquier texto.

#### **Ejemplo 1:**

**Msgbox "NO HAY STOCK SUFICIENTE"**

#### **Ejemplo 2:**

**Msgbox "El número de control es " & ActiveCell.value**

#### Sintaxis para Recibir información (caso 3)

***Respuesta = Msgbox("Mensaje", Botones/icons, "Título")***

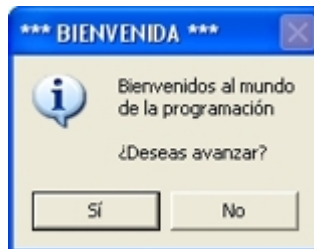
Si el programa utilizará la respuesta del usuario, estas son las cifras que devolverá de acuerdo a los botones que se hayan colocado:

Ok = 1 - Cancel = 2 - Abort = 3 - Retry = 4 - Ignore = 5 - Yes = 6 - No = 7

### Ejemplo 3:

```
Sub Mensajes()
Dim miOpcion As Byte
miOpcion = MsgBox("Bienvenid@ al mundo" & Chr(10) & _
"de la programación" & Chr(10) & Chr(10) & _
"¿Deseas avanzar?", vbInformation + vbYesNo, "*** BIENVENIDA ***")
If miOpcion = 6 Then
MsgBox "¡ Buena decisión !"
Else
MsgBox "Vuelve otro día! "
End If
End Sub
```

Nótese que en los últimos cuadros de mensaje no se colocaron botones sino que se dejó el predeterminado de Excel.



## 17.2 Construcción de InputBox

Los **InputBox** muestran un mensaje en un cuadro de diálogo, esperan que el usuario ingrese un dato o haga clic en un botón y devuelve un tipo *String* con el contenido del cuadro de texto.

### Sintaxis:

**InputBox(prompt[, title][, default][, xpos][, ypos][, helpfile, context])**

**Prompt:** expresión de cadena que se muestra como un mensaje de texto en un cuadro de diálogo

**Title** Opcional. Expresión de cadena que se muestra en la barra de título del cuadro de diálogo. Si omite **title**, en la barra de título se coloca el nombre de la aplicación.

**Default** Opcional. Expresión de cadena que se muestra en el cuadro de texto como respuesta predeterminada cuando no se suministra una cadena. Si omite **default**, se muestra el cuadro de texto vacío.

**xpos** Opcional. Expresión numérica que especifica, en twips, la distancia en sentido horizontal entre el borde izquierdo del cuadro de diálogo y el borde izquierdo de la pantalla. Si se omite **xpos**, el cuadro de diálogo se centra horizontalmente.

**ypos** Opcional. Expresión numérica que especifica, en twips, la distancia en sentido vertical entre el borde superior del cuadro de diálogo y el borde superior de la pantalla.

**Helpfile** Opcional. Expresión de cadena que identifica el archivo de Ayuda que se utilizará para proporcionar ayuda interactiva para el cuadro de diálogo. Si se especifica **helpfile**, también deberá especificarse **context**.

**Context** Opcional. Expresión numérica que es el número de contexto de Ayuda asignado por el autor al tema de Ayuda correspondiente. Si se especifica **context**, también deberá especificarse **helpfile**.

**Type** Variant opcional. Especifica el tipo de datos que se devuelve.

**Ejemplo:** solicitar una clave de ingreso

```
Sub SolicitoClave()  
Dim miClave as String  
miClave = InputBox("Ingrese Clave", "*** Solicitud ***")  
'si la clave es correcta  
If miClave = "Macros" then  
'se desproteje la Hoja2  
Sheets("Hoja2").Unprotect  
End If  
End Sub
```

En este ejercicio se desproteje la Hoja2 si la clave ingresada es 'Macros', de lo contrario no realiza ninguna acción.

## 17.3 Controlar que se ha ingresado un valor

Al utilizar **InputBox** para el ingreso de datos que luego serán utilizados en la rutina, es necesario verificar si no se ha presionado el botón '**Cancelar**' o si no se ha cerrado el cuadro sin ingresar valores.

La manera entonces de construir una rutina con **InputBox** puede ser como la siguiente, donde se tenga un bucle que no nos permita salir sin haber ingresado un valor (luego quedara controlar que ese valor sea correcto según nuestras necesidades)

```
Sub mensajes()  
Dim valor1  
'comienza el bucle  
Do  
valor1 = InputBox("Ingrese nro de mes", "Ingreso de datos")  
'si la variable no está vacía  
If valor1 <> "" Then  
'muestra un mensaje, opcional, y sale del bucle  
    MsgBox "correcto"  
Exit Do
```



```
End If  
'continúa el bucle  
Loop  
'siguen las instrucciones  
End Sub
```



Roger Perez - Presenta

CURSO DE PROGRAMACION EN VBA

# Capítulo

XVIII

## 18 18- Controlando Barras y Menú de Excel

### 18.1 Ocultar Barras de Desplazamiento

En este capítulo veremos cómo quitar algunos elementos de la hoja, que manualmente los quitamos desde el menú **Herramientas, Opciones, Ver**.

**En versión Excel 2007, se quitan o habilitan desde el botón Office, Opciones de Excel, Avanzadas.**

En este caso, veremos cómo ocultar las **Barras de Desplazamiento**.

```
Sub Quita_Desplazamiento
With ActiveWindow
'quitar barras de desplazamiento
.DisplayHorizontalScrollBar = False
.DisplayVerticalScrollBar = False
End With
End Sub
```

**Nota:** recordar de ejecutar la misma rutina con el valor en **True** para devolver a Excel su estado normal.

(ver aclaración en [Tema General](#), de este mismo cap.)

### 18.2 Ocultar Pestañas de hoja

La rutina que oculta las pestañas u hojas es la siguiente:

```
Sub Quita_Pestañas
ActiveWindow.DisplayWorkbookTabs = False
End Sub
```

**Nota:** recordar de ejecutar la misma rutina con el valor en **True** para devolver a Excel su estado normal.

(ver aclaración en [Tema General](#), de este mismo cap.)

## 18.3 Ocultar Encabezados de filas y col

La siguiente rutina quita los encabezados de filas (1-2-3....) y columnas (A-B-C....)

```
Sub Quita_Encabezados()  
ActiveWindow.DisplayHeadings = False  
End Sub
```

**Nota:** recordar de ejecutar la misma rutina con el valor en True para devolver a Excel su estado normal.

(ver aclaración en [Tema General](#), de este mismo cap.)

## 18.4 Ocultar Líneas de División

Quitando las líneas de división, la hoja se verá como un formulario. Luego será posible dar borde solo a las celdas importantes.

```
Sub Quita_Lineas()  
ActiveWindow.DisplayGridlines = False  
End Sub
```

**Nota:** recordar de ejecutar la misma rutina con el valor en True para devolver a Excel su estado normal.

(ver aclaración en [Tema General](#), de este mismo cap.)

## 18.5 Quitar barras frecuentes

Ahora veremos cómo quitar algunas barras. Las más frecuentes son: **Estándar, Formato, Fórmulas y Estado**. (**Ver Excel 2007**)

**Nota:** aquí se agruparon todas dentro del bucle [With...End With](#), visto en el **cap 12**.

```
Sub Quita_barras()  
With Application  
'barra Estándar (la de los botones Nuevo, Abrir, Guardar, etc)  
.CommandBars("Standard").Visible = False  
'barra Formato  
.CommandBars("Formatting").Visible = False  
'barra Fórmula
```

```
.DisplayFormulaBar = False  
'barra de Estado  
.DisplayStatusBar = False  
End With  
End Sub
```

**En versión Excel 2007, no se tienen las barras Estandar ni Formato, por lo que esas líneas de instrucción pueden ser quitadas.**

**Nota:** recordar de ejecutar la misma rutina con el valor en True para devolver a Excel su estado normal.

(ver aclaración en [Tema General](#), de este mismo cap.)

## 18.6 Aclaración General

A lo largo de este capítulo hemos visto **individualmente**, cómo **quitar barras o elementos** de la hoja Excel.

Lo más recomendable es que estas instrucciones se ejecuten en el evento **Open** del libro (con valor en False) y luego se restablezcan al momento de cerrar el libro, es decir en el evento **BeforeClose**.

Entonces lo mejor será tener una **única rutina** en un módulo con una **variable pública** a la que asignamos su valor en **False o True**, según desde que evento será llamada.

Los pasos a realizar serían estos:

**En un módulo escribir:**

### **Public estado**

```
Sub Elementos_Hoja()  
'quitar o mostrar elementos de la hoja  
With ActiveWindow  
'quitar barras de desplazamiento  
.DisplayHorizontalScrollBar = estado  
.DisplayVerticalScrollBar = estado  
'ocultar las pestañas  
.DisplayWorkbookTabs = estado  
'quitar encabezados de filas y col  
.DisplayHeadings = estado  
'las líneas de división  
.DisplayGridlines = estado  
End With  
  
'quitar barras
```

```
With Application
'barra Estándar (la de los botones Nuevo, Abrir, Guardar, etc)
.CommandBars("Standard").Visible = estado
'barra Formato
.CommandBars("Formatting").Visible = estado
'barra Fórmula
.DisplayFormulaBar = estado
'barra de Estado
.DisplayStatusBar = estado
End With

End Sub
```

Y en el objeto **ThisWorkbook** (O EsteLibro) escribir estas rutinas:

```
Private Sub Workbook_Open()
estado = False
Call Elementos_Hoja
End Sub

Private Sub Workbook_BeforeClose(Cancel As Boolean)
estado = True
Call Elementos_Hoja
End Sub
```



# Capítulo

XIX

## 19 19- Buscando-Comparando-Evaluando Datos

### 19.1 Devolver en una celda el resultado de una búsqueda

Este ejemplo busca un **valor** utilizando la función **FIND**.

Si el dato es encontrado devolverá en otra celda la ubicación del mismo, si no devolverá un mensaje.

El valor a buscar es el contenido de la celda **B2** de la hoja activa, y la búsqueda se realiza en una hoja llamada **Hoja3** dentro del rango **A1:B20**.

**Todos estos datos pueden ser modificados y adaptados a otros modelos.**

```
Sub buscador()
Dim mihoja As String, Donde As String
Dim Quebusco As String
Dim resulta As Object
Dim ubicado As String

'la variable mihoja guarda la hoja donde se hará la búsqueda
mihoja = "Hoja3"

'la variable Donde guarda el rango donde debe efectuarse la búsqueda
Donde = "A1:B20"

'la variable Quebusco guarda el dato a buscar que se encuentra en la celda B2
Quebusco = ActiveSheet.Range("B2").Value

'se crea un objeto con el resultado de la función Find
Set resulta = Sheets(mihoja).Range(Donde).Find(Quebusco, LookIn:=xlValues,
LookAt:=xlWhole)

If resulta Is Nothing Then
'si no se encuentra el dato puede mostrar un mensaje de error como el siguiente
MsgBox "No se encontró el dato", vbCritical, "NO ENCONTRADO"
Else

'si encontró el dato puede devolver en la celda G1 la dirección de la celda que contiene el
dato
ActiveSheet.Range("G1").Value = resulta.Address(False, False)
ActiveSheet.Range("G1").Select
End If

'se limpia la variable
Set resulta = Nothing
End Sub
```

**Nota:** el uso de **variables** para definir el **valor a buscar y la hoja y rango** donde hacer la búsqueda, permite repetir la búsqueda con solo asignar otros valores a las variables.

## 19.2 Buscar un dato. Copiar la fila de todos los registros encontrados

La siguiente rutina **busca cierto dato en la columna C** de una hoja (por ejemplo Hoja 3). Si el dato es encontrado completa en la hoja activa (por ejemplo Hoja 2), una fila con el contenido de ese registro y la búsqueda continúa completando otras filas, hasta recorrer toda la columna C de la Hoja 3.

En este ejercicio, utilizaremos las expresiones: **'Find', y 'Find Next'** . También se utiliza el comando especial: **Do... Loop While**

(\*Ver el cap **12 Bucles- Instrucciones o Comandos....**)

```
Sub BusquedaContinua()  
Dim busca As Object  
Dim Primero  
Dim hojaBusc As String, quebusco As String, mihoja As String  
Dim filalibre As Integer  
'en la siguiente variable se indica la hoja dónde buscar  
hojaBusc = "Hoja3"  
  
'el dato a buscar se encuentra en B2 de la hoja activa  
'la variable "mihoja" será donde se volcarán los datos  
mihoja = "Hoja2"  
filalibre = 6  
quebusco = Sheets(mihoja).Range("B2")  
  
'la búsqueda se realiza sobre la columna C de la Hoja3  
Set busca = Sheets(hojaBusc).Range("C5:C65500").Find(quebusco, LookIn:=xlValues,  
Lookat:=xlWhole)  
  
'si busca No es Vacío... es decir si la búsqueda es exitosa y encuentra el dato, guarda la  
dirección en la variable Primero  
  
If Not busca Is Nothing Then  
Primero = busca.Address  
  
'comienza el bucle  
Do  
'completa la fila de la hoja activa (Hoja2) con datos del registro encontrado  
Sheets(mihoja).Cells(filalibre, 1) = busca.Offset(0, -2) 'dato de col A  
Sheets(mihoja).Cells(filalibre, 2) = busca.Offset(0, -1) 'dato de col B  
Sheets(mihoja).Cells(filalibre, 3) = busca 'dato de col C  
filalibre = filalibre + 1  
  
'continúa la búsqueda  
Set busca = Sheets(hojaBusc).Range("C6:C65500").FindNext(busca)  
  
'se repite la rutina hasta volver a la primer dirección guardada. Ahí termina el ciclo  
Loop While Not busca Is Nothing And busca.Address <> Primero  
End If  
  
'se libera la variable  
Set busca = Nothing  
End Sub
```

**Nota:** detalle de la devolución de valores en la instrucción:

***Sheets(mihoja).Cells(filalibre, 1) = busca.Offset(0, -2)***

**busca.Offset(0,-2)** : devuelve el valor de la col **A**, puesto que '**busca**' se encuentra en la col **C**

**Cells(filalibre,1)** : será la columna **A** de la fila libre donde comenzará a volcar el resultado de la búsqueda

En la imagen vemos la tabla de la Hoja3 (hojaBusc) y el resultado en la Hoja2 (mihoja)

	A	B	C
1	<b>Factura</b>	<b>Importe</b>	<b>Cliente</b>
2	A100	\$ 25,00	AA1
3	A105	\$ 28,00	BB2
4	A110	\$ 31,00	AA3
5	A115	\$ 34,00	AA3
6	A120	\$ 37,00	AA2
7	A125	\$ 40,00	BB3
8	A130	\$ 43,00	AA4
9	A135	\$ 46,00	BB6
10	A140	\$ 49,00	AA3
11	A145	\$ 52,00	BB4
12	A150	\$ 55,00	AA5
13	A155	\$ 58,00	BB7
14	A160	\$ 61,00	AA4
15			

	A	B	C
1			
2	<b>Cliente a Buscar</b>	<b>AA3</b>	
3			
4	<b>Factura</b>	<b>Importe</b>	<b>Cliente</b>
5			
6	A110	\$ 31,00	AA3
7	A115	\$ 34,00	AA3
8	A140	\$ 49,00	AA3
9			

### 19.3 Buscar cierto dato en un rango. Si se encuentra borrar la fila que lo contiene

Este código sirve para **eliminar filas** que contienen cierto valor en una columna.

La rutina busca en un rango el valor de la celda activa al momento de ejecutarla. De encontrarse ese valor, se borrará la fila.

(Ver otras maneras de establecer el dato a buscar, en las **Notas al pie**)

```
Sub eliminando() Dim
rango As String Dim
dato As String Dim
midato As Object Dim
fila As Byte
'el contenido de la celda será el dato a buscar
dato = ActiveCell.Value
```

*'rango donde se efectuará la búsqueda*  
*rango = "H1:H15"*

*'se busca en el rango indicado*

*Set midato = ActiveSheet.Range(rango).Find(dato, LookIn:=xlValues, LookAt:=xlWhole)*

*'si midato No es Vacío.... es decir si la búsqueda es exitosa.....*

*If Not midato Is Nothing Then*

*'se borra la fila*

*midato.EntireRow.Delete*

*Else*

*'si el dato no fue encontrado aparecerá un mensaje indicándolo*

*MsgBox "No se encuentra el dato en el rango establecido"*

*End If*

*Set midato = Nothing*

*End Sub*

**Nota:** puede reemplazarse la instrucción: **dato = ActiveCell.Value** por estas otras opciones:

1. Utilizar un Inputbox: **dato = InputBox("Ingrese dato a buscar")**
2. Utilizar otra celda distinta a la activa, incluso de otra hoja:

**dato = Sheets("OtraHoja").Range("B5")**

3. Utilizar el resultado de un cálculo: **dato = variable1 + variable2**

En la primer imagen se muestra la tabla completa y en la segunda, la tabla resultante luego de ejecutar la macro.

	A	B	C	D	E	F
1	Factura	Importe	Cliente	Estado		Dato a buscar
2	A100	\$ 25,00	AA1			Anulada
3	A105	\$ 28,00	BB2			
4	A110	\$ 31,00	AA3	Anulada		
5	A120	\$ 37,00	AA2			
6	A125	\$ 40,00	BB3			
7	A130	\$ 43,00	AA4			
8	A135	\$ 46,00	AA3	Anulada		
9	A145	\$ 52,00	BB4			
10	A150	\$ 55,00	AA5			
11	A155	\$ 58,00	AA3			
12						

	A	B	C	D	E	F
1	Factura	Importe	Cliente	Estado		Dato a buscar
2	A100	\$ 25,00	AA1			Anulada
3	A105	\$ 28,00	BB2			
4	A120	\$ 37,00	AA2			
5	A125	\$ 40,00	BB3			
6	A130	\$ 43,00	AA4			
7	A145	\$ 52,00	BB4			
8	A150	\$ 55,00	AA5			
9	A155	\$ 58,00	AA3			
10						

## 19.4 Comparando cadenas

Si deseamos comparar 2 celdas (o 2 cadenas), utilizaremos la función **InStr**, que nos devolverá los valores: 0 si no coinciden, 1 si coinciden.

La función **InStr** tiene estos argumentos:

**InStr([start, ]string1, string2[, compare])**

(Si el argumento 'compare' es 1 hará una comparación textual)

```
Sub compara1()
Dim valor1, valor2 As String
Dim resulta As Integer
'guardamos el contenido de 2 celdas en variables
valor1 = Range("A3").Value
valor2 = Range("B5").Value
'ejecutamos la comparación a partir de la posición 1
resulta = InStr(1, valor1, valor2, 1)
'devolvemos el resultado de la comparación
MsgBox resulta
End Sub
```

## 19.5 Extraer la parte numérica de una cadena

Para extraer la parte numérica de un texto utilizamos la función **Val**

**Ejemplo 1:** la cadena se encuentra en una variable

```
Sub extraeNum1 ()
Dim texto1 as String
texto1= "100 Pesos"
texto1 = Val(texto1)
MsgBox texto1
End Sub
```

**Ejemplo 2:** si una cadena del tipo: 2008 Facturas se encuentra en la celda activa, en celda siguiente se obtendrá: 2008

```
Sub extraeNum2 () Dim
texto1 as String
texto1= Activecell.value
'dejamos el valor en la columna siguiente
Activecell.offset(0,1) = Val(texto1)
End Sub
```

## 19.6 Eliminar filas si las celdas de cierta columna están vacías

Se compara una columna que va de A1 a A50. Si alguna celda está vacía se elimina la fila completa. De esta manera no quedan filas en blanco.

Aquí se utiliza el concepto de **Colección** con la instrucción **For Each item in Collection**

```
Sub CeldasVacias()
Dim miCelda As Range
'se controla cada celda del rango A1:A50
For Each miCelda In ActiveSheet.Range("A1:A50").Cells
'si la celda está vacía
If IsEmpty(miCelda) Then
'se elimina la fila completa
miCelda.EntireRow.Delete
End If
'se repite el bucle
Next miCelda
End Sub
```

## 19.7 Rellenar celdas vacías de un rango con cierto valor

Esta rutina también es útil si necesitamos completar celdas vacías de un rango con cierto valor, por ejemplo la expresión "Anulado"

```
Sub AnulandoVacias()
Dim miCelda As Range
'controla cada celda del rango de A1 a D50
For Each miCelda In ActiveSheet.Range("A1:D50").Cells
'si la celda está vacía
If IsEmpty(miCelda) Then
'colocará como valor la expresión "Anulado"
miCelda.Value = "Anulado"
End If
Next miCelda
End Sub
```

**Nota:** ver otro ejemplo en el **cap 12: Bucles- comandos especiales**, tema: **Do Until ....Loop**, Ejemplo 2



Roger Perez - Presenta

CURSO DE PROGRAMACION EN VBA

# Capítulo

---



XX

## 20 20- Copiando Datos

### 20.1 Copiar rango de datos de una hoja a la siguiente

En los siguientes ejemplos, copiaremos información de una hoja a otra y también entre distintos libros.

*Recuerde que hay distintas formas de seleccionar el rango a copiar, como ya hemos visto a lo largo de este manual.*

**Nota:** Para más detalles ver el cap **11: Trabajando con celdas**  
– **Selección de celdas o rangos**

Esta rutina se ejecuta **luego** de seleccionar un rango, y copia el rango seleccionado en la **hoja siguiente**, a partir de la celda **D1**

**Nota:** para ejecutarla se podrá utilizar un atajo de teclado, un botón, o buscarla a través del menú Herramientas, Macros

```
Sub CopiaSeleccion()  
    'copiamos el rango seleccionado  
    Selection.Copy  
    'pegamos en la hoja siguiente a partir de la celda D1  
    ActiveSheet.Paste Destination:=ActiveSheet.Next.Cells(1,4)  
    'desactivamos el modo Copiar  
    Application.CutCopyMode= False  
End Sub
```

### 20.2 Copiar cierta fila en otro libro. Conocer última fila con datos

En esta rutina solo se **copia una fila** en **otro libro**. Asumo que el libro ya está abierto (\*Ver instrucciones de cómo abrir otros libros)

Primero se busca la **primer fila libre** de ese libro al que en el ejemplo, llamé **Resumen**.

```
Sub CopiaFila()  
    Dim filalibre As Byte  
    'busca la primer fila libre de la Hoja2 donde se copiarán los datos  
    Workbooks("Resumen").Activate  
    Sheets("Hoja2").Select  
    filalibre = ActiveSheet.Range("A65536").End(xlUp).Row + 1  
  
    'se copia la fila 4 del primer libro y se pega en la primer fila libre del libro Resumen  
    Workbooks(1).Sheets("Hoja1").Activate
```

```
ActiveSheet.Rows("4:4").Copy  
Destination:=Workbooks("Resumen").Sheets("Hoja2").Cells(filalibre, 1)  
'se cierra el libro 2 guardando los cambios  
Workbooks("Resumen").Close Save  
'se desactiva el modo Copiar  
Application.CutCopyMode = False  
End Sub
```

## 20.3 Copiar solo valores- Pegado Especial

Para realizar una copia con **Pegado Especial**, con la opción que presenta el menú **Edición: solo valores**, este es el código:

```
Sub PegarValor()  
'previamente se habrá seleccionado el rango a copiar  
Selection.Copy  
Selection.PasteSpecial Paste:=xlValues  
'inhabilita el modo Copiar/Pegar  
Application.CutCopyMode = False  
End Sub
```

Lo que logramos con estas instrucciones, es dejar una celda que contiene fórmulas, solo con su valor, copiando y pegando sobre sí misma.

## 20.4 Copiar formato - Pegado Especial

Otro ejemplo del modo **Pegado Especial: copiar solo formatos**.

```
Sub PegarFormato()  
'previamente se habrá seleccionado un rango a copiar  
Selection.Copy  
Selection.Offset(-2, 1).PasteSpecial Paste:=xlFormats  
Application.CutCopyMode = False  
End Sub
```

Lo que logramos con estas instrucciones, es copiar el formato de una celda que contiene o no fórmulas, en otra.

## 20.5 Crear libro como copia de hoja

Esta rutina **crea un libro como copia** de una hoja del libro activo.

En aplicaciones como Facturación, quizás necesitemos crear un archivo que solo contenga la hoja Factura.

*El siguiente ejemplo, crea un archivo, en la misma carpeta, como copia de la Hoja4 y le asigna como nombre el contenido de la celda C5 de la misma hoja.*

Para ello copiar en un módulo esta rutina:

```
Sub copiahoja()  
Dim mihoja, minbre, miruta As String  
Dim wb  
miruta = ThisWorkbook.Path  
mihoja = "Hoja4"  
minbre = Trim(Sheets("Hoja4").Range("C5").Value)  
Sheets(mihoja).Copy  
Application.DisplayAlerts = False  
Set wb = ActiveWorkbook  
'por error (ya existe archivo o nbre inválido) muestra ventana para cambiar nbre o ruta  
On Error Resume Next  
With wb  
.SaveAs miruta & "\" & minbre & ".xls" 'ajustar la extensión tratándose de versión  
Excel 2007  
Application.DisplayAlerts = True  
.Close True  
End With  
Set wb = Nothing  
Sheets("Hoja4").Select  
End Sub
```

**Nota:** En el **capítulo 12** se encuentra desarrollado el tema **SET** en detalle.

**FIN DEL MANUAL**