MACROS EN EXCEL CON VBA

OSCAR MARIO GIL RÍOS

INGENIERO DE SISTEMAS Y ESPECIALISTA EN REDES

CREAR UNA MACRO

Si hay tareas de Microsoft Excel que realiza reiteradamente, puede grabar una macro o bien programarla usando el editor de visual para aplicaciones y generar la automatización de las mismas, usando la técnica de la programación usted podrá realizar aplicaciones para el área de trabajo del programa y lograr una integración con el entorno del sistema operativo de microsoft . Una macro es una acción o un conjunto de acciones que se puede ejecutar todas las veces que desee. Cuando se crea una macro, se graban los clics del mouse y las pulsaciones de las teclas (eventos de usuario). Después de crear una macro, puede modificarla para realizar cambios menores en su funcionamiento.

La productividad individual y corporativa crecerá de manera significativa al tener la capacidad de programar sus tareas y con simples eventos de usuario estás puedan ser generalizadas, nos introduciremos al mundo de la programación utilizando el lenguaje basic como lenguaje de programación aplicado en el entorno de Microsoft Excel.

CONTENIDO

- Grabar un Macro
- Ejecutar una Macro
- Configuración del centro de confianza de Office y activación de la ficha programador
- Utilidad de una Macro
- Grabar una macro con referencias relativas y absolutas
- El Editor de Visual Basic para aplicaciones.
- Ejemplo de Macro programada
- Explorador de proyectos
- Ventana de propiedades.
- Ventana de código
- Formularios
- Crear un formulario.
- Controles de formulario.
- Formulario de Excel
- Controles de formulario
- Objetos como herramientas de dibujo
- Determinar el tipo de control en una hoja de cálculo
- Casillas de verificación, botones de opción y botones de alternancia.

CONTENIDO

- Aplicación de la casilla de verificación
- Aplicación del botón de opción. (Control de formulario)
- Grabar un Macro
- Ejecutar una Macro
- Configuración del centro de confianza de Office
- Resumen de las propiedades por categorías funcionales.
- 14 formas de acelerar y optimizar tus macros en Microsoft Excel.
- Msg Box
- Trabajo con variables
- Estructuras repetitivas
- Instrucción Select
- Códigos de Vba for App.

Grabar una macro

¿Cómo?

1 Antes de grabar una macro

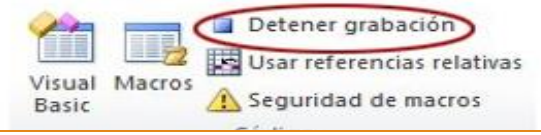
Compruebe que se muestra la ficha **Programador** en la cinta de opciones. Dado que la ficha **Programador** no se muestra de manera predeterminada, haga lo siguiente:

1. Haga clic en la pestaña **Archivo**, elija **Opciones** y, a continuación, haga clic en la categoría **Personalizar cinta de opciones**.
2. En **Personalizar cinta de opciones**, en la lista **Fichas principales**, haga clic en **Programador** y, a continuación, haga clic en **Aceptar**.

2 Grabe una macro.

1. En el grupo **Código** en la pestaña **Programador**, haga clic en **Grabar macro** y luego haga clic en **Aceptar** para comenzar a grabar.

2. Realice algunas acciones en la hoja de cálculo como escribir algún texto, seleccionar algunas columnas o filas o rellenar con algunos datos.
3. En el grupo **Código** en la pestaña **Programador**, haga clic en **Detener grabación**.



3 Examine la macro y pruébela.

Al modificar la macro que ha grabado, puede aprender un poco acerca del lenguaje de programación Visual Basic.

Para editar una macro, en el grupo **Código** en la pestaña **Programador**, haga clic en **Macros**, seleccione el nombre de la macro que ha grabado y haga clic en **Editar**. Esta acción hará que se inicie el Editor de Visual Basic.

Observe el código y vea de qué manera las acciones que ha grabado aparecen como código. Es probable que entienda bien algo del código y que otra parte le resulte un poco misteriosa.

Experimente con el código, cierre el Editor de Visual Basic y ejecute la macro nuevamente. Esta vez observe si sucede algo distinto.

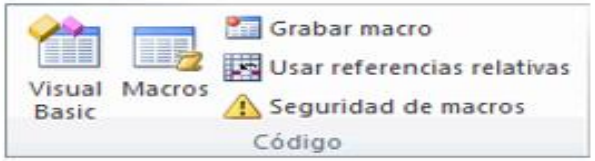
Ejecutar una Macro

Existen varias maneras de ejecutar una macro en Microsoft Excel. Una macro es una acción o un conjunto de acciones que permiten automatizar tareas. Las macros se graban en el lenguaje de programación de Visual Basic para Aplicaciones. Para ejecutar una macro, se hace clic en el comando **Macros** de la cinta de opciones (ficha **Programador**, grupo **Código**). En función de cómo se haya asignado la ejecución de una macro, para ejecutar la macro también se podrá usar un método abreviado de combinación con la tecla CTRL, o se podrá hacer clic en la barra de herramientas de acceso rápido o en un grupo personalizado de la cinta de opciones, o en una área de un objeto, gráfico o control. Además, una macro se puede ejecutar automáticamente al abrirse un libro.

Nota Cuando se establece el nivel de seguridad de la macro en Excel en **Deshabilitar todas las macros sin notificación**, Excel ejecuta únicamente aquellas macros que están firmadas digitalmente o almacenadas en una ubicación de confianza, como la carpeta de inicio de Excel del equipo. Si la macro que desea ejecutar no está firmada digitalmente o no se encuentra en una ubicación de confianza, puede cambiar temporalmente el nivel de seguridad para habilitar todas las macros.

Ejecutar una macro

1. Si la pestaña **Programador** no está disponible, haga lo siguiente para mostrarla:
   1. Haga clic en la pestaña **Archivo**, elija **Opciones** y, a continuación, haga clic en la categoría **Personalizar cinta de opciones**.
   2. En la lista **Fichas principales**, active la casilla de verificación **Programador** y haga clic en **Aceptar**.
2. Para establecer el nivel de seguridad de manera que estén habilitadas temporalmente todas las macros, haga lo siguiente:
   1. En la pestaña **Programador**, en el grupo **Código**, haga clic en **Seguridad de macros**.



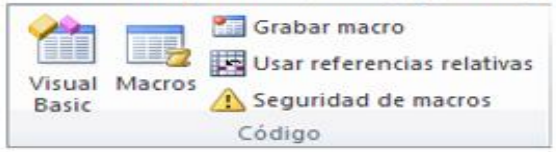
   2. En la categoría **Configuración de macros**, en **Configuración de macros**, haga clic en **Habilitar todas las macros (no recomendado; puede ejecutarse código posiblemente peligroso)** y, a continuación, haga clic en **Aceptar**.

Nota Para ayudar a evitar que se ejecute código potencialmente peligroso, recomendamos que vuelva a cualquiera de las configuraciones que deshabilitan todas las macros cuando termine de trabajar con las macros.

1. Abra el libro que contiene la macro.
2. En la ficha **Programador**, en el grupo **Código**, haga clic en **Macros**.

Ejecutar una Macro

2. En la ficha **Programador**, en el grupo **Código**, haga clic en **Macros**.



3. En el cuadro **Nombre de la macro**, haga clic en la macro que desea ejecutar.
4. Siga uno de los procedimientos siguientes:

- Para ejecutar una macro en un libro de Excel, haga clic en **Ejecutar**.

Sugerencia También puede presionar CTRL+F8 para ejecutar la macro. Puede interrumpir la ejecución de la macro presionando ESC.

- Para ejecutar una macro desde un módulo de Microsoft Visual Basic para Aplicaciones (VBA), haga clic en **Editar** y, en el menú **Ejecutar**, haga clic en **Ejecutar Sub/UserForm** , o presiones F5.

Ejecutar una macro presionando una combinación de teclas de método abreviado con CTRL

1. Si la pestaña **Programador** no está disponible, haga lo siguiente para mostrarla:
   1. Haga clic en la pestaña **Archivo**, elija **Opciones** y, a continuación, haga clic en la categoría **Personalizar cinta de opciones**.
   2. En la lista **Fichas principales**, active la casilla de verificación **Programador** y haga clic en **Aceptar**.
2. En la ficha **Programador**, en el grupo **Código**, haga clic en **Macros**.

3. En el cuadro **Nombre de la macro**, haga clic en la macro a la que desea asignar una combinación de teclas con CTRL.
4. Haga clic en **Opciones**.

Aparecerá el cuadro de diálogo **Opciones de la macro**.

5. En el cuadro **Tecla de método abreviado**, escriba cualquier letra minúscula o mayúscula que desee usar.

Nota La tecla de método abreviado invalidará a cualquier tecla de método abreviado predeterminada equivalente en Excel mientras esté abierto el libro que contiene la macro.

6. Escriba una descripción de la macro en el cuadro **Descripción**.
7. Haga clic en **Aceptar** para guardar los cambios y, a continuación, en **Cancelar** para cerrar el cuadro de diálogo **Macro**.

Ejecutar una Macro

Ejecutar una macro mediante un botón de un grupo personalizado de la cinta de opciones

Si aprovecha la capacidad de personalización de la cinta de opciones en Excel 2010, puede crear un grupo personalizado que aparezca en una ficha de la cinta de opciones y, a continuación, asignar una macro a un botón de ese grupo. Por ejemplo, puede agregar un grupo personalizado denominado "Mis macros" a la ficha Programador, y agregar una macro (que aparece como un botón) al nuevo grupo.

Ejecutar una macro haciendo clic en un área de un objeto gráfico

Puede crear una zona activa en un gráfico donde los usuarios pueden hacer clic para ejecutar una macro.

1. En la hoja de cálculo, inserte un objeto gráfico, como una imagen, una imagen prediseñada, una forma o un gráfico SmartArt.

Para obtener información sobre cómo insertar un objeto gráfico, vea el tema sobre cómo agregar, cambiar o eliminar formas.

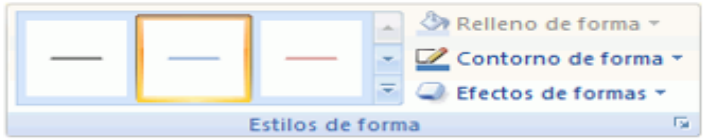
2. Para crear una zona activa en el objeto existente, en la pestaña **Insertar**, en el grupo **Ilustraciones**, haga clic en **Formas**, seleccione la forma que desea usar y, a continuación, dibuje dicha forma en el objeto existente.

3. Haga clic con el botón secundario en la zona activa que ha creado y, a continuación, elija **Asignar Macro**.
4. Siga uno de los procedimientos siguientes:
   - Para asignar una macro al botón u objeto gráfico, haga doble clic en la macro o escriba el nombre de la misma en el cuadro **Nombre de la macro**.
   - Para grabar una nueva macro y asignarla al objeto gráfico seleccionado, haga clic en **Grabar**, escriba un nombre para la macro en el cuadro de diálogo **Grabar macro** y haga clic en **Aceptar** para comenzar a grabar la macro. Cuando termine de grabarla, haga clic en **Detener grabación** en la ficha **Programador** del grupo **Código**.

Sugerencia También puede hacer clic en **Detener grabación** en el lado izquierdo de la barra de estado.

   - Para modificar una macro existente, haga clic en el nombre de la macro, en el cuadro **Nombre de la macro** y, a continuación, haga clic en **Modificar**.

5. Haga clic en **Aceptar**.
6. En la hoja de cálculo, seleccione la zona activa. Se mostrarán las herramientas de **Dibujo** y se agregará una ficha **Formato**.
7. En el grupo **Formato**, en el grupo **Estilos de forma**, haga clic en la flecha junto a **Relleno de forma** y elija **Sin relleno**.



8. Haga clic en la flecha situada junto a **Contorno de forma** y, a continuación, haga clic en **Sin contorno**.

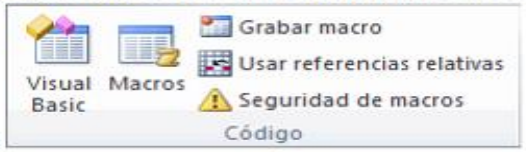
Ejecutar una Macro

Configurar una macro para que se ejecute automáticamente al abrirse un libro

Si graba una macro y la guarda con el nombre "Auto_abrir", la macro se ejecutará cada vez que se abra el libro que contiene la macro. Otra forma de ejecutar automáticamente una macro al abrir un libro es escribir un procedimiento de VBA en el evento **Open** del libro usando el Editor de Visual Basic. El evento **Open** es un evento de libro integrado que ejecuta su código de macro cada vez que se abre el libro.

CREAR UNA MACRO AUTO_ABRIR

1. Si la pestaña **Programador** no está disponible, haga lo siguiente para mostrarla:
   1. Haga clic en la pestaña **Archivo** y, a continuación, elija **Opciones**.
   2. En la categoría **Personalizar cinta**, en la lista **Pestañas principales**, active la casilla **Programador** y, a continuación, haga clic en **Aceptar**.
2. Para establecer el nivel de seguridad de manera que estén habilitadas temporalmente todas las macros, haga lo siguiente:
   1. En la ficha **Programador**, en el grupo **Código**, haga clic en **Seguridad de macros**.



   2. En la categoría **Configuración de macros**, bajo **Configuración de macros**, haga clic en **Habilitar todas las macros (no recomendado; puede ejecutarse código posiblemente peligroso)** y, a continuación, haga clic en **Aceptar**.

Nota Para ayudar a evitar que se ejecute código potencialmente peligroso, recomendamos que vuelva a cualquiera de las configuraciones que deshabilitan todas las macros cuando termine de trabajar con las macros.

