VERSIÓN DEMO

MANUAL de PROGRAMACIÓN EXCEL

---

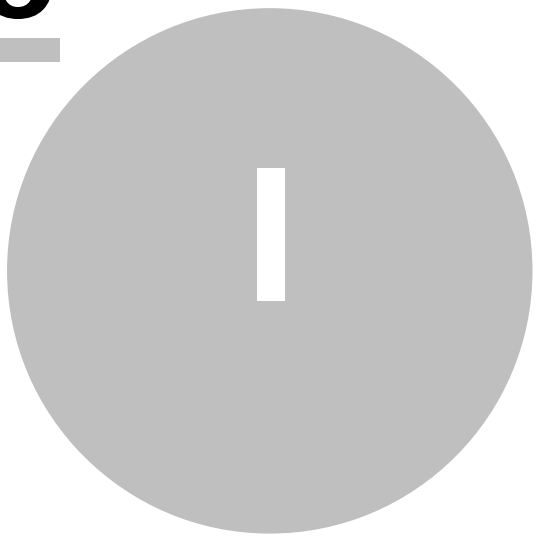
por Elsa Matilde Meyer

(para versiones Excel del 2000 al 2007 inclusive)

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

Tabla de Contenidos

Capítulo



1 1 - Introducción

1.1 BIENVENID@S

Manual: Programando MACROS en Excel

para versiones Excel: 2000 al 2007 inclusive - (v.2.0)

Autor: Elsa M. Meyer (Elsamatilde)

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:

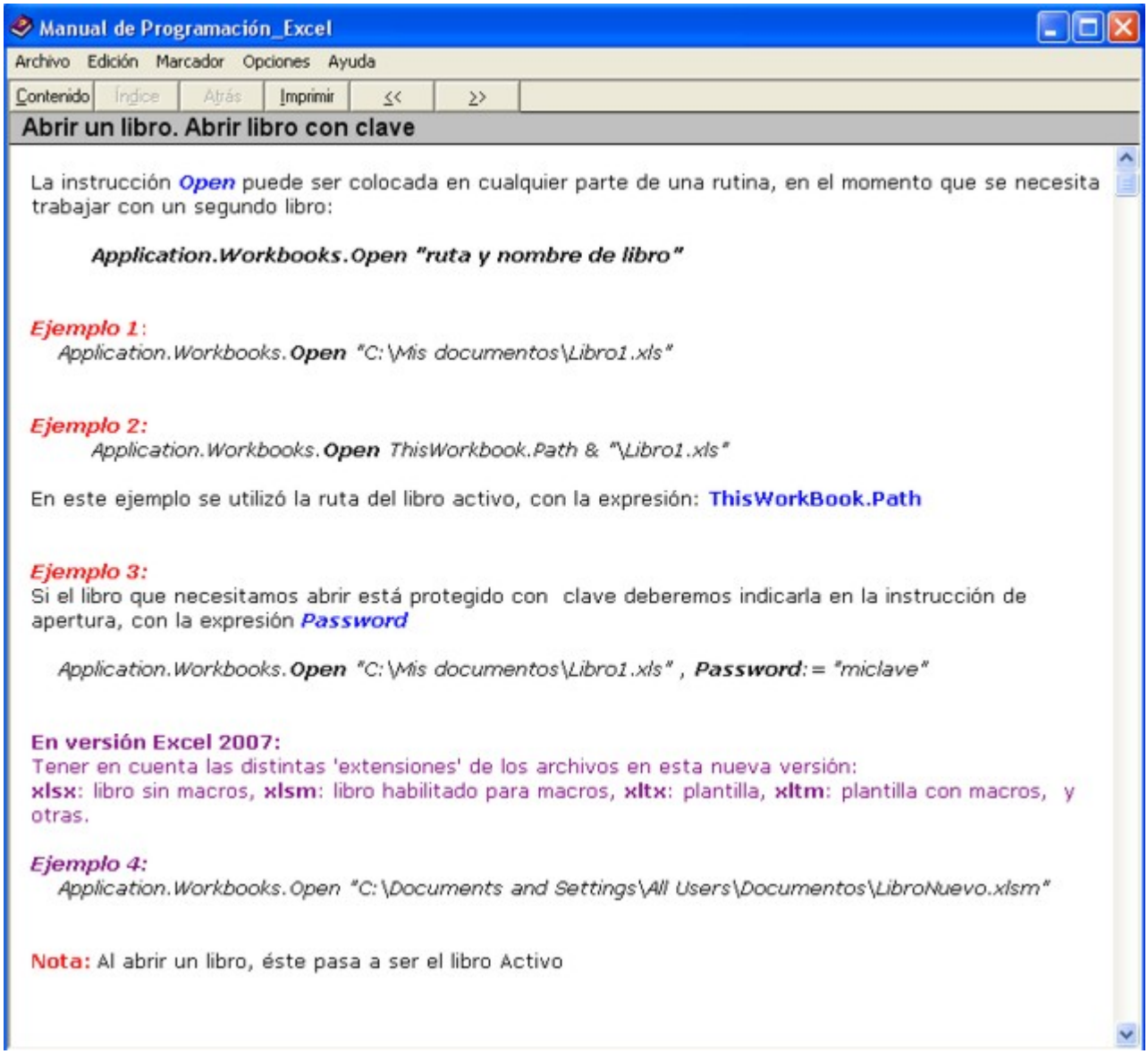
1- los que ***nunca han programado*** (los primeros 7 capítulos en detalle y con imágenes contienen toda la teoría necesaria)

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 empezado a programar en VBA** (nuevos estilos en la programación, más, lo que ha cambiado en Excel 2007)

Contiene: varios ejemplos para un mismo comando, imágenes, aclaración para distintas versiones, vínculos a temas relacionados, notas aclaratorias.



NOTA: Las rutinas contenidas en este manual, fueron desarrolladas y probadas en las siguientes versiones: Windows98/Windows XP con Office 2000, Office XP (2002) , Office 2003, Office 2007.

En cada tema se hace mención a las diferencias entre versiones si las hubiere.

Se incluyen imágenes, tanto de la versión 2007 como anteriores, en los temas que así lo requieren para una mejor comprensión.

IMPORTANTE: por tratarse de una versión **DEMO**, solo se encuentran desarrollados aquellos temas que aparecen en **mayúsculas** en el **índice**.

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.

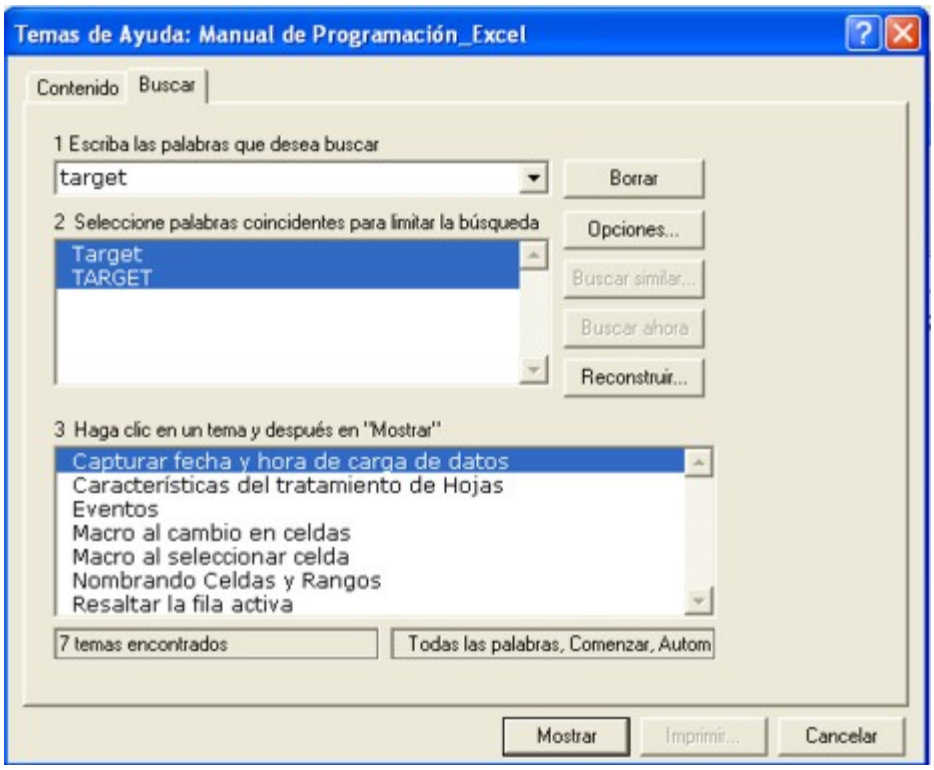
Así, partiendo de las principales instrucciones básicas, y luego de **más de 100 rutinas**, se llegará a un gran dominio en programación de Macros para Excel.

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.

IMPORTANTE:
Las principales características de este Manual y su formato, además de su contenido, son:

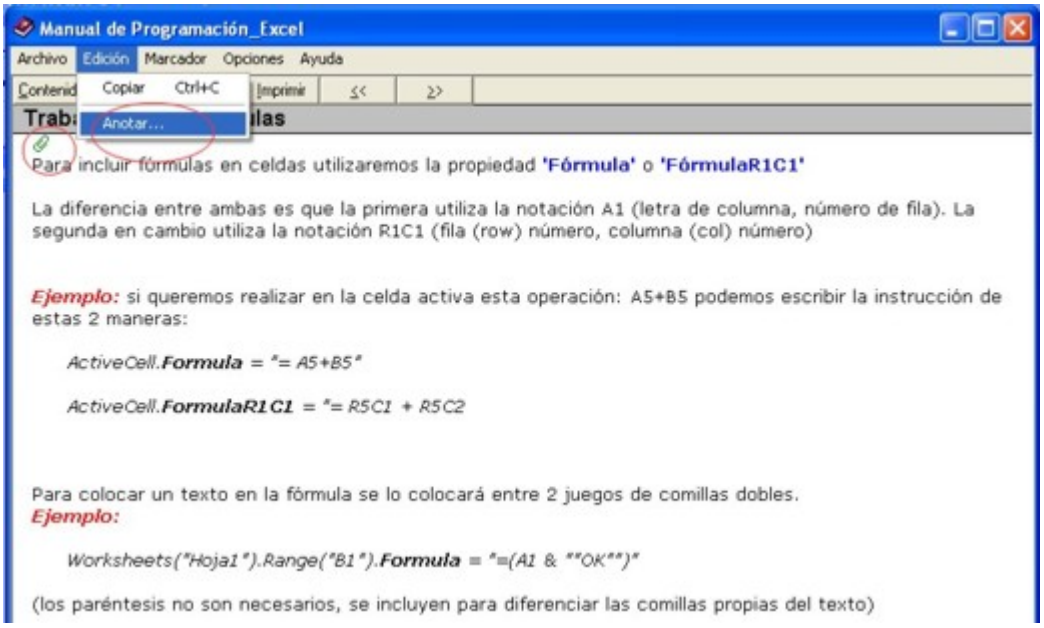
1- *su potente buscador*. Al ***'buscar'*** un tema, el programa nos devolverá la lista de entradas que aparecen en todo el manual, como en la imagen donde se ingresó la palabra 'find' apareciendo 11 entradas (para la primer coincidencia) lo que nos permite abarcar de manera completa el tema buscado.



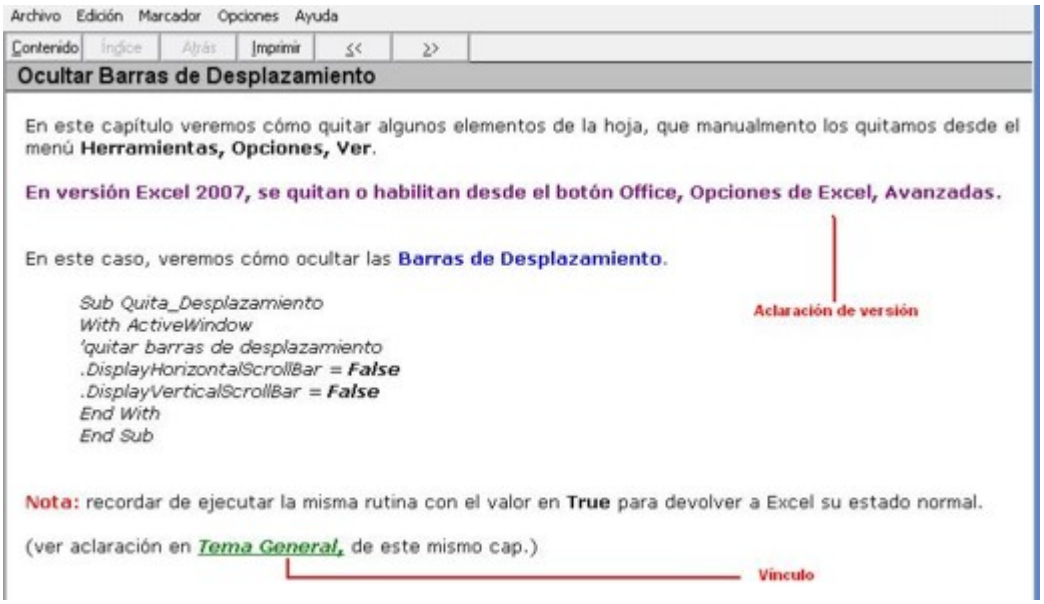
2- posibilidad de *agregar comentarios* en cada tema. El usuario podrá agregar sus propias observaciones, ejemplos u otros datos de interés.

Desde el menú **Edición, Anotar** se mostrará una ventana para escribir el comentario.

Cada vez que se acceda a este tema, aparecerá en el margen superior un clip recordando que el tema tiene agregados. Haciendo **clic** en él, aparecerán las notas personales.



3- posibilidad de seguir los *temas relacionados*, siguiendo los ***vínculos*** que pueden encontrarse en algunos casos, como en la imagen:



Con el botón **ATRÁS** *(en la imagen desactivado)* , se volverá al tema que **anterior** desde donde se llamó al **vínculo**.

Capítulo

2 2- Cómo programar en Excel?

2.1 QUÉ ES UNA MACRO?

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 ocultadas.

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)

(ver img01 que se acompaña con la demo)

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

'Mas frecuentes' y tildar la opción:
'Mostrar ficha Programador en cinta de opciones'.

(ver imágenes img02 - img03 que se acompañan con la Demo)

2.2 Breve introducción al Editor

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

2.3 Cómo crear una macro

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

2.4 Uso de la grabadora de macros

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

2.5 Privada o Pública

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo

3 3 - Referencias a objetos Excel

3.1 Nombrando Libros y hojas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

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 evaluan 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.

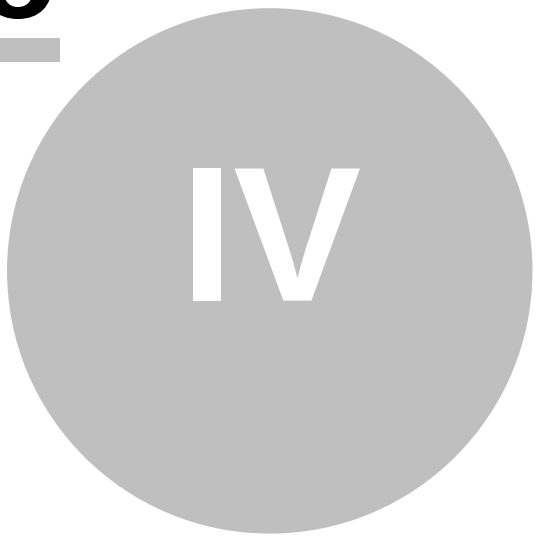
Si omitimos los argumentos **'false'** tanto para fila como columna, nos evaluará una referencia 'absoluta':

```
If Target.Address = "$A$5"
```

3.3 Nombrando Objetos

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo



4 4 - Eventos, Metodos y Propiedades

4.1 EVENTOS

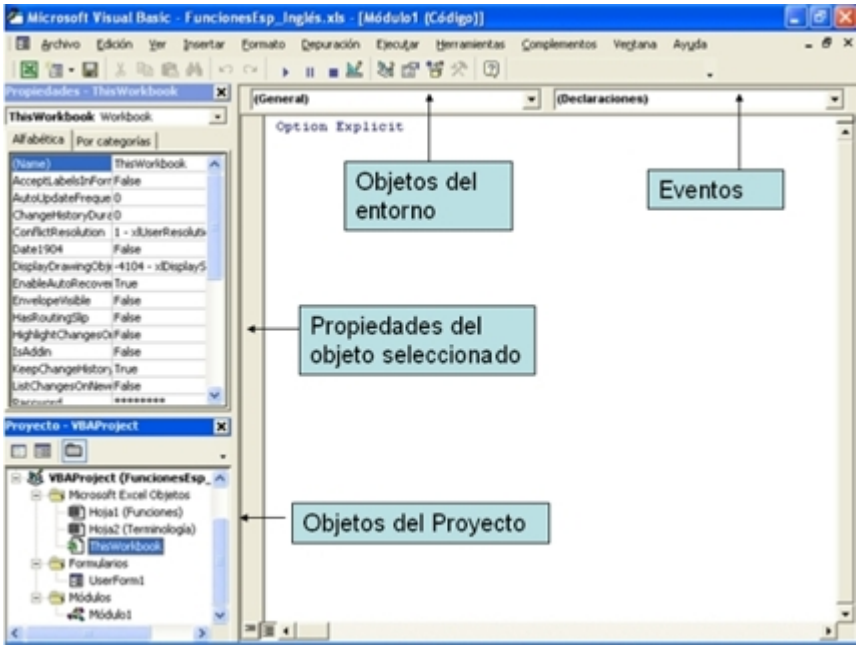
Hemos visto en el capítulo 2, tema 'Qué es una macro', que se necesitaba en principio de un espacio (el Editor) y una acción o **evento** que ejecutara una macro.

¿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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

4.3 Eventos de Hojas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

4.4 Macro al cambio en celdas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

4.5 Macro al seleccionar celda

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

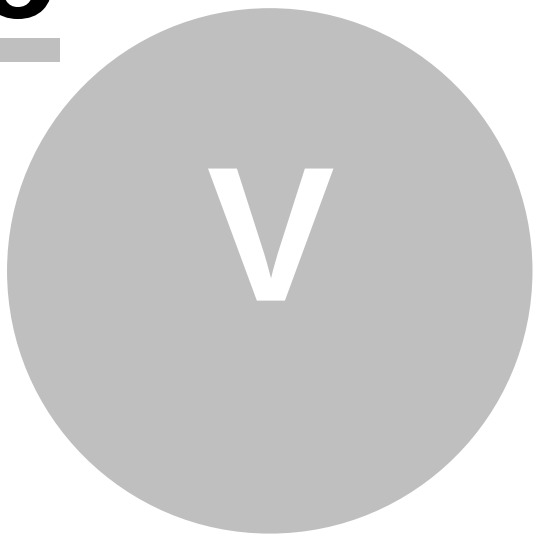
4.6 Metodos

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

4.7 Propiedades

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo



5 5 - Ejecutando macros

5.1 Dónde colocar las macros

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

5.2 Cómo ejecutar una macro

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

5.3 EJECUTAR MACRO DESDE UN BOTÓN

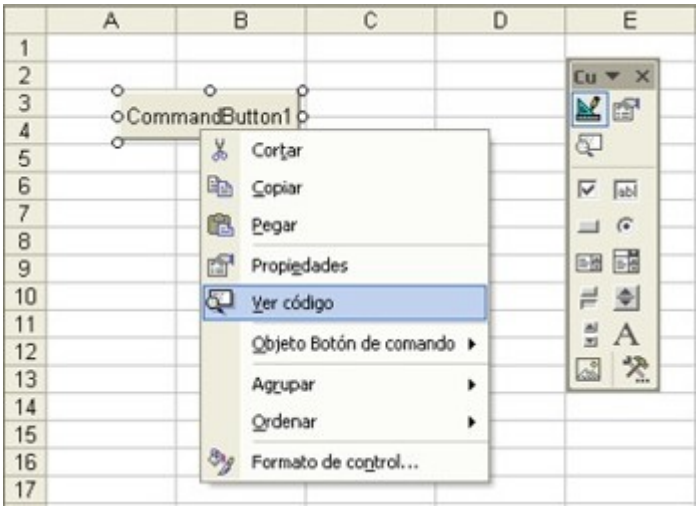
1. Utilizando botones
Desde el menú Ver de Excel, Barras de Herramientas, Cuadro de controles o Formularios, es posible insertar botones de comando. Una vez dibujados en la hoja, se le asignarán o escribirán las instrucciones a ejecutar al clic de esos botones.

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:

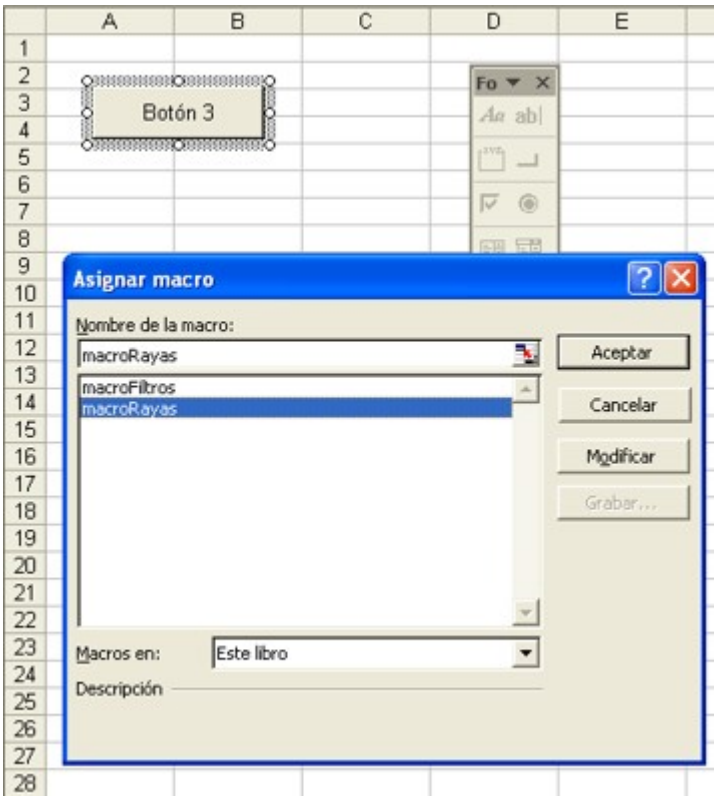
```
Private Sub CommandButton1_Click()
End Sub
```

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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

5.5 Acerca de las macros Auto-Open

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

5.6 Acerca de las macros Auto-Close

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

5.7 Cómo probar una macro

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

5.8 Cómo controlar la ejecución de una macro

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

5.9 Conocer el valor que toman las variables

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

5.10 Cómo evitar que una instrucción se ejecute

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

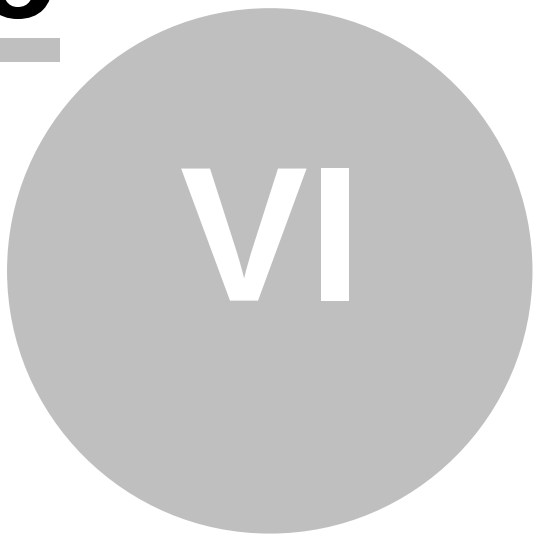
5.11 Acceder a la Ayuda desde una línea de código

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

5.12 Salir de una rutina

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo



6 6 - Seguridad en el proyecto

6.1 CÓMO PROTEGER UN PROYECTO?

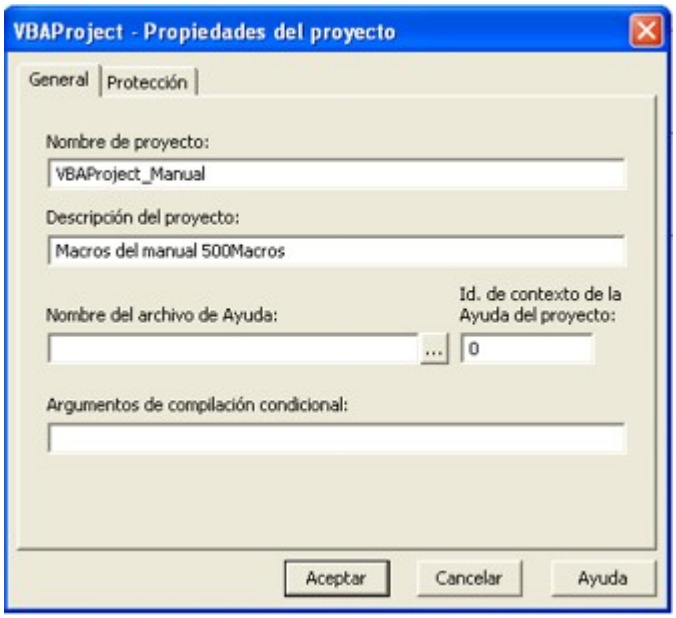
Proteger el proyecto VBA:

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

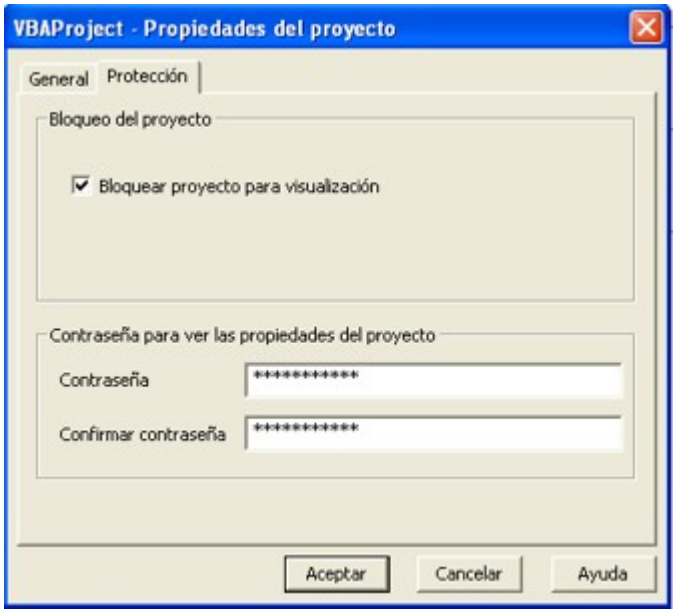
Para esto, dentro del **Editor**, pestaña **Herramientas**, optar por ***'Propiedades de VBA Proyect'***

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 VBAProyect (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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

6.3 Cómo interrumpir una macro

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

6.4 Habilitar o no las macros

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo

7 7- Tratamiento de Variables

7.1 Tipo de Variables

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

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***. Recomiendo 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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

7.4 Convirtiendo variables

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

7.5 Limpiando variables

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo

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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

8.3 Obtener el largo de una cadena

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

8.4 Introducir caracteres especiales

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

8.5 Detectar o encontrar texto en una cadena

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

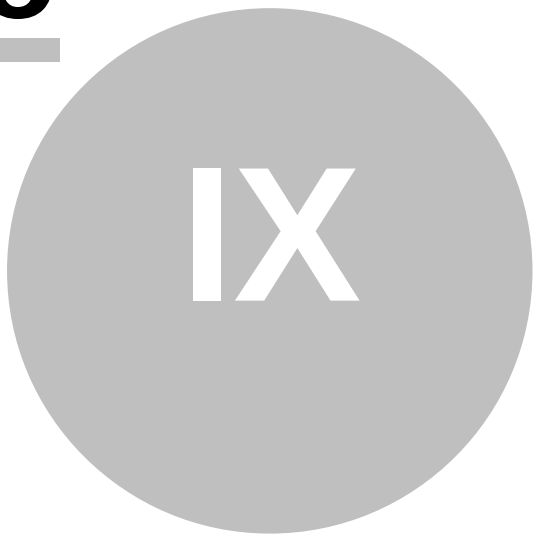
8.6 Creando cadenas de largo fijo

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

8.7 Obtener la parte numerica de una cadena

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo



9 9 - Trabajando con Libros

9.1 Principales Metodos y Propiedades de Libros

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

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 disponible 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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.4 Al abrir libro incrementar un contador

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.5 Seleccionar un libro

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.6 Activar otro libro distinto al actual

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.7 Obtener la ruta de un libro

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.8 Guardando Libros

9.8.1 Guardar el libro activo

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.8.2 Guardar un libro con otro nombre o formato

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.8.3 Guardar un libro con clave

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.8.5 Guardar un libro cuyo nombre serán datos concatenados

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.9 Cerrando Libros

9.9.1 Cerrar todos los libros en uso

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.9.2 Cerrar un solo libro

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

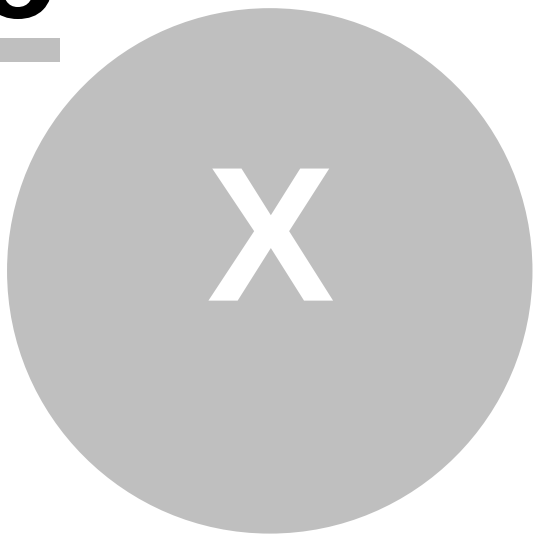
9.9.3 Cerrar un libro SIN guardar los cambios

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

9.9.4 Cerrar un libro guardando los cambios

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo



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 se desarrollan ejemplos de los **métodos y propiedades** más comunes de las **hojas**.

Ejemplo 2:
Si una rutina debe ser ejecutada en **todas** l**as 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)

End Sub
```

Nota: Ver más detalles en cap 4: ***Eventos***, *Evaluar cambios en hojas*.

10.2 Principales Metodos y Propiedades de Hojas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.3 Activar o seleccionar otras hojas distintas a la actual

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.4 Seleccionar la hoja anterior o posterior a la activa

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.5 Devolver el nombre de la hoja en una variable

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.6 Seleccionar todas las hojas de un libro

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.7 Proteger una hoja

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.8 Desproteger una hoja

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.9 Vista previa de la hoja activa y de otras hojas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.10 Imprimir hojas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.11 Insertar hojas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.12 Eliminar hojas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.13 Copiar hojas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.14 Ocultar hojas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

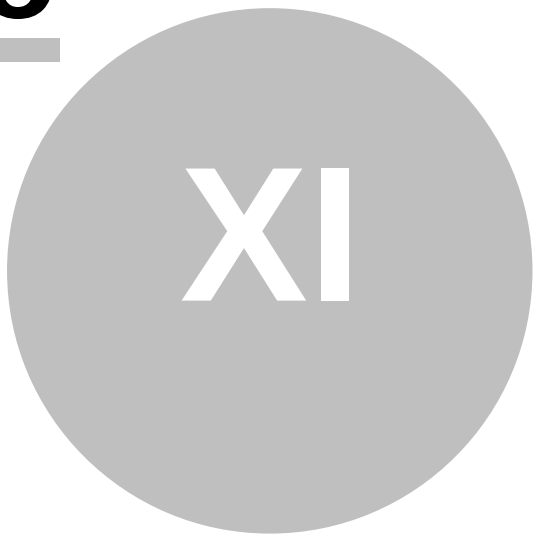
10.15 Mostrar hoja oculta

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

10.16 Establecer area visible de una hoja

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo



11 11 - Trabajando con Celdas y Rangos

11.1 General

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.2 Principales Metodos y Propiedades de Rangos

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.3 Selección de Celdas o Rangos

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.4 Selección de rango utilizando variables

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.5 Seleccionar celdas a cierta distancia de la celda activa

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.6 Seleccionar la región donde se encuentra la celda activa

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.7 Seleccionar hasta la última celda vacía -Fin de rango

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.8 Obtener primer fila libre

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.11 Borrar o Limpiar celdas o rangos

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.12 Eliminar celdas o rangos

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.13 Eliminar varias filas segun condicion

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.14 Insertar Filas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.15 Eliminar Filas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.16 Ocultar filas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.17 Mostrar filas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.18 Insertar Columnas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.19 Eliminar columnas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.20 Ocultar columnas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.21 Mostrar Columnas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.22 Capturar fecha y hora de carga de datos

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.23 Ordenar un rango

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.24 Detectar si la celda contiene formula

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

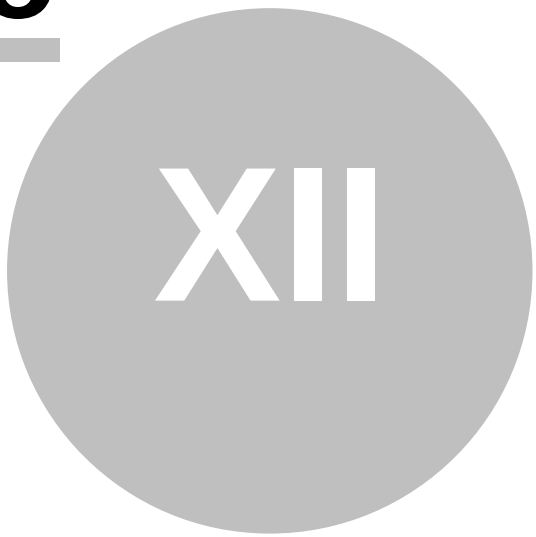
11.25 Resaltar la fila activa

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

11.26 Cambiar color de fuente a celdas con datos

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo



12 12 - Bucles= Comandos Especiales

12.1 General

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

12.2 FOR EACH....NEXT

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

Sintaxis:

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

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, se encontrarán otros ejemplos del uso de este comando.

12.3 For ..... Next

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

12.4 While .... Wend

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

12.5 If.... ElseIf....Else....

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

12.6 Do While .... Loop

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

12.7 Do Until.... Loop

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

12.8 Uso de SET

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

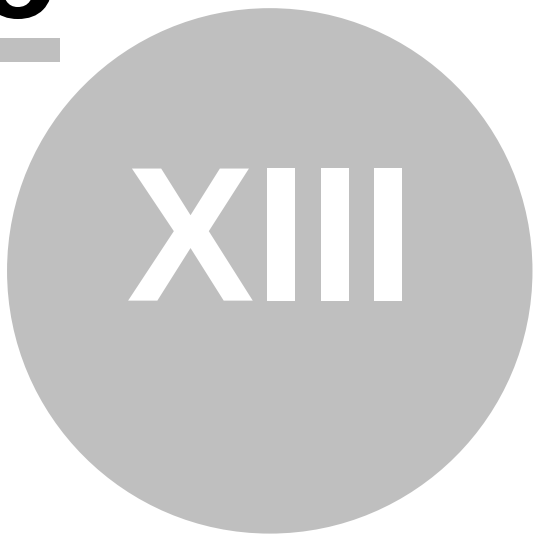
12.9 With....End With

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

12.10 Uso de Select Case

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo



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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo

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 propiedadas 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:

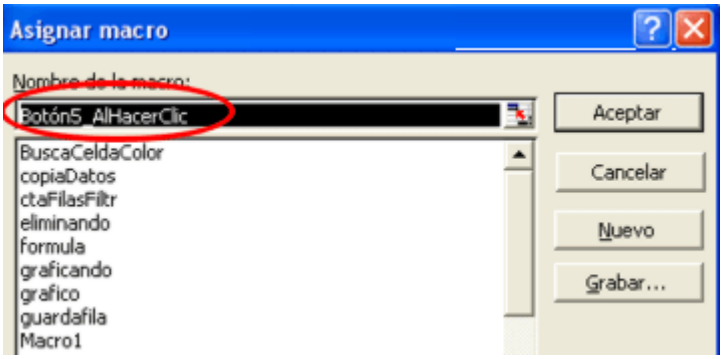
```
ActiveSheet.ComboBox1.Visible = True   'Cuadro de controles
```

```
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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

14.3 Enviar valor de un Combo a una celda

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

14.4 Enviar valores de Combo de 4 columnas a celdas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

14.5 Buscar valor del Combo en base Devolver otros datos en textbox

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

14.6 Limpiar un combo

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

14.7 Funciones de validación

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

14.8 Asignar formato moneda a un TextBox

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

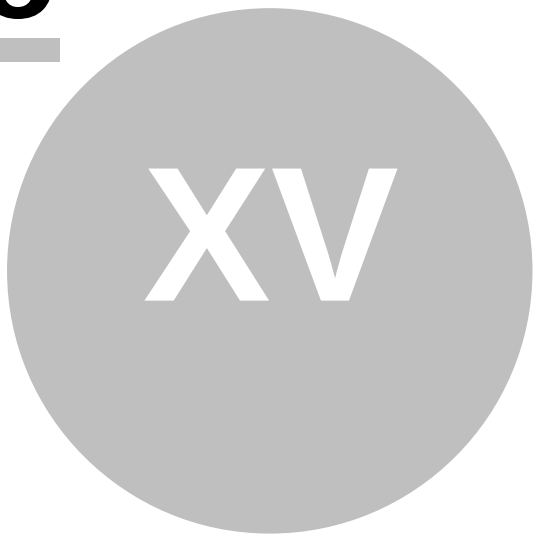
14.9 Validar campos fecha en Textbox

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

14.10 Validar campos numéricos en Textbox

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo



15 15 - Controlando Mensajes de Excel

15.1 General

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

15.2 No mostrar avisos de alerta

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

15.3 NO MOSTRAR AVISO AL GUARDAR UN ARCHIVO....

Este ejemplo se utiliza en el objeto ***ThisWorkbook*** e impide que se muestre un aviso al guardar el archivo, de que el ***nombre de archivo ya existe***.

Antes de cerrar el libro, se lo guardará como Libro1. Si ya existe un libro con ese nombre en la carpeta asignada, Excel lo sobreescribirá ***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:

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo

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 siguente 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 ....

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

16.3 On Error GoTo 0

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo

17 17 - Uso de MsgBox e InputBox

17.1 Construcción de MsgBox

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

17.2 Construcción de InputBox

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

17.3 CONTROLAR QUE SE INGRESE DATO EN INPUTBOX

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
```

Capítulo

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 manualmento 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 *Aclaración General*, de este mismo cap.)

18.2 Ocultar Pestañas de hoja

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

18.3 Ocultar Encabezados de filas y col

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

18.4 Ocultar Líneas de División

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

18.5 Quitar barras frecuentes

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

18.6 Aclaración: Rutina General

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo

19 19- Buscando-Comparando-Evaluando Datos

19.1 DEVOLVER EN UNA CELDA EL RESULTADO DE UNA BUSQUEDA

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.**

Nota: 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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

19.3 Buscar cierto dato en un rango. Si se encuentra borrar la fila que lo contiene

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

19.4 Comparando cadenas

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

19.5 Extraer la parte numérica de una cadena

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

19.6 Eliminar filas si las celdas de cierta columna están vacías

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

19.7 Rellenar celdas vacías de un rango con cierto valor

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

Capítulo

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

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

20.3 Copiar solo valores- Pegado Especial

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

20.4 Copiar formato - Pegado Especial

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

20.5 Crear libro como copia de hoja

Por tratarse de una versión DEMO, este tema no se encuentra disponible.

FIN DEL MANUAL