1. Si desea guardar la macro con un libro determinado, abra primero ese libro.
2. En la pestaña **Programador**, en el grupo **Código**, haga clic en **Grabar macro**.
3. En el cuadro **Nombre de la macro**, escriba **Auto_abrir**.
4. En la lista **Guardar macro en**, seleccione el libro donde desea almacenar la macro.

Sugerencia Si desea que la macro esté disponible siempre que use Excel, seleccione **Libro de macros personal**. Cuando se selecciona **Libro de macros personal**, Excel crea un libro oculto de macros personal (Personal.xlsb), si aún no existe, y guarda la macro en este libro. En Windows Vista, este libro se guarda en la carpeta C:\Usuarios*nombre de usuario*\AppData\Local\Microsoft\Excel\XLStart. Si no lo encuentra aquí, es posible que se haya guardado en la subcarpeta Roaming, en lugar de en Local. En Microsoft Windows XP, este libro se guarda en la carpeta C:\Documents and Settings*nombre de usuario*\Datos de programa\Microsoft\Excel\XLStart. Los libros almacenados en la carpeta XLStart se abren automáticamente al iniciar Excel. Si desea que se ejecute automáticamente una macro del libro de macros personal en otro libro, también debe guardar ese libro en la carpeta XLStart, de forma que ambos libros se abran cuando se inicie Excel.

5. Haga clic en **Aceptar** y realice las acciones que desea grabar.
6. En la pestaña **Programador**, en el grupo **Código**, haga clic en **Detener grabación**  .

Sugerencia También puede hacer clic en **Detener grabación** en el lado izquierdo de la barra de estado.

Ejecutar una Macro

Notas

- Si en el paso 6 eligió guardar la macro en **Este libro** o en **Libro nuevo**, guarde o mueva el libro a una de las carpetas XLStart.
- La grabación de una macro Auto_abrir tiene las limitaciones siguientes:
  - Si el libro en donde se guarda la macro Auto_abrir ya contiene un procedimiento de VBA en su evento **Open**, el procedimiento de VBA del evento **Open** invalidará todas las acciones contenidas en la macro Auto_abrir.
  - Las macros Auto_abrir se omiten cuando se abren libros mediante programación utilizando el método **Open**.
  - Una macro Auto_abrir se ejecuta antes de que se abra cualquier otro libro. Por lo tanto, si graba acciones que desea que realice Excel en el libro predeterminado Libro1 o en un libro cargado desde la carpeta XLStart, la macro Auto_abrir producirá un error cuando reinicie Excel porque la macro se ejecuta antes de abrir los libros de inicio y el predeterminado.

Si encuentra estas limitaciones, en vez de grabar una macro Auto_abrir, debe crear un procedimiento de VBA para el evento **Open** como se describe en la sección siguiente de este artículo.

- Si desea iniciar Excel sin ejecutar una macro Auto_abrir, mantenga presionada la tecla MAYÚS al abrir el programa.

CREAR UN PROCEDIMIENTO DE VBA PARA EL EVENTO OPEN DE UN LIBRO

El ejemplo siguiente utiliza el evento **Open** para ejecutar una macro al abrir el libro.

1. Si la pestaña **Programador** no está disponible, haga lo siguiente para mostrarla:
   1. Haga clic en la pestaña **Archivo** y, a continuación, elija **Opciones**.
   2. En la categoría **Personalizar cinta**, en la lista **Pestañas principales**, active la casilla **Programador** y, a continuación, haga clic en **Aceptar**.
2. Para establecer el nivel de seguridad de manera que estén habilitadas temporalmente todas las macros, haga lo siguiente:
   1. En la ficha **Programador**, en el grupo **Código**, haga clic en **Seguridad de macros**.

   2. En la categoría **Configuración de macros**, bajo **Configuración de macros**, haga clic en **Habilitar todas las macros (no recomendado; puede ejecutarse código posiblemente peligroso)** y, a continuación, haga clic en **Aceptar**.

Nota Para ayudar a evitar que se ejecute código potencialmente peligroso, recomendamos que vuelva a cualquiera de las configuraciones que deshabilitan todas las macros cuando termine de trabajar con las macros.

1. Guarde y cierre todos los libros abiertos.
2. Abra el libro donde desea agregar la macro o cree un nuevo libro.
3. En la pestaña **Programador**, en el grupo **Código**, haga clic en **Visual Basic**.

Eliminar una Macro

4. En la ventana Explorador de proyectos, haga clic con el botón secundario en el objeto **ThisWorkbook** y, a continuación, haga clic en **Ver código**.

Sugerencia Si la ventana Explorador de proyectos no está visible, en el menú **Ver**, haga clic en **Explorador del proyecto**.

5. En la lista **Objeto** situada encima de la ventana Código, seleccione **Libro**.

De esta manera se crea un procedimiento vacío para el evento **Open**, como el siguiente:

```
Private Sub Workbook_Open()


End Sub
```

6. Agregue al procedimiento las líneas de código siguientes:

```
Private Sub Workbook_Open()
    MsgBox Date
    Worksheets("Hoja1").Range("A1").Value = Date
End Sub
```

7. Cambie a Excel y guarde el libro como libro habilitado para macros (.xlsm).
8. Cierre y vuelva a abrir el libro. Al abrir de nuevo el libro, Excel ejecuta el procedimiento Private Sub Workbook_Open, que muestra la fecha actual en un cuadro de mensaje.
9. Haga clic en **Aceptar** en el cuadro de mensaje.

Observe que la celda A1 de la Hoja1 también contiene la fecha, como resultado de ejecutar el procedimiento Private Sub Workbook_Open.

Eliminar una macro

1. Siga uno de los procedimientos siguientes:
   - Abra el libro que contiene la macro que desee eliminar.
   - Si la macro que quiere eliminar está almacenada en el libro de macros personal (Personal.xlsb) y este libro se encuentra oculto, proceda de la siguiente manera para mostrarlo:
     1. En la ficha **Ver**, en el grupo **Ventana**, haga clic en **Mostrar**.
     2. En **Mostrar libro**, haga clic en PERSONAL y después en **Aceptar**.
2. Si la ficha **Programador** no está disponible, haga lo siguiente para mostrarla:
   1. Haga clic en la pestaña **Archivo**.
   2. Haga clic en **Opciones** y, a continuación, haga clic en **Personalizar cinta de opciones**.
   3. En la categoría **Personalizar cinta de opciones**, en la lista **Fichas principales**, active la casilla de verificación **Programador** y, a continuación, haga clic en **Aceptar**.
1. En la ficha **Programador**, en el grupo **Código**, haga clic en **Macros**.

2. Seleccione el libro que contiene la macro que desea eliminar en la lista **Macros en**. Por ejemplo, haga clic en **Este libro**.
3. En el cuadro **Nombre de la macro**, haga clic en el nombre de la macro que

4. Haga clic en **Eliminar**.

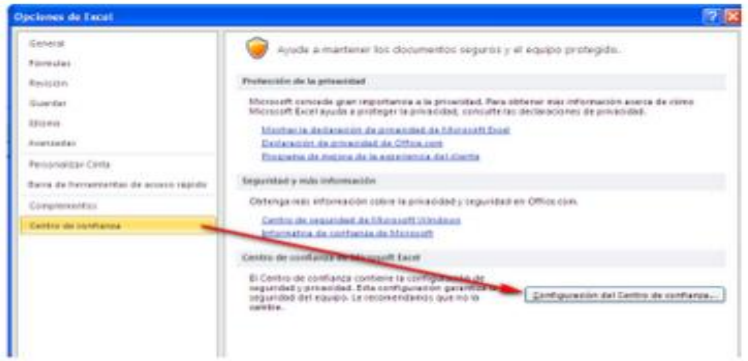
Configuración del Centro de Confianza

Activar Macros y controles ActiveX en Excel 2010

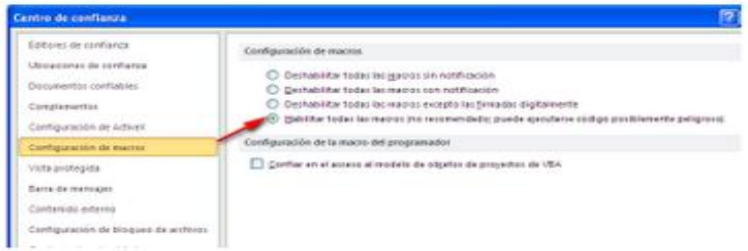
Quien haya descargado e instalado Excel 2010 descubrirá, como en Excel 2007, que por defecto la pestaña de Programador no aparece en la cinta. Esto se debe a que por defecto las macros no están habilitadas en Excel 2010. Tampoco tenemos acceso a los controles (formulario y ActiveX).

Para poder usar las macros y los controles en Excel 2010 tenemos que seguir los siguientes pasos:
Activamos la pestaña Archivo y apretamos el botón Opciones

Activamos el Centro de Confianza y apretamos el botón Configuración del centro de confianza



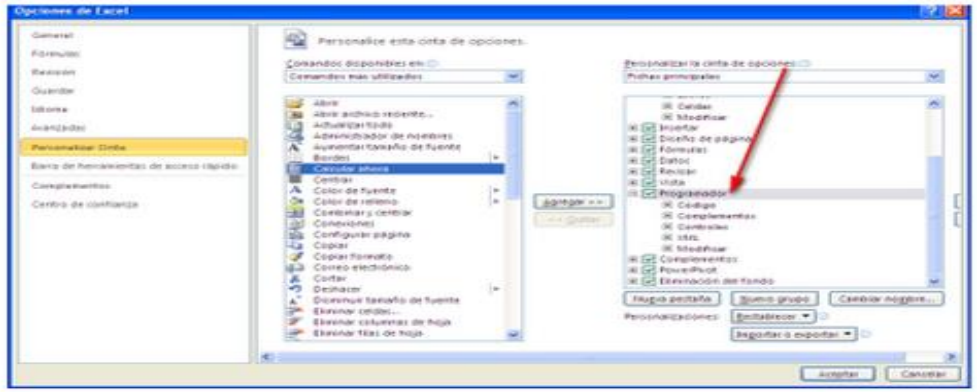
En el centro e confianza activamos la opción Configuración de macros y habilitamos la opción Habilitar todas las macros



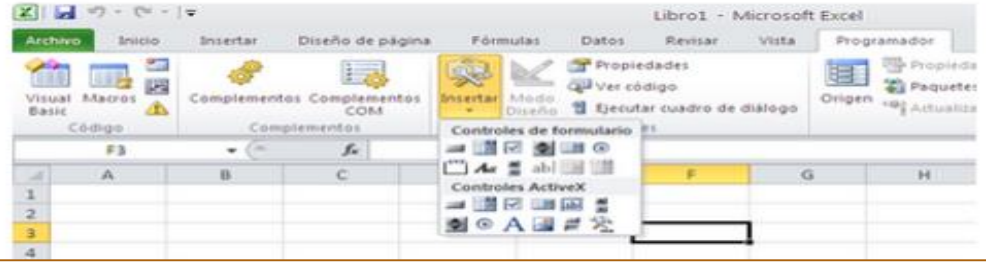
Apretamos Aceptar con lo que habremos habilitado las macros en nuestra copia de Excel 2010.

Configuración del Centro de Confianza

Sin embargo la pestaña de Programador no aparecerá. Para hacerla aparecer usamos nuevamente el menú Opciones de Archivo y en Personalizar la cinta de opciones marcamos Programador



A partir de ese momento podemos grabar, editar y crear macros y también usar controles en nuestros cuadernos.



Utilidad de una Macro

Anteriormente definimos a las macros como algo que nos permite expandir e incrementar las prestaciones de la hoja Excel. ¿Cuáles serían algunos ejemplos de esto? Bueno, los ejemplos los iremos viendo en este manual, pero a grandes rasgos podemos comentar que las 4 grandes "áreas" donde se aplican las macros son:

1. Automatización de procesos

¿Cansado de realizar tareas repetitivas o manuales en Excel? Las macros nos permiten grabar o definir nuestros procesos y luego ejecutarlos automáticamente con 1 clic. Nos pueden ahorrar muchas horas de trabajo! Con las macros podrás hacer tu trabajo mucho más rápido. Hemos visto casos extremos de trabajos en Excel que llevaban 6 o 7 horas de armado manual y luego pudieron ser automatizados en una macro que hacía todo en pocos segundos!

2. Creación de funciones a medida

¿Deseas crear nuevas funciones y cálculos personalizados? Las funciones y fórmulas Excel son la esencia de la hoja de cálculos. Podemos programar funciones a medida que hagan exactamente lo que nosotros queremos. Y esas funciones se comportarán igual que las de Excel (aparecerán en el menú de funciones en la categoría que nosotros indiquemos, tendrán sus respectivos argumentos, etc).

3. Creación de nuevos comandos, complementos y menús

Excel trae una gran cantidad de comandos y menús predefinidos que nos asisten para las operaciones más habituales. Las macros nos permiten crear nuestros propios comandos y menús personalizados, e incorporarlos al Excel. La utilidad de los mismos depende tan solo de nuestras necesidades. Los complementos Excel también están creados con macros.

4. Creación de aplicaciones a medida

Excel es utilizado en diversos campos y por una gran cantidad de usuarios. Las macros te permitirán construir complejas y elegantes aplicaciones para cualquier uso que quieras darles. El límite solo es tu imaginación. Una aplicación Excel consiste en algo más que una simple plantilla con datos y fórmulas. Una aplicación Excel es un verdadero programa de software con una serie de características que lo hacen utilizable por cualquier usuario sin que el mismo tenga que entender la lógica "exceliana" que hay por detrás.

5. Formularios

Finalmente, con las macros podremos armar todo tipo de formulario para entrada y gestión de datos. Dichos formularios pueden tener botones, listas desplegables, y todo las herramientas que encuentras en formularios profesionales. Además, puedes hacer que los datos del formulario se vayan guardando en una tabla Excel para posteriores análisis y reportes!

Grabar una Macro con referencias Relativas y Absolutas

Referencias Relativas y Absolutas

Cuando Excel construye las macros a través de grabar las acciones o tareas que se están realizando, normalmente graba referencias absolutas a las celdas. Es decir, cuando se selecciona una celda, recordará o almacenará la posición exacta de esa celda dentro de la Hoja, y no su su posición relativa respecto de la celda anteriormente activa.

Así por ejemplo, en el caso de nuestra macro *semana*, podemos ver como hace referencia a una posición específica para la primera celda activa, esto es **B1**. En este caso diremos entonces que se han utilizado referencias absolutas. El uso de este tipo de referencias implicará que al ejecutar la macro se ejecutarán las mismas tareas programadas y exactamente en las mismas celdas, sin reparar en que celda se encontraba activa al momento de ejecutar la macro.

En el caso de que se desee ejecutar una macro, partiendo de la posición en que se encuentra activa una celda (sea hacia la derecha, izquierda, arriba o abajo, según corresponda) deberán utilizarse *referencias relativas*.

Realicemos ahora un ejemplo con este tipo de referencias. Supongamos que nos encontramos en la celda B4 y que necesitamos que tres filas más arriba se escriban automáticamente los días de Lunes a Viernes, esto es desde B1 a F1. Entonces comenzaremos a grabar nuestra macro, pero antes haremos clic en el icono de *Referencia relativa* que aparece al costado de la opción *Detener Grabación*.

Si revisamos posteriormente veremos que la siguiente será la codificación que ahora nos mostrará Excel:

```
Sub relativa()
'
' relativa Macro
' Macro grabada el 25/05/2004 por Marci-Anto
'
    ActiveCell.Offset(-3, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Lunes"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Martes"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Miércoles"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Jueves"
    ActiveCell.Offset(0, 1).Range("A1").Select
    ActiveCell.FormulaR1C1 = "Viernes"
    ActiveCell.Offset(0, -4).Range("A1").Select'

End Sub
```

Vemos ahora como la referencia hacia la celda activa ya no es absoluta como antes (B1), sino que ahora hace referencia acerca de cuantas filas y columnas hacia arriba, abajo, izquierda o derecha respecto de la posición original. En este caso específico a tres filas hacia arriba en relación a la celda que se encontraba activa al momento de ejecutar la macro (-3,0).

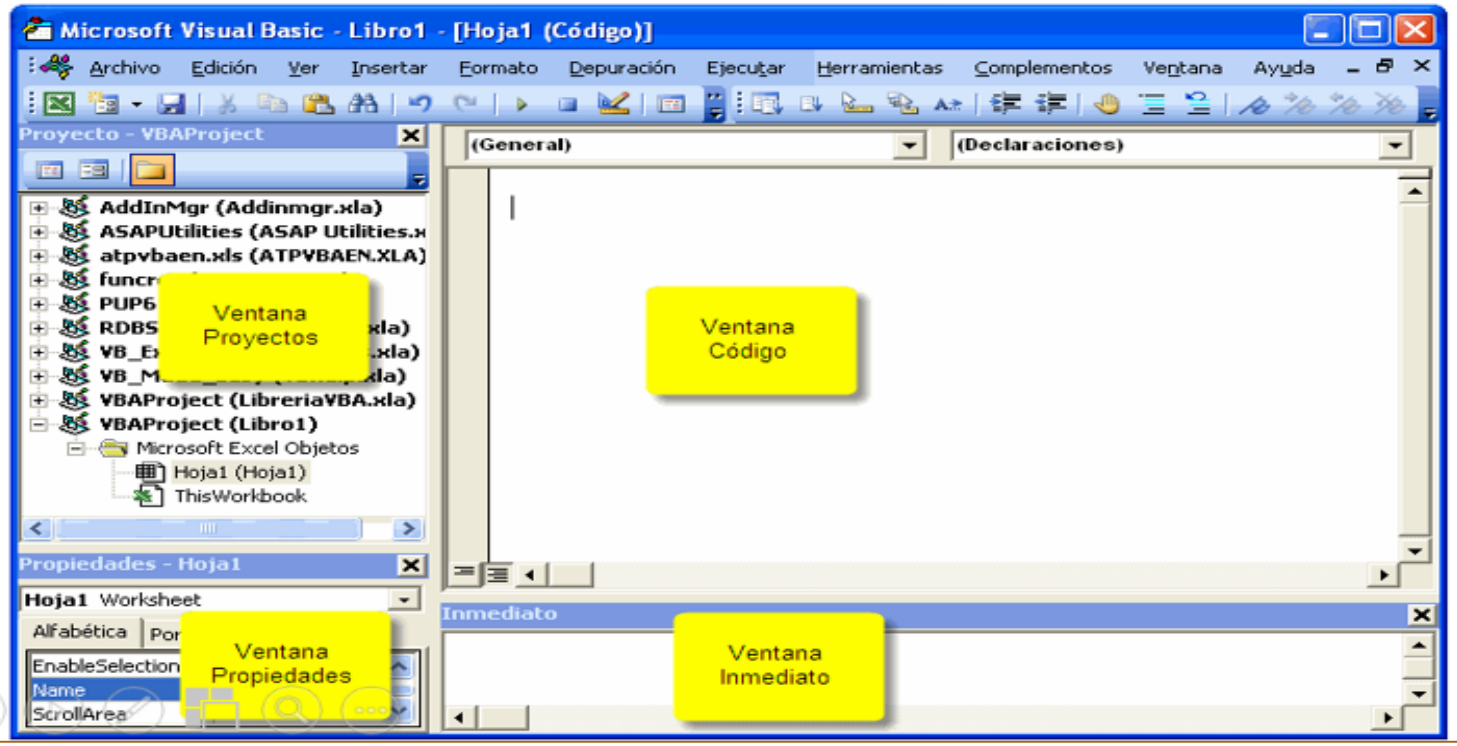
Editor de Visual Basic

1. Desde la ficha Programador > botón Visual Basic
2. Desde el teclado (Teclas de Método Abreviado): ALT+F11

(el acceso a la ficha programador lo explicamos en la sección anterior)

➔ Nuestra forma preferida de acceder al editor de macros es con las teclas ALT + F11

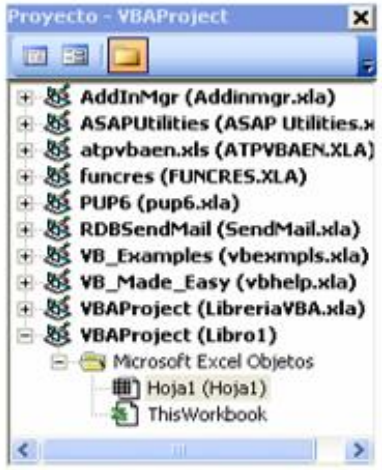
Una vez que hayas accedido al editor de visual basic verás algo similar a la figura siguiente (no importa si no lo ves estrictamente similar, eso dependerá de las ventanas que tengas visibles y ocultas). Puedes mostrar u ocultar estas ventanas desde el menú Ver.



Editor de Visual Basic

la Ventana Proyecto (parte izquierda superior)
la Ventana de Código (parte derecha)
la Ventana Propiedades (parte izquierda inferior)
la Ventana Inmediato (parte inferior derecha)

Ventana Proyecto - VBA Project: esta ventana muestra los libros excel (xls) o los complementos (xla) abiertos. Usualmente verás nombres del tipo "VBAProject" y entre paréntesis el nombre del archivo o complemento excel. Veamos el caso de VBAProject (Libro1). Simplemente significa que tienes abierto un libro Excel llamado Libro 1. Luego cuelgan 3 carpetas más: Hoja1 (Hoja1), ThisWorkbook y Modulo (no importa si ves todos estos elementos ahora, luego te enseñaremos como activarlos). Estas carpetas es donde habitan las macros. Haciendo doble clic en ellas activarás la ventana donde se escriben las macros.



Editor de Visual Basic

Ventana de Código: esta es el lugar donde escribiremos el código propiamente dicho de las macros. Como no hemos escrito ninguna macro todavía veremos la hoja en blanco. Recuerda bien estas dos ventanas, ya que las usaremos a continuación para escribir nuestra primera macro.

Como verás el Editor de VB tiene muchas ventanas. Pero de momento solo nos interesan las dos ventanas que te indicamos anteriormente: la Ventana Proyecto – VBA Project y la Ventana de Código donde se escribe el código de las macros. La Ventana de Propiedades e Inmediato las dejaremos para más adelante.

Recuerda: presionando las teclas ALT+F11 puedes ir desde Excel al Editor o desde el Editor al Excel de forma alterna (debes mantener presionada la tecla ALT y presionar F11 repetidas veces, verás como pasas de Excel al editor y viceversa).

Si estas en el editor y quieres regresar a la hoja Excel también puedes utilizar el ícono de Excel que se encuentra en el menú superior del editor (primer ícono de la izquierda, con la X de Excel).

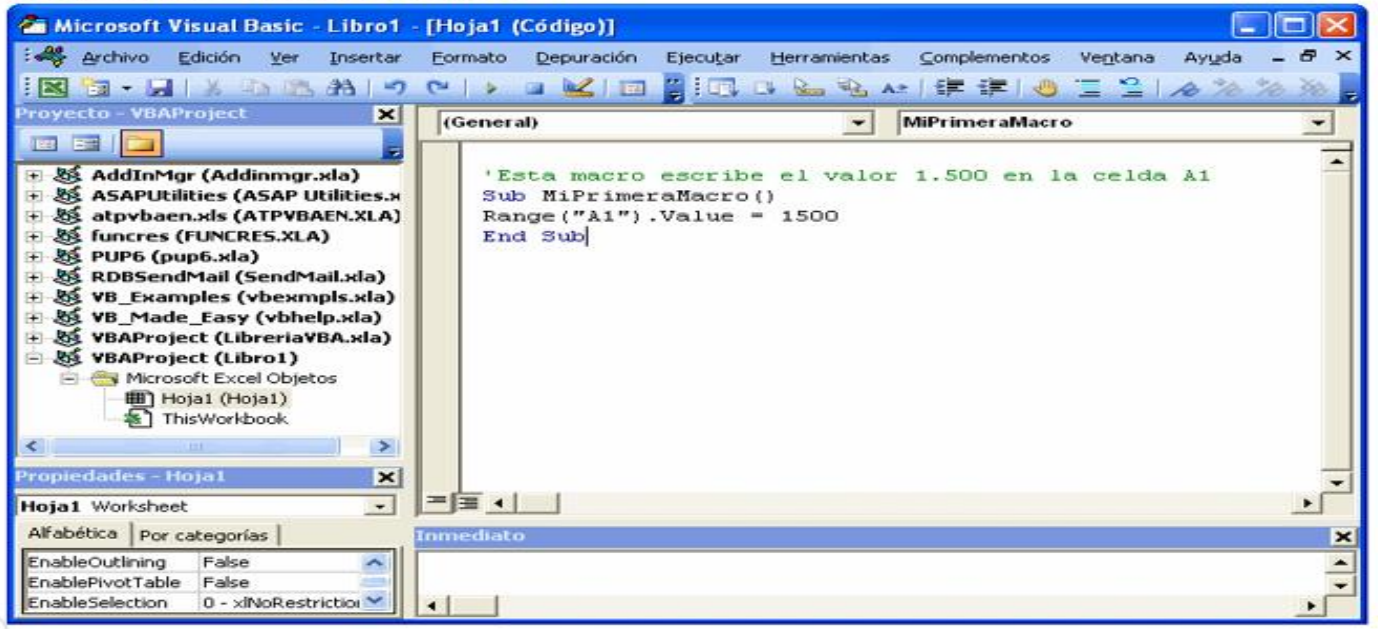
Ejemplo de Macro Programada

Nuestro objetivo fue crear una macro que escribiera el valor 1500 en la celda A1 de la Hoja1 de Excel. Como era una macro sencilla decidimos escribir el código manualmente. Vamos a resumir todos los pasos que hicimos hasta aquí:

1. Creamos un nuevo libro Excel y lo guardamos con el nombre Libro1.
2. Accedimos el Editor de Visual Basic con las teclas ALT+F11
3. En la Ventana Proyecto, en VBAProject (Libro1), hicimos doble clic en Hoja1 (Hoja1).
4. En la Ventana de Código escribimos textualmente el siguiente código:

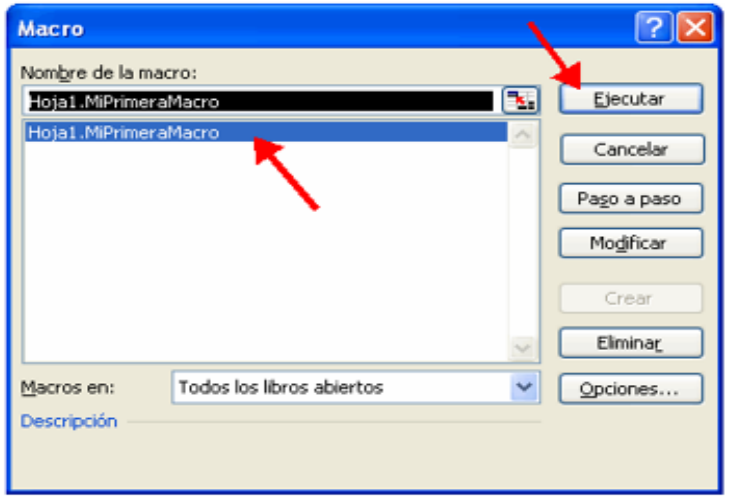
```
' Esta macro escribe el 1500 en la celda A1
Sub MiPrimeraMacro()
Range("A1").Value = 1500
End Sub
```

Todo quedó como se aprecia en la foto...



Ejemplo de Macro Programada

5. Finalmente abrimos el menú de macros con ALT+F8, seleccionamos MiPrimeraMacro desde la lista de nombres y presionamos Ejecutar.



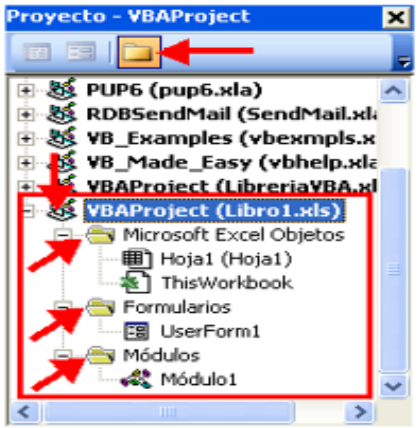
6. El resultado es que en la celda A1 se vuelve a escribir el valor 1500, que fue justamente lo que le indicamos a la macro que haga.

	A	B	C
1	1500		
2			
3			
4			
5			
6			

Recuerda: en estos 5 pasos hemos visto rápidamente como escribir una macro sencilla y ejecutarla. En las secciones siguientes iremos profundizando más estos temas, sobre todo el referente el código de la macro.

Explorador de Proyectos

En la ventana de Proyecto hay carpetas donde se guardarán y escribirán las macros. Hemos presionado el botón Alternar Carpetas para que se ordenen mejor los elmentos, como se ve a continuación:



Proyectos

En un primer nivel tenemos los proyectos. Por regla general, cada libro excel tiene asociado un proyecto. Por ejemplo, si creas un nuevo libro Excel desde el menú de Excel Archivo > Nuevo y vuelves al editor de macros (ALT+F11), verás que aparece el proyecto VBAProject(Libro1). Dentro del paréntesis aparece el nombre de tu libro excel y lo de VBAProject podrás cambiarlo luego.

En resumen VBAProject(Libro1) es la carpeta principal, asociada a un libro Excel determinado, donde insertaremos todo lo referido a las macros para dicho libro.

Carpetas

En un segundo nivel tenemos 3 carpetas:

- Microsoft Excel Objetos
- Formularios
- Módulos.

Explorador de Proyectos

Veamos cada una de ellas:

1. Microsoft Excel Objetos

En esta carpeta vemos que cuelgan dos elementos:

1.1 ThisWorkbook siempre está presente. Si escribimos una macro aquí la misma afectará a todo el libro.

1.2 Hoja1(Hoja1) hace referencia las hojas de Excel (habrá 1 por cada hoja de nuestro libro). Si escribimos una macro aquí la misma solo afectará a la hoja en cuestión.

2. Formularios

Los formularios son más conocidos como UserForms. Si no lo visualizas puedes agregarlos desde el menú Insertar > Userform.

2.1 Dentro de la carpeta de Formularios vemos un elemento llamado Userform1.

Todo el tema de formularios lo veremos con más en detalle en el capítulo "Formularios".

3. Módulos

Los módulos sirven para escribir macros a nivel genérico, sin estar relacionadas a la hoja o libro en particular. Si no visualizas ninguno puedes agregarlos desde el menú Insertar > Módulo.

3.1 Dentro de la carpeta de Módulos vemos que hay un elemento llamado Módulo1. Podemos insertar tantos módulos como necesitemos. En los módulos podemos escribir macros que operan de forma genérica, sin distinguir entre hojas o libros.

Haciendo doble clic izquierdo en cualquiera de estos objetos verás que se habilita la Ventana de Código de la izquierda (una hoja en blanco grande). En la misma es donde se escriben las macros.

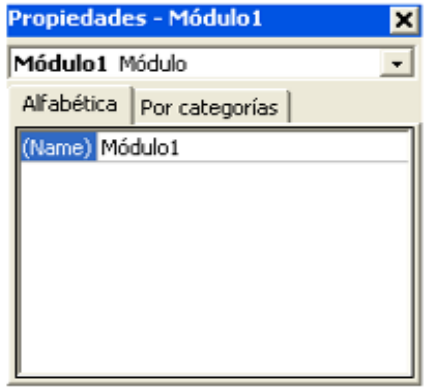
Recuerda: antes de escribir una macro debes evaluar donde hacerlo.

1. Si es una macro que solo debe afectar una hoja en particular escríbela en los objetos de Hoja, en el nombre de Hoja correspondiente.
2. Si la macro debe afectar a todo un libro en particular escríbela en el objeto ThisWorkbook. Estas suelen ser macros que se ejecutan al abrir, cerrar o guardar el libro.
3. Si la macro es de tipo genérica escríbela en un Módulo.
4. Si quieres hacer un formulario utiliza el objeto Userform.

Ventana de Propiedades

Veamos como se ve la Ventana Propiedades. Esta ventana se ubica en la parte inferior izquierda de la pantalla. Si no la puedes visualizar puedes activarla desde el menú Ver > Ventana propiedades.

Cada objeto tiene sus propiedades. Por ejemplo, si hacemos doble clic en el objeto Módulo1 que vimos en la sección anterior, podremos ver sus propiedades:



Algunos objetos tienen muchas propiedades (por ej. los Userforms) así que tenemos la opción de ordenar lso mismos de forma alfabética o por categoría. El objeto Módulo1 solo tiene la propiedad (Name)Módulo1. Si quisiéramos podríamos reemplazar el nombre de Módulo1 y asignar otro nombre a gusto.

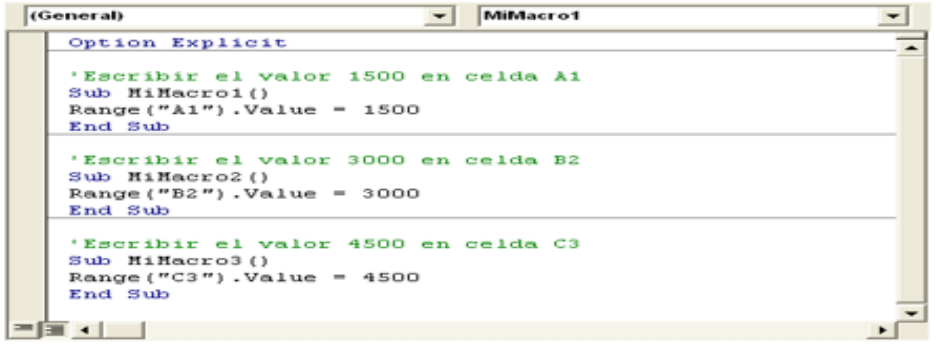
Consejo: en la medida que vamos insertando muchas macros en nuestro proyecto, conviene ir creando nuevos módulos con nombres apropiados para mantener ordenadas nuestras macros.

El resto de las propiedades de los otros objetos escapa al alcance de esta sección, pero podrás ver algunas en la 2º parte de este curso.

Ventana de Código

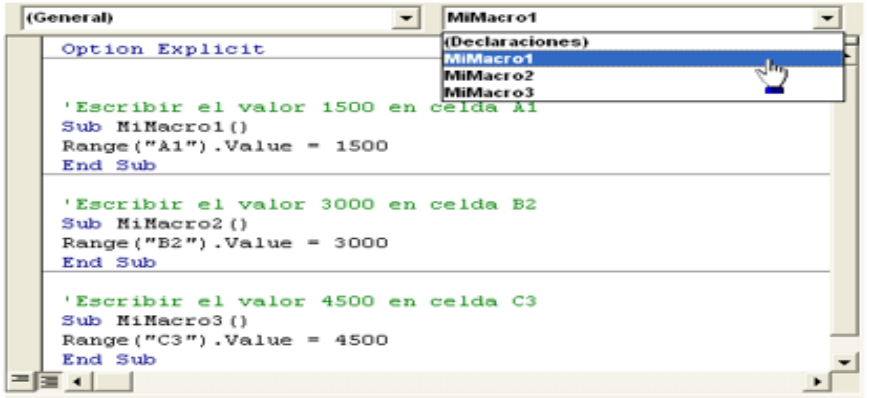
Veamos una foto para ver como se ve la Ventana de Código. Cada vez que hagas doble clic izquierdo en algún Objeto de la Ventana Proyecto se activará a la derecha la Ventana de Código. Si no la puedes visualizar puedes activarla posicionándote en el Objeto en cuestión y luego desde el menú Ver > Código.

Si todavía no has escrito ninguna macro verás la ventana de código en blanco. En nuestro ejemplo hemos escrito 3 macros:



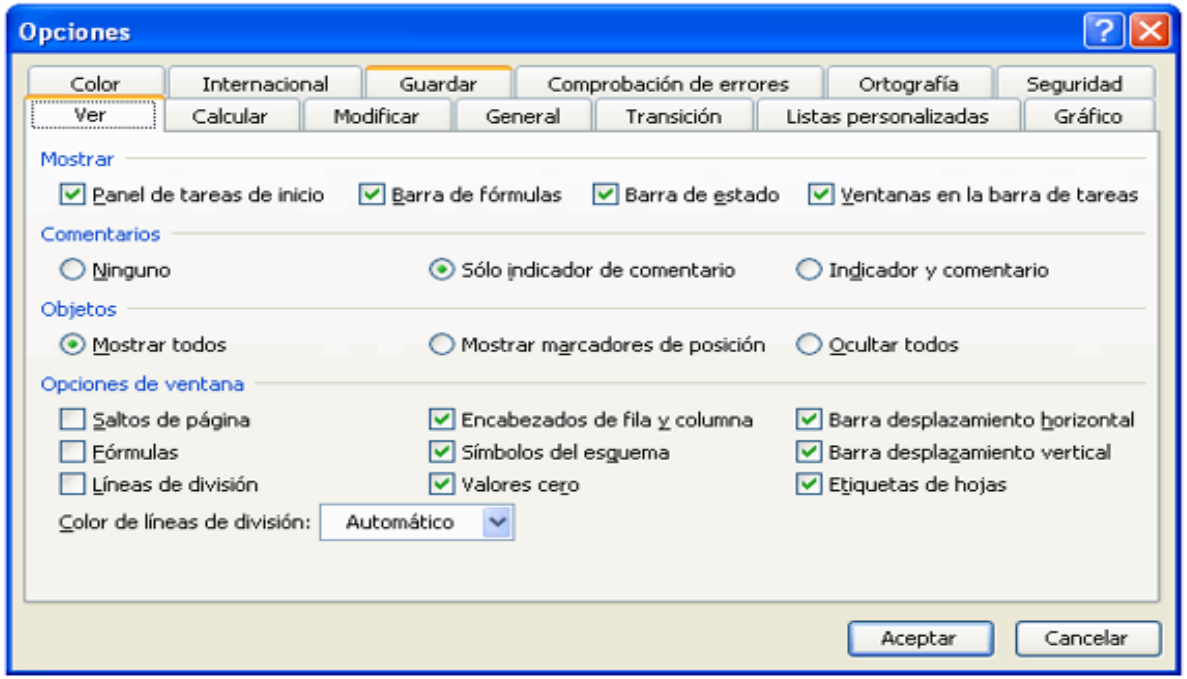
Esta ventana no tiene mayores complicaciones. Lo que interesa es más bién el código que escribamos aquí. Puedes escribir todas las macros que quieras en esta ventana. Recuerda que el mismo editor asignará automáticamente los colores al código y además separará con una línea contínua cada macro.

En la medida que empecemos a escribir muchas macros, existe una forma rápida de movernos entre ellas. Desde la lista desplegable de la parte superior derecha podremos ver la lista de macros escritas. Haciendo clic en cualquiera de ellas el editor nos posicionará rápidamente en la misma.



Formularios

Un Formulario (o su denominación en inglés Userform) se utiliza para crear un Cuadro de Diálogo donde el usuario puede introducir información, o realizar otras operaciones. Al ejecutar muchas de las opciones del menú de Excel se abren formularios. Por ejemplo, desde el menú Herramientas > Opciones se abre un formulario como el siguiente, desde donde se pueden activar o desactivar distintas opciones de Excel.



Este formulario es muy completo y posee pestañas en la parte superior, casillas de selección (cuadraditos con tildes), casillas de opciones (círculos con un punto dentro) y listas desplegables (lista con una flechita que despliega distintas opciones).

También podemos crear nuestros propios formularios como se ve a contoinuación.

Formularios

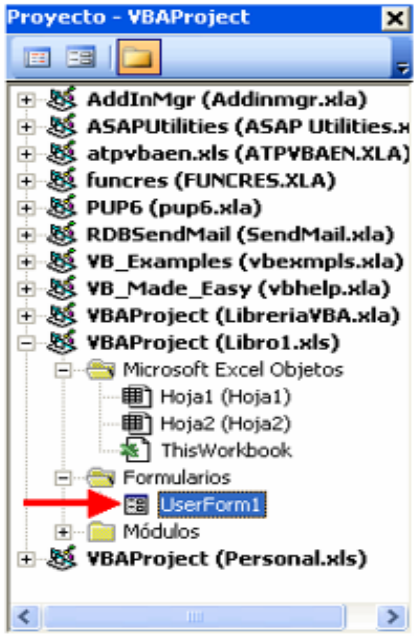
Los Userforms se utilizan mucho para crear aplicaciones Excel que luzcan de forma profesional y permitan al usuario introducir datos o elegir opciones de una forma guiada y más intuitiva.

En este capítulo aprenderemos como construir nuestros propios Userforms !

Crear un Formulario

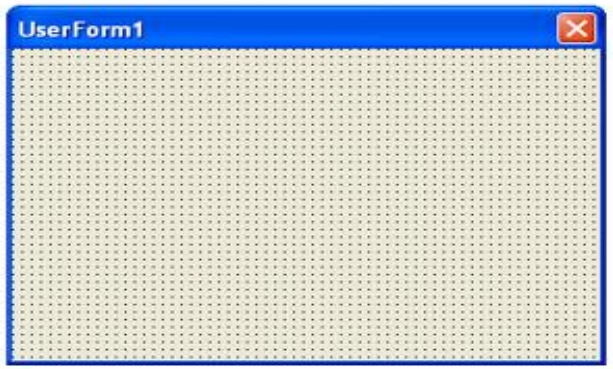
Para crear un Userform debemos hacer lo siguiente:

1. Abrir el editor de macros (por ejemplo, con CTRL+F11).
2. Insertar un objeto Userform. Esto podemos hacerlo fácilmente:
2.1 Seleccionamos el Libro Excel donde trabajaremos, en nuestro caso VBAProject (Libro2).
2.2 Hacemos clic derecho en el mismo y elegimos la opción Insertar > Userform
2.3 Como se ve en la fotografía, veremos que aparece un objeto Userform1 que cuelga de la carpeta Formularios



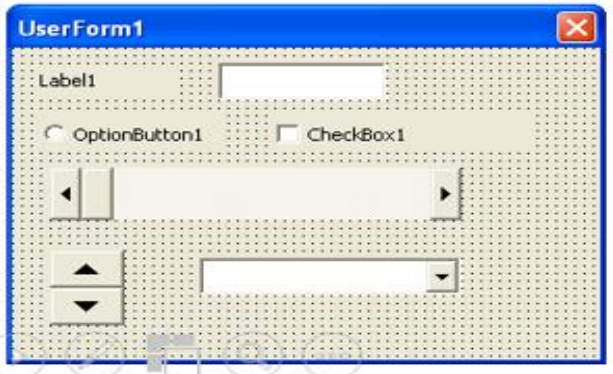
3. En la ventana de Código de la derecha, veremos que aparece un Userform en blanco, sin controles. Para abrir su Cuadro de Herramientas puedes hacerlo desde el menú Ver > Cuadro de Herramientas. Ahí figuran los controles que podremos agregar dentro del Userform. Para añdir uno, simplemente hacemos clic izquierdo en el control deseado y luego clic izquierdo dentro del Userform, en el sitio donde querramos agregar el control.

Crear un Formulario



La idea es que mediante estos controles podremos tanto capturar como enviar información o datos desde o hacia las celdas de Excel respectivamente. También podemos asociar macros a los controles que agreguemos al Userform (todo esto lo veremos en secciones siguientes).

Si te posicionas sobre los Controles del Cuadro de herramientas, verás que aparece su nombre. Simplemente hacemos clic izquierdo en el control deseado y luego clic izquierdo en el Userform, en el sitio donde querramos agregar el control. En nuestro caso hemos agregado algunos controles de forma desordenada dentro del Userform.



Controles del Formulario

Como veíamos anteriormente, tenemos una serie de controles para agregar al Userform, que los podíamos visualizar desde el menú Ver > Cuadro de Herramientas.

Ahora explicaremos brevemente la utilidad de cada control. Los controles se explican en el mismo orden que aparecen en el Cuadro de Herramientas y en la figura anterior (de izquierda a derecha y de arriba hacia abajo).

Seleccionar objetos: sirve para seleccionar controles que hayamos insertado en el Userform.

Label (etiqueta): sirve para poner un título o un texto.

Ejemplo: podemos poner un texto del tipo "Complete las opciones a continuación" y ubicarlo en cualquier sitio del Userform. También podemos agregar títulos o descripciones al resto de comandos que agregemos en el Userform.

Textbox (cuadro de texto): sirve para que un usuario introduzca datos.

Ejemplo: queremos que el usuario introduzca una fecha o un nombre (que luego llevaremos a alguna celda de Excel).

Controles del Formulario

ComboBox (cuadro combinado): sirve para que un usuario elija una opción de una lista.
Ejemplo: creamos una lista con los meses de Enero a Diciembre para que el usuario elija uno de ellos.

ListBox (cuadro de lista): sirve para que un usuario rellene o elija varias opciones de una lista.
Ejemplo: creamos una lista con Regiones o Ciudades y el usario deberá elegir una o varias de ellas.

CheckBox (casilla de verificación): sirve para que un usuario active una determinada función.
Ejemplo: podemos hacer que al cerrar el Userform se imprima un reporte solo si la casilla de selección está tildada.

OptionButton (botón de opción): sirve para que un usuario seleccione una opción determinada entre varias posibilidades.
Ejemplo: queremos que el usuario indique si es de sexo Masculino o Femenino. De todas las opciones solo se puede seleccionar una de ellas.

ToggleButton (botón de alternar): sirve para activar o desactivar alguna funcionalidad. Este botón adopta el modo "Encendido" / "Apagado".
Ejemplo: queremos que el usuario defina su idioma, en modo encendido español y en modo apagado inglés.

Frame (marco): sirve para agrupar elementos de un Userform (los elementos se deben ubicar dentro del Frame).
Ejemplo: tenemos varias grupos de OptionButton y para distinguirlos los agrupamos con un Frame. Si tenemos un grupo de opciones tipo masculino/femenino los agrupamos dentro de un frame. Si luego tenemos otro gurpo de opciones del tipo Mayor de Edad / Menor de Edad los agrupamos dentro de otro Frame.

Controles del Formulario

Frame (marco): sirve para agrupar elementos de un Userform (los elementos se deben ubicar dentro del Frame).

Ejemplo: tenemos varias grupos de OptionButton y para distinguirlos los agrupamos con un Frame. Si tenemos un grupo de opciones tipo masculino/femenino los agrupamos dentro de un frame. Si luego tenemos otro gurpo de opciones del tipo Mayor de Edad / Menor de Edad los agrupamos dentro de otro Frame.

CommandButton (botón de comando): es un simple botón que nos permite ejecutar acciones.

Ejemplo: un botón de Ayuda que ejecuta otro Userform con ayuda para el usuario.

TabStrip (barra de tabulaciones): en un mismo Userform se pueden crear distintas secciones.

Ejemplo: un userform con cuatro secciones: Norte, Sur, Este y Oeste. Dentro de cada sección podemos ubicar distintos controles.

MultiPage (página múltiple): en un mismo Userform se pueden crear distintas páginas.

Ejemplo: un userform con 2 páginas: España y Resto del Mundo. Dentro de cada página podemos ubicar distintos controles o distintas secciones.

ScrollBar (barra de desplazamiento): si tenemos una lista con muchos elementos el scrollbar nos permite navegarlos.

Ejemplo: tenemos una lista con 150 países. Con el ScrollBar podemos subir y bajar por la lista de los mismos utilizando las flechas de desplazamiento.

SpinButton (botón de número): permite aumentar o disminuir valores.

Ejemplo: tenemos una lista con tipos de interés y queremos que sean incrementados o disminuídos en cantidades predeterminadas desde el SpinButton.

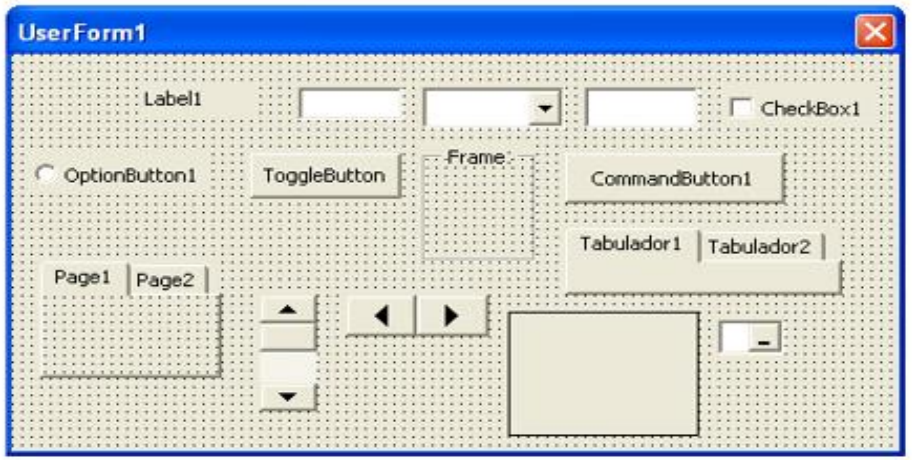
Controles del Formulario

Image (imagen): permite introducir imágenes en el Userform.
Ejemplo: queremos introducir una fotografía como fondo del Userform para darle un aspecto más profesional.

RefEdit (referencia): permite hacer referencia a una celda de Excel.
Ejemplo: queremos que el usuario seleccione un dato que fue introducido previamente en una celda Excel.

Haciendo doble clic en cada control se abrirá una venta donde podremos escribir y asociarle una macro. En la sección siguiente veremos un ejemplo simple sobre como crear un Userform paso a paso.

Para ver como se verían estos controles dentro del Userform, vamos a crear un Userform que contiene los 15 controles que se pueden agregar. Los hemos agregado en el mismo orden en que aparecen en el Cuadro de Herramientas.



Formulario de Excel

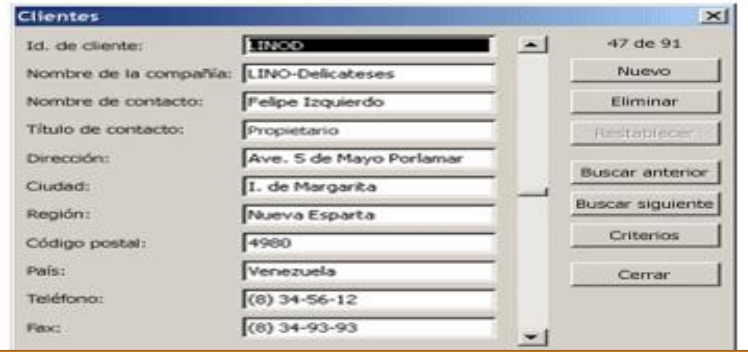
¿Qué es un formulario?

Un formulario, ya sea impreso o en línea, es un documento diseñado con formato y estructura estándar que facilita la captura, la organización y la edición de la información.

- Los formularios impresos contienen instrucciones, formato, etiquetas y espacios en blanco para escribir datos. Puede usar Excel y plantillas de Excel para crear formularios impresos.

- Los formularios en línea tienen las mismas características que los formularios impresos. Además, contienen controles, que son objetos que muestran datos o hacen que sea más fácil para los usuarios entrar o editar los datos, realizar una acción o seleccionar una opción. En general, los controles facilitan el uso de los formularios. Algunos ejemplos de controles comunes son los cuadros de lista, los botones de opción y los botones de comando. Los controles también pueden ejecutar macros asignadas y responder a eventos, tales como clics del mouse, mediante la ejecución de código de Visual Basic para Aplicaciones (VBA).

Excel puede generar



Tipos de formularios de Excel

Existen diversos tipos de formularios que puede crear en Excel: formularios de datos, hojas de cálculo que contienen controles ActiveX y de formulario, y formularios del usuario de VBA. Puede usar cada tipo de formulario por separado o puede combinarlos de diferentes maneras para crear una solución que sea apropiada para su caso particular.

FORMULARIO DE DATOS

Un formulario de datos brinda una forma conveniente de escribir o mostrar una fila completa de información en un intervalo o una tabla sin desplazarse horizontalmente. Verá que la entrada de datos será más sencilla con el uso de un formulario de datos, dado que no tendrá que desplazarse de columna en columna en caso de que tenga más columnas de datos que las que pueden verse en pantalla. Use un formulario de datos cuando sea suficiente un formulario simple de cuadros de texto que enumeren los encabezados de columna como etiquetas y no necesite características de formulario personalizadas ni sofisticadas, como un control de número o cuadro de lista.

automáticamente un formulario de datos integrado para el rango o la tabla. El formulario de datos muestra todos los encabezados de columna como etiquetas en un único cuadro de diálogo. Cada etiqueta tiene un cuadro de texto en blanco adyacente en el que el usuario puede escribir los datos para cada columna, hasta un máximo de 32 columnas. En un formulario de datos, puede agregar nuevas filas, buscar nuevas filas mediante navegación o, según el contenido de la celda, actualizar o eliminar filas. Si una celda contiene una fórmula, su resultado se muestra en el formulario de datos, pero no puede cambiar la fórmula mediante dicho formulario.

Controles ActiveX y de Formulario

HOJA DE CÁLCULO CON CONTROLES ACTIVEX Y DE FORMULARIO

Una hoja de cálculo es un tipo de formulario que permite que el usuario escriba datos y los vea en una cuadrícula; existen diversas características similares a los controles ya integradas en las hojas de cálculo de Excel, como validación de datos y comentarios. Las celdas se asemejan a cuadros de texto dado que el usuario puede escribir texto y aplicarles formato de diversas maneras. Las celdas con frecuencia se usan como etiquetas y, ajustando su alto y ancho, así como combinándolas, puede hacer que una hoja de cálculo se comporte como un simple formulario de entrada de datos. Otras características similares a los controles, como los comentarios de celda, los hipervínculos, las imágenes de fondo, la validación de datos, el formato condicional, los gráficos incrustados y el Filtro automático, pueden hacer que una hoja de cálculo se comporte como un formulario avanzado.

Para incrementar la flexibilidad, puede agregar controles y otros objetos de dibujo al lienzo de dibujo de una hoja de cálculo, y combinarlos y coordinarlos con las celdas de la hoja de cálculo. Por ejemplo, puede usar un control de cuadro de lista para facilitar la selección por parte del usuario de un elemento de una lista, o bien puede usar un control de control de número para facilitar la escritura de un número por parte de un usuario.

Dado que los controles y objetos se almacenan en el lienzo de dibujo, puede mostrarlos o verlos a lo largo de texto asociado que es independiente de los límites de fila y columna sin cambiar el diseño de una cuadrícula o tabla de datos en la hoja de cálculo. La mayor parte del tiempo, muchos de estos controles también pueden vincularse con celdas de la hoja de cálculo y no requieren código de VBA para hacer que funcionen. Puede establecer propiedades que determinan si un control flota libremente o se mueve y cambia de tamaño junto con una celda. Por ejemplo, probablemente tenga una casilla de verificación que desee mover junto con su celda subyacente cuando se ordene el intervalo. No obstante, si tiene un cuadro de lista que desee mantener en una ubicación específica en todo momento, probablemente no desee que se mueva junto con dicha celda.

Excel tiene dos tipos de controles: controles de formulario y controles ActiveX. Además de estos conjuntos de controles, también puede agregar objetos desde las Herramientas de dibujo, como autoformas, WordArt, elementos gráficos SmartArt o cuadros de texto.

Las siguientes secciones describen estos objetos de dibujo y controles, y además explican en más detalle cómo trabajar con estos controles y objetos.

Controles de formulario

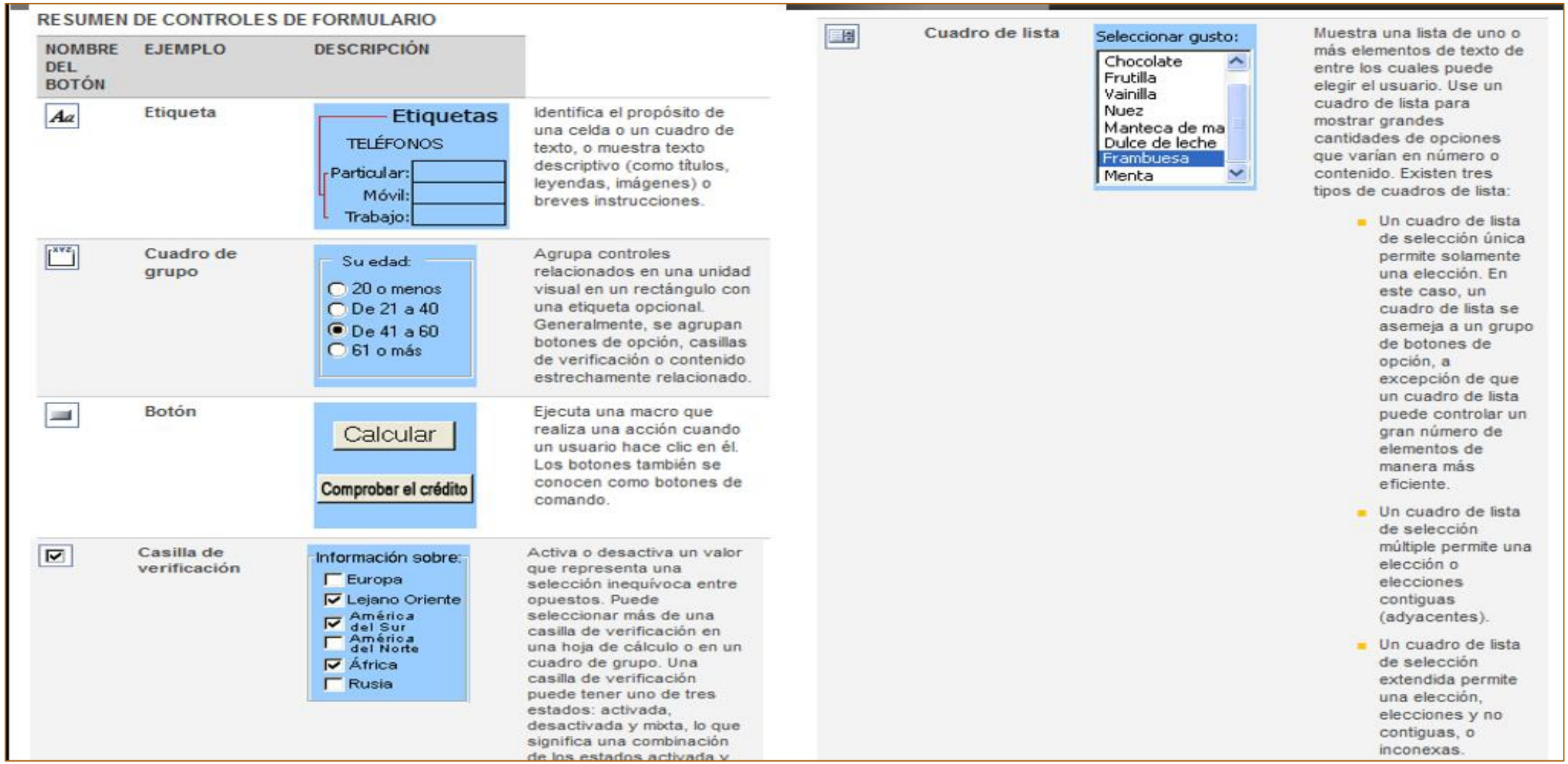
Los controles de formulario son controles originales que son compatibles con versiones anteriores de Excel, a partir de la versión 5.0 de Excel. Los controles de formulario también están diseñados para usarse en hojas de macros XLM.

Los controles de formulario se usan cuando se desea hacer referencia e interactuar fácilmente con datos de celda sin usar código de VBA y cuando se desea agregar controles a hojas de gráfico. Por ejemplo, después de agregar un control de cuadro de lista a una hoja de cálculo y vincularlo con una celda, puede devolver un valor numérico para la posición actual del elemento seleccionado en el control. Después, puede usar dicho valor numérico junto con la función **INDICE** para seleccionar elementos diferentes de la lista.

También puede ejecutar macros usando los controles de formulario. Puede adjuntar una macro existente a un control, o bien escribir o grabar una nueva macro. Cuando un usuario del formulario hace clic en el control, éste ejecuta la macro.

No obstante, estos controles no se pueden agregar a formularios del usuario, usarse para controlar eventos, ni modificarse para que ejecuten scripts web en páginas web.

Controles ActiveX y de Formulario



RESUMEN DE CONTROLES DE FORMULARIO

NOMBRE DEL BOTÓN	EJEMPLO	DESCRIPCIÓN
Etiqueta	Etiquetas TELÉFONOS Particular: Móvil: Trabajo:	Identifica el propósito de una celda o un cuadro de texto, o muestra texto descriptivo (como títulos, leyendas, imágenes) o breves instrucciones.
Cuadro de grupo	Su edad: 20 o menos De 21 a 40 De 41 a 60 61 o más	Agrupa controles relacionados en una unidad visual en un rectángulo con una etiqueta opcional. Generalmente, se agrupan botones de opción, casillas de verificación o contenido estrechamente relacionado.
Botón	Calcular Comprobar el crédito	Ejecuta una macro que realiza una acción cuando un usuario hace clic en él. Los botones también se conocen como botones de comando.
Casilla de verificación	Información sobre: Europa Lejano Oriente América del Sur América del Norte África Rusia	Activa o desactiva un valor que representa una selección inequívoca entre opuestos. Puede seleccionar más de una casilla de verificación en una hoja de cálculo o en un cuadro de grupo. Una casilla de verificación puede tener uno de tres estados: activada, desactivada y mixta, lo que significa una combinación de los estados activada y
Cuadro de lista	Seleccionar gusto: Chocolate Frutilla Vainilla Nuez Manteca de ma Dulce de leche Frambuesa Menta	Muestra una lista de uno o más elementos de texto de entre los cuales puede elegir el usuario. Use un cuadro de lista para mostrar grandes cantidades de opciones que varían en número o contenido. Existen tres tipos de cuadros de lista: ▪ Un cuadro de lista de selección única permite solamente una elección. En este caso, un cuadro de lista se asemeja a un grupo de botones de opción, a excepción de que un cuadro de lista puede controlar un gran número de elementos de manera más eficiente. ▪ Un cuadro de lista de selección múltiple permite una elección o elecciones contiguas (adyacentes). ▪ Un cuadro de lista de selección extendida permite una elección, elecciones y no contiguas, o inconexas.

Controles ActiveX y de Formulario

	Cuadro combinado		Combina un cuadro de texto con un cuadro de lista para crear un cuadro de lista desplegable. Un cuadro combinado es más compacto que un cuadro de lista pero requiere que el usuario haga clic en la flecha abajo para mostrar una lista de elementos. Use un cuadro combinado para permitir que un usuario escriba una entrada o elija solamente un elemento de la lista. El control muestra el valor actual en el cuadro de texto, sin importar el modo en que dicho valor se haya proporcionado.
	Barra de desplazamiento		Se desplaza por un intervalo de valores cuando el usuario hace clic en las flechas de desplazamiento o arrastra el cuadro de desplazamiento. Además, se puede mover por una página (en un intervalo preestablecido) de valores haciendo clic en el área entre el cuadro de desplazamiento y cualquiera de las flechas de desplazamiento. Generalmente, el usuario también puede escribir un valor de texto directamente en un cuadro de texto o una celda asociados.
	Control de número		Aumenta o disminuye un valor, como un incremento numérico, una hora o una fecha. Para incrementar el valor, es necesario hacer clic en la flecha arriba; para disminuirlo, se debe hacer clic en la flecha abajo. Generalmente, el usuario también puede escribir un valor de texto directamente en un cuadro de texto o una celda asociados.

Controles ActiveX

Los controles ActiveX pueden usarse en formularios de hoja de cálculo, con o sin el uso de código VBA, y en formularios del usuario de VBA. En general, use controles ActiveX cuando necesite requisitos de diseño más flexibles que los proporcionados por los controles de formulario. Los controles ActiveX tienen amplias propiedades que puede usar para personalizar su apariencia, comportamiento, fuentes y demás características.

También puede controlar los diversos eventos que se producen cuando se interactúa con un control ActiveX. Por ejemplo, puede realizar diferentes acciones, según qué opción seleccione el usuario en un control de cuadro de lista, o puede consultar una base de datos para rellenar un cuadro combinado con elementos cuando el usuario hace clic en un botón. También puede escribir macros que respondan a eventos asociados con controles ActiveX. Cuando un usuario del formulario interactúa con el control, el código de VBA se ejecuta para procesar cualquier evento que se produzca para dicho control.

El equipo también contiene muchos controles ActiveX instalados por Excel y otros programas, como el Control de calendario 12.0 y el Reproductor de Windows Media.

IMPORTANTE No se pueden usar todos los controles ActiveX directamente en hojas de cálculo. Algunos solamente pueden usarse en formularios del usuario de Visual Basic para Aplicaciones (VBA). Si intenta agregar alguno de estos controles ActiveX en particular a una hoja de cálculo, Excel muestra el mensaje "No se puede insertar el objeto".

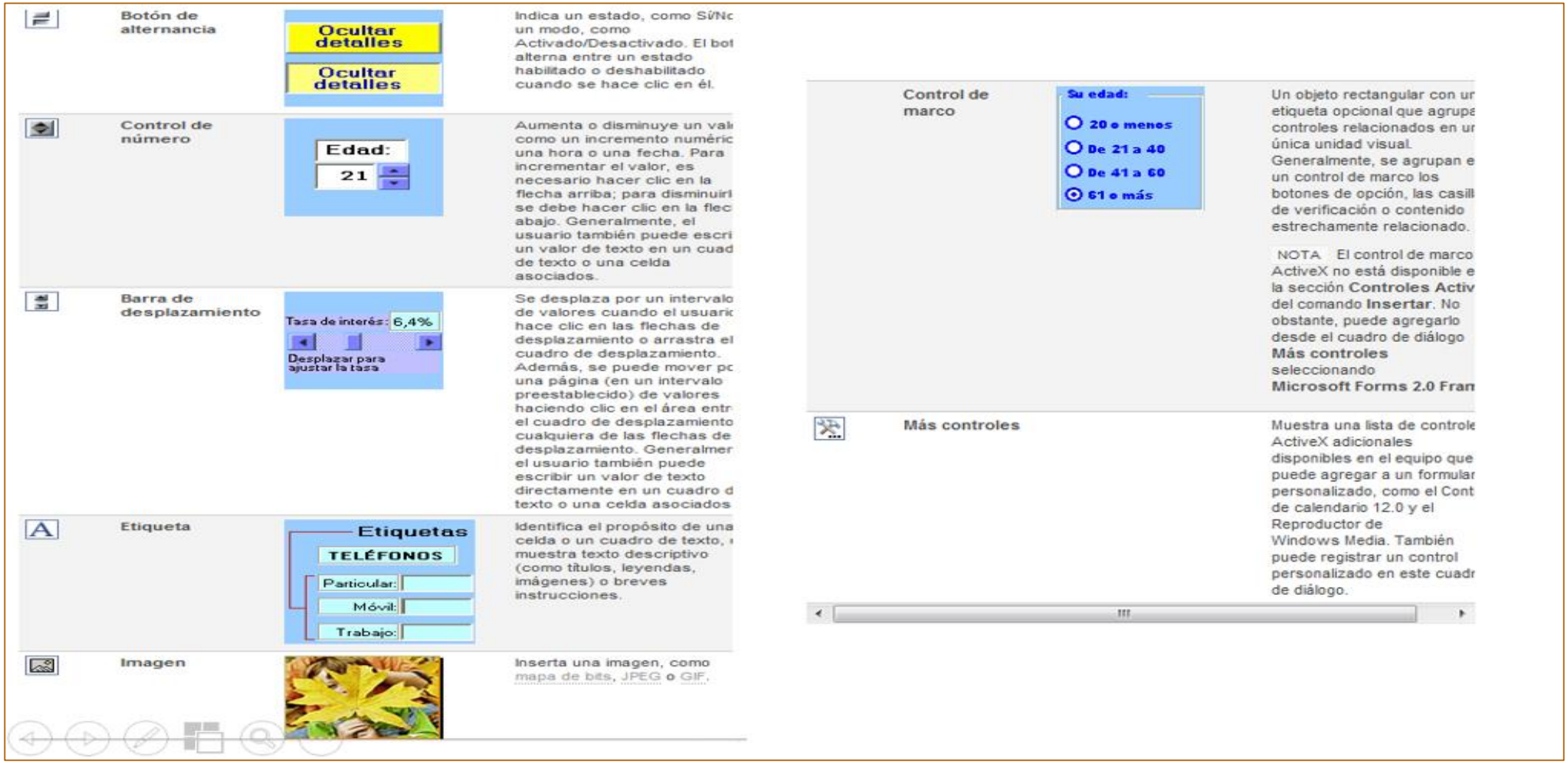
No obstante, los controles ActiveX no se pueden agregar a hojas de gráfico desde la interfaz de usuario ni a hojas de macros XLM. Tampoco se puede asignar una macro para ejecutarla directamente desde un control ActiveX del mismo modo en que puede hacerlo desde un control de formulario.

Controles ActiveX y de Formulario

RESUMEN DE CONTROLES ACTIVEX

NOMBRE DEL BOTÓN	EJEMPLO	DESCRIPCIÓN
	Casilla de verificación	Activa o desactiva un valor que representa una selecció inequívoca entre opuestos. Puede seleccionar más de u casilla de verificación a la ve en una hoja de cálculo o en cuadro de grupo. Una casilla de verificación puede tener uno de tres estados: activad desactivada y mixta, lo que significa una combinación de los estados activada y desactivada (como en una selección múltiple).
	Cuadro de texto	Permite al usuario ver, escrib o editar texto o datos enlazados a una celda, en u cuadro rectangular. Un cuad de texto también puede ser campo de texto estático que presenta información de sol lectura.
	Botón de comando	Ejecuta una macro que realiz una acción cuando un usuar hace clic en él.
	Botón de opción	Permite una única elección dentro de un conjunto limitad de opciones que se excluye mutuamente; generalmente está contenido en un marco cuadro de grupo. Un botón d opción puede tener uno de t estados: activado, desactiva y mixto, lo que significa una combinación de los estados activado y desactivado (com en una selección múltiple). L botones de opción también s conocen como botones de radio.
	Cuadro de lista	Muestra una lista de uno o m elementos de texto de entre cuales puede elegir el usuar Use un cuadro de lista para mostrar grandes cantidades opciones que varían en número o contenido. Existen tres tipos de cuadros de list • Un cuadro de lista d selección única permite solamente u elección. En este caso, un cuadro de lista se asemeja a u grupo de botones de opción, a excepción de que un cuadro de lista puede controlar un gran número de elementos de maner más eficiente. • Un cuadro de lista d selección múltiple permite una elección elecciones contigua (adyacentes). • Un cuadro de lista d selección extendida permite una elección elecciones y no contiguas, o inconexas.
	Cuadro combinado	Combina un cuadro de texto con un cuadro de lista para crear un cuadro de lista desplegable. Un cuadro combinado es más compact que un cuadro de lista pero requiere que el usuario haga clic en la flecha abajo para mostrar una lista de element Úselo para permitir que un usuario escriba una entrada elija solamente un elemento la lista. El control muestra el valor actual en el cuadro de texto, sin importar el modo e que dicho valor se haya proporcionado.

Controles ActiveX y de Formulario



Objetos como Herramientas de Dibujo

Objetos de herramientas de dibujo

Historial de cotizaciones de Blue Sky Airlines

Haga clic aquí para ir a la página web de cotizaciones actuales.

Fecha	Superior	Inferior
3-Abr	56 3/8	55 1/4
10-Abr	56	54 1/8
17-Abr	56 3/8	56
24-Abr	55 7/8	54 3/4

Probablemente también desee incluir elementos gráficos SmartArt, formas, WordArt y cuadros de texto en el formulario. Puede cambiar el tamaño, girar, voltear, colorear y combinar estos objetos para crear formas aún más complejas. Cuando escribe texto directamente en un objeto de cuadro de texto o una forma, el texto se convierte en parte del objeto, si gira o voltea el objeto, el texto gira o se voltea junto con él. A diferencia de los controles ActiveX, puede asignar diferentes atributos, como tamaño de fuente o estilo de fuente, a caracteres o palabras individuales en el objeto. También puede asignar macros y agregar hipervínculos a estos objetos. Incluso puede vincular texto en un objeto de cuadro de texto o una forma con una celda de una hoja de cálculo y mostrar dinámicamente valores actualizados en dichos objetos.

Trabajar con controles y objetos en el formulario de la hoja de cálculo

Después de agregar controles ActiveX y de formulario a un formulario de una hoja de cálculo, generalmente el usuario desea ajustar y reorganizar los controles de diversas maneras para crear un formulario fácil de usar y correctamente diseñado. Las tareas comunes son, entre otras, las siguientes:

- Controlar la visualización de las líneas de la cuadrícula mientras se trabaja con los controles y decidir si se muestran las líneas de la cuadrícula a los usuarios en el formulario de la hoja de cálculo final.
- Seleccionar y anular la selección de los controles para que el usuario pueda especificar propiedades o realizar ajustes adicionales.
- Editar texto en un control, como la leyenda o la etiqueta.
- Agrupar, copiar, mover y alinear controles para organizar el diseño del formulario de la hoja de cálculo.
- Cambiar el tamaño y aplicar formato a los controles para obtener la apariencia que desea.
- Posicionar o cambiar el tamaño de un control con una celda.
- Proteger controles y celdas vinculadas de acuerdo con sus necesidades de protección de datos específicas.
- Habilitar o deshabilitar la impresión de controles cuando se imprime el formulario de la hoja de cálculo.
- Eliminar controles no usados.

Puede diseñar un formulario de hoja de cálculo con o sin líneas de la cuadrícula de celda en el fondo. Por ejemplo, probablemente desee desactivar las líneas de la cuadrícula de celdas y luego aplicar formato a todas las celdas con el mismo color o trama, o incluso usar una imagen como fondo de una hoja. Para ocultar o mostrar las líneas de la cuadrícula, en la ficha **Ver**, en el grupo **Mostrar u ocultar**, active o desactive la casilla de verificación **Líneas de la cuadrícula**.

Determinar el tipo de Control de una Hoja de Calculo

Determinar el tipo de control en una hoja de cálculo

Debido a que hay tres tipos diferentes de controles y objetos que puede modificar de forma única, probablemente no tenga certeza de qué tipo de control es con tan solo mirarlo. Para determinar el tipo de control (formulario o ActiveX), seleccione el control, haga clic con el botón secundario en él y luego muestre el menú contextual:

- Si el menú contextual contiene el comando **Propiedades**, el control es un control ActiveX y se encuentra en el modo de diseño.
- Si el menú contextual contiene el comando **Asignar macro**, el control es un control de formulario.

  SUGERENCIA Para mostrar el menú contextual correcto para el control de dro de grupo, asegúrese de seleccionar el perímetro en del cuadro de grupo.
- Si el menú contextual contiene el comando **Editar texto**, el objeto es un objeto de dibujo.

VOLVER AL PRINCIPIO

FORMULARIOS DEL USUARIO DE VBA

Para obtener una máxima flexibilidad, puede crear formularios del usuario, que son cuadros de diálogo personalizados que generalmente incluyen uno o más controles ActiveX. La disponibilidad de los formularios del usuario se establece mediante código de VBA creado en el Editor de Visual Basic. A continuación, se ofrece un esquema de los pasos para crear un formulario del usuario:

1. Inserte un formulario del usuario en el proyecto de VBA del libro. Para obtener acceso al proyecto de VBA de un libro, primero abra el Editor de Visual Basic (presione ALT+F11) y, a continuación, haga clic en **Formulario del usuario** en el menú **Insertar** del Editor de Visual Basic.
2. Escriba un procedimiento para mostrar el formulario del usuario.
3. Agregue controles ActiveX.
4. Modifique las propiedades para los controles ActiveX.
5. Escriba procedimientos del controlador de eventos para los controles ActiveX.

Mediante los formularios del usuario, también puede usar la funcionalidad de formularios avanzados. Por ejemplo, puede agregar mediante programación un botón de opción diferente para cada letra del alfabeto o puede agregar una casilla de verificación para cada elemento en una gran lista de fechas y números.

Antes de crear un formulario del usuario, considere usar cuadros de diálogo integrados disponibles desde Excel que puedan adaptarse a sus necesidades. Estos cuadros de diálogo integrados incluyen las funciones de VBA **CuadroEntr** y **CuadroMsj**, el método de Excel **InputBox**, el método **GetOpenFilename**, el método **GetSaveAsFilename** y el objeto Dialogs del objeto **Application**, que contiene todos los cuadros de diálogo de Excel integrados.

Casillas de Verificacion,Botones de Opción y Botones de Alternancia

Más información sobre casillas de verificación, botones de opción y botones de alternancia

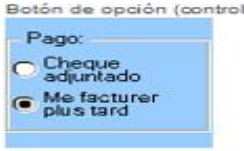
Casilla de verificación Permite a un usuario seleccionar o anular la selección de uno o varios valores en un grupo de opciones. Puede activar más de una casilla de verificación a la vez en una hoja de cálculo o en un cuadro de grupo. Por ejemplo, puede usar una casilla de verificación para crear un formulario de pedido que contenga una lista de artículos disponibles o en una aplicación de seguimiento de inventario para mostrar si se ha interrumpido la producción de un artículo.

Casilla de verificación (control de formulario)

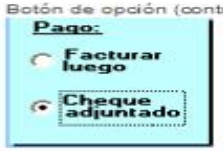
Casilla de verificación (control ActiveX)

Botón de opción Permite una sola opción de un conjunto limitado de opciones mutuamente excluyentes. Por lo general, un botón de opción (o botón de radio) está contenido en un marco o cuadro de grupo. Por ejemplo, se puede usar un botón de opción en un formulario de pedido para que un usuario pueda seleccionar un tamaño de un intervalo de tamaños, como pequeño, mediano, grande o extra grande. También se puede usar para seleccionar diferentes opciones de envío, como por tierra, expreso o al día siguiente.

Botón de opción (control de formulario)



Botón de opción (control ActiveX)



Botón de alternancia Indica un estado, como Sí o no, o un modo, como Activado o desactivado. El botón alterna entre un estado habilitado o deshabilitado cuando se hace clic en él. Se puede usar un botón de alternancia para, por ejemplo, alternar entre el modo de diseño y el modo de edición, o como una alternativa a una casilla de verificación.

Nota El botón de alternancia no está disponible como control de formulario, solo como control ActiveX.

Botón de alternancia (control ActiveX)

Casillas de Verificacion,Botones de Opción y Botones de Alternancia

Agregar una casilla de verificación (control de formulario)

1. Si la ficha **Programador** no está disponible, muéstrela.
   - Mostrar la ficha **Programador**
     1. Haga clic en la pestaña **Archivo**.
     2. Haga clic en **Opciones** y, a continuación, haga clic en la categoría **Personalizar cinta de opciones**.
     3. En la lista **Fichas principales**, active la casilla de verificación **Programador** y haga clic en **Aceptar**.
2. En la ficha **Programador**, en el grupo **Controles**, haga clic en **Insertar** y, a continuación, en **Controles de formulario**, haga clic en **Casilla de verificación**.

3. Haga clic en la ubicación de la hoja de cálculo donde desea que aparezca la esquina superior izquierda del control.
4. En la ficha **Programador**, en el grupo **Controles**, haga clic en **Propiedades**.

Sugerencia También puede hacer clic con el botón secundario en el control y, a continuación, hacer clic en **Formato de control**.

Para especificar las propiedades del control, siga este procedimiento:

1. En **Valor**, especifique el estado inicial de la casilla de verificación siguiendo uno de estos procedimientos:
   - Para mostrar una casilla de verificación que tiene una marca de verificación, haga clic en **Activada**. Una marca de verificación indica que la casilla de verificación está activada.
   - Para mostrar una casilla de verificación desactivada, haga clic en **Desactivada**.
   - Para mostrar una casilla de verificación con sombra, haga clic en **Mixta**. La sombra indica una combinación de los estados activado y desactivado; por ejemplo, cuando hay una selección múltiple.
2. En el cuadro **Vincular con la celda**, escriba una referencia de celda que contenga el estado actual de la casilla de verificación:
   - Cuando la casilla de verificación está activada, la celda vinculada devuelve un valor TRUE.
   - Cuando la casilla de verificación está desactivada, la celda vinculada devuelve un valor FALSE.

     Nota Cuando la celda vinculada está vacía, Excel interpreta el estado de la casilla de verificación como FALSE.
   - Si el estado de la casilla de verificación es mixto, la celda vinculada devuelve un valor de error #N/A.

Use el valor devuelto en una fórmula para responder al estado actual de la casilla de verificación.

Por ejemplo, un formulario de encuesta de viaje contiene dos casillas de verificación denominadas **Europa** y **Australia** en un cuadro de grupo **Lugares visitados**. Estas dos casillas de verificación están vinculadas a las celdas C1 (para Europa) y C2 (para Australia).

Cuando un usuario activa la casilla de verificación **Europa**, la siguiente fórmula de la celda D1 se evalúa como "Visitados en Europa":

```
=SI(C1=TRUE,"Visitados en Europa","Nunca visitó Europa")
```

Cuando un usuario desactiva la casilla de verificación **Australia**, la siguiente fórmula de la celda D2 se evalúa como "Nunca visitó Australia":

```
=SI(C2=TRUE,"Visitados en Australia","Nunca visitó Australia")
```

Si tiene tres estados para evaluar (**Activada**, **Desactivada** y **Mixta**) en el mismo grupo de opciones, puede usar las funciones **ELEGIR** o **BUSCAR** de forma similar.

Aplicación de Casilla de Verificación

Agregar una casilla de verificación (control ActiveX)

1. Si la ficha **Programador** no está disponible, muéstrela.
   - Mostrar la ficha **Programador**
     1. Haga clic en la pestaña **Archivo**.
     2. Haga clic en **Opciones** y, a continuación, haga clic en la categoría **Personalizar cinta de opciones**.
     3. En la lista **Fichas principales**, active la casilla de verificación **Programador** y haga clic en **Aceptar**.
2. En la ficha **Programador**, en el grupo **Controles**, haga clic en **Insertar** y, a continuación, en **Controles ActiveX**, haga clic en **Casilla de verificación**

3. Haga clic en la ubicación de la hoja de cálculo en la que desea que aparezca la esquina superior izquierda de la casilla de verificación.
4. Para editar el control ActiveX, asegúrese de que está en el modo Diseño. En la ficha **Programador**, en el grupo **Controles**, active el **Modo Diseño**.
5. Para especificar las propiedades del control, en la ficha **Programador**, en el grupo **Controles**, haga clic en **Propiedades**.

   Sugerencia También puede hacer clic con el botón secundario en el control y, a continuación, hacer clic en **Propiedades**.

   Aparece el cuadro de diálogo **Propiedades**. Para obtener información detallada acerca de cada propiedad, seleccione la propiedad y, a continuación, presione F1 para mostrar un tema de la Ayuda de Visual Basic. También puede escribir el nombre de la propiedad en el cuadro **Buscar** de la Ayuda de Visual Basic. En la siguiente sección se resumen las propiedades disponibles.

RESUMEN DE LAS PROPIEDADES POR CATEGORÍAS FUNCIONALES

SI DESEA ESPECIFICAR	USE ESTA PROPIEDAD
General:	
Si el control se carga al abrir el libro. (excepto para controles ActiveX)	**AutoLoad** (Excel)
Si el control puede recibir el foco y responder a eventos generados por el usuario.	**Enabled** (formulario)
Si se puede editar el control.	**Locked** (formulario)
El nombre del control.	**Name** (formulario)
La manera en que el control está unido a las celdas que están debajo de él (libre flotante, mover sin cambiar el tamaño o mover y cambiar el tamaño).	**Placement** (Excel)
Si se puede imprimir el control.	**PrintObject** (Excel)
Si el control está visible u oculto.	**Visible** (formulario)
Texto:	
La posición del control en relación con el título (izquierda o derecha).	**Alignment** (formulario)
Atributos de fuente (negrita, cursiva, tamaño, tachado, subrayado y grosor).	**Bold**, **Italic**, **Size**, **StrikeThrough**, **Underline**, **Weight** (formulario)
Texto descriptivo sobre el control que lo identifica o lo describe.	**Caption** (formulario)
La manera en que se alinea el texto en el control (izquierda, centro o derecha).	**TextAlign** (formulario)
Si el contenido del control se ajusta automáticamente al final de una línea.	**WordWrap** (formulario)
Datos y enlace:	
El rango que está vinculado al valor del control.	**LinkedCell** (Excel)
El contenido o estado del control.	**Value** (formulario)
Tamaño y posición:	
Si el tamaño del control se ajusta automáticamente para mostrar todo el contenido.	**AutoSize** (formulario)
El alto o ancho en puntos.	**Height**, **Width** (formulario)
La distancia entre el control y el borde izquierdo o superior de la hoja de cálculo.	**Left**, **Top** (formulario)

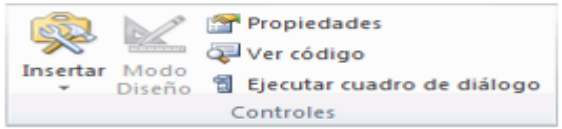
Aplicación del Botón Opción

Formato:	
El color de fondo.	**BackColor** (formulario)
El estilo de fondo (transparente u opaco).	**BackStyle** (formulario)
El color de primer plano.	**ForeColor** (formulario)
Si el control tiene una sombra.	**Shadow** (Excel)
La apariencia visual del borde (sin formato, en relieve, de bajo relieve, grabado o rugoso).	**SpecialEffect** (formulario)
Imagen:	
El mapa de bits que se muestra en el control.	**Picture** (formulario)
La ubicación de la imagen en relación con su título (izquierda, superior, derecha, etc.).	**PicturePosition** (formulario)
Teclado y mouse:	
La tecla de método abreviado para el control.	**Accelerator** (formulario)
Un icono del mouse personalizado.	**MouseIcon** (formulario)
El tipo de puntero que se muestra cuando el usuario sitúa el mouse sobre un objeto determinado (por ejemplo: estándar, flecha o cursor en I).	**MousePointer** (formulario)
Específicas para la casilla de verificación:	
Grupo de botones de opción mutuamente excluyentes.	**GroupName** (formulario)
Si un usuario puede especificar el estado Null para el control desde la interfaz de usuario.	**TripleState** (formulario)

Nota El tamaño de la casilla de verificación dentro del control y la distancia a la que se encuentra de su texto asociado no se pueden ajustar.

Agregar un botón de opción (control de formulario)

1. Si la ficha **Programador** no está disponible, muéstrela.
   - Mostrar la ficha **Programador**
     1. Haga clic en la pestaña **Archivo**.
     2. Haga clic en **Opciones** y, a continuación, haga clic en la categoría **Personalizar cinta de opciones**.
     3. En la lista **Fichas principales**, active la casilla de verificación **Programador** y haga clic en **Aceptar**.
2. En la ficha**Programador**, en el grupo **Controles**, haga clic en **Insertar** y, a continuación, en **Controles de formulario**, haga clic en **Botón de opción**



3. Haga clic en la ubicación de la hoja de cálculo donde desea que aparezca la esquina superior izquierda del botón de opción.
4. En la ficha **Programador**, en el grupo **Controles**, haga clic en **Propiedades**

Sugerencia También puede hacer clic con el botón secundario en el control y, a continuación, hacer clic en **Formato de control**.

Para establecer las propiedades del control, siga este procedimiento:

1. En **Valor**, especifique el estado inicial del botón de opción siguiendo uno de estos procedimientos:
   - Para mostrar un botón de opción activado, haga clic en **Activado**.
   - Para mostrar un botón de opción desactivado, haga clic en **Desactivado**.
2. En el cuadro **Vincular con la celda**, escriba una referencia de celda

Aplicación del Botón Opción

La celda vinculada devuelve el número del botón de opción activado en el grupo de opciones. Use la misma celda vinculada para todas las opciones de un grupo. El primer botón de opción devuelve un 1, el segundo botón de opción devuelve un 2, etc. Si tiene dos o más grupos de opciones en la misma hoja de cálculo, use una celda vinculada diferente para cada grupo de opciones.

Use el número devuelto en una fórmula para responder a la opción seleccionada.

Por ejemplo, un formulario de personal, con un cuadro de grupo **Tipo de trabajo**, contiene dos botones de opción con los nombres **Tiempo completo** y **Tiempo parcial** vinculados a la celda C1. Después de que un usuario selecciona una de las dos opciones, la siguiente fórmula de la celda D1 se evalúa como "Tiempo completo" si se selecciona el primer botón de opción o como "Tiempo parcial" si se selecciona el segundo botón de opción.

```
=SI(C1=1,"Tiempo completo","Tiempo parcial")
```

Si tiene tres o más opciones para evaluar en el mismo grupo de opciones, puede usar las funciones **ELEGIR** o **BUSCAR** de forma similar.

Agregar Botones de Alternancia

Agregar un botón de alternancia (control ActiveX)

1. Si la ficha **Programador** no está disponible, muéstrela.
   - Mostrar la ficha **Programador**
     1. Haga clic en la pestaña **Archivo**.
     2. Haga clic en **Opciones** y, a continuación, haga clic en la categoría **Personalizar cinta de opciones**.
     3. En la lista **Fichas principales**, active la casilla de verificación **Programador** y haga clic en **Aceptar**.
2. En la ficha **Programador**, en el grupo **Controles**, haga clic en **Insertar** y, a continuación, en **Controles ActiveX**, haga clic en **Botón de alternancia** .

3. Haga clic en la ubicación de la hoja de cálculo donde desea que aparezca la esquina superior izquierda del botón de alternancia.
4. Para editar el control ActiveX, asegúrese de que está en el modo Diseño. En la ficha **Programador**, en el grupo **Controles**, active el **Modo Diseño** .
5. Para establecer las propiedades del control, en la ficha **Programador**, en el grupo **Controles**, haga clic en **Propiedades** .

   Sugerencia También puede hacer clic con el botón secundario en el control y, a continuación, hacer clic en **Propiedades**.

   Aparece el cuadro de diálogo **Propiedades**. Para obtener información detallada acerca de cada propiedad, seleccione la propiedad y, a continuación, presione F1 para mostrar un tema de la Ayuda de Visual Basic. También puede escribir el nombre de la propiedad en el cuadro **Buscar** de la Ayuda de Visual Basic. En la siguiente sección se resumen las propiedades disponibles.

SI DESEA ESPECIFICAR	USE ESTA PROPIEDAD
General:	
Si el control se carga al abrir el libro. (excepto para controles ActiveX)	**AutoLoad** (Excel)
Si se puede editar el control.	**Locked** (formulario)
El nombre del control.	**Name** (formulario)
La manera en que el control está unido a las celdas que están debajo de él (libre flotante, mover sin cambiar el tamaño o mover y cambiar el tamaño).	**Placement** (Excel)
Si se puede imprimir el control.	**PrintObject** (Excel)
Si el control puede recibir el foco y responder a eventos generados por el usuario.	**Enabled** (formulario)
Si el control está visible u oculto.	**Visible** (formulario)
Texto:	
Texto descriptivo sobre el control que lo identifica o lo describe.	**Caption** (formulario)
La manera en que se alinea el texto en el control (izquierda, centro o derecha).	**TextAlign** (formulario)
Si el contenido del control se ajusta automáticamente al final de una línea.	**WordWrap** (formulario)
Datos y enlace:	
El rango que está vinculado al valor del control.	**LinkedCell** (Excel)
El contenido o estado del control.	**Value** (formulario)
Tamaño y posición:	
Si el tamaño del control se ajusta automáticamente para mostrar todo el contenido.	**AutoSize** (formulario)
El alto o ancho en puntos.	**Height, Width** (formulario)
La distancia entre el control y el borde izquierdo o el superior de la hoja de cálculo.	**Left, Top** (formulario)
Formato:	
El color de fondo.	**BackColor** (formulario)
El estilo de fondo (transparente u opaco).	**BackStyle** (formulario)
El color de primer plano.	**ForeColor** (formulario)

Resumen de las Propiedades por Categorías Funcionales

RESUMEN DE LAS PROPIEDADES POR CATEGORÍAS FUNCIONALES

SI DESEA ESPECIFICAR	USE ESTA PROPIEDAD
General:	
Si el control se carga al abrir el libro. (excepto para controles ActiveX)	**AutoLoad** (Excel)
El nombre del control.	**Name** (formulario)
La manera en que el control está unido a las celdas que están debajo de él (libre flotante, mover sin cambiar el tamaño o mover y cambiar el tamaño).	**Placement** (Excel)
Si se puede editar el control.	**Locked** (formulario)
Si se puede imprimir el control.	**PrintObject** (Excel)
Si el control puede recibir el foco y responder a eventos generados por el usuario.	**Enabled** (formulario)
Si el control está visible u oculto.	**Visible** (formulario)
Texto:	
Texto descriptivo sobre el control que lo identifica o lo describe.	**Caption** (formulario)
Atributos de fuente (negrita, cursiva, tamaño, tachado, subrayado y grosor).	**Bold**, **Italic**, **Size**, **StrikeThrough**, **Underline**, **Weight** (formulario)
La manera en que se alinea el texto en el control (izquierda, centro o derecha).	**TextAlign** (formulario)
La posición del control en relación con el título (izquierda o derecha).	**Alignment** (formulario)
Si el contenido del control se ajusta automáticamente al final de una línea.	**WordWrap** (formulario)
Datos y enlace:	
El contenido o estado del control.	**Value** (formulario)
El rango que está vinculado al valor del control.	**LinkedCell** (Excel)
Tamaño y posición:	
La distancia entre el control y el borde izquierdo o el superior de la hoja de cálculo.	**Left**, **Top** (formulario)
El alto o ancho en puntos.	**Height**, **Width** (formulario)
Si el tamaño del control se ajusta automáticamente para mostrar todo el contenido.	**AutoSize** (formulario)
Formato:	
El color de fondo.	**BackColor** (formulario)
El estilo de fondo (transparente u opaco).	**BackStyle** (formulario)
El color de primer plano.	**ForeColor** (formulario)
La apariencia visual del borde (sin formato, en relieve, de bajo relieve, grabado o rugoso).	**SpecialEffect** (formulario)
Si el control tiene una sombra.	**Shadow** (Excel)
Imagen:	
El mapa de bits que se muestra en el control.	**Picture** (formulario)
La ubicación de la imagen en relación con su título (izquierda, superior, derecha, etc.).	**PicturePosition** (formulario)
Teclado y mouse:	
Un icono del mouse personalizado.	**MouseIcon** (formulario)
La tecla de método abreviado para el control.	**Accelerator** (formulario)
El tipo de puntero que se muestra cuando el usuario sitúa el mouse sobre un objeto determinado (por ejemplo: estándar, flecha o cursor en I).	**MousePointer** (formulario)
Específicas para el botón de opción:	
Grupo de botones de opción mutuamente excluyentes.	**GroupName** (formulario)
Si un usuario puede especificar el estado Null para el control desde la interfaz del usuario.	**TripleState** (formulario)

Nota El tamaño del botón de opción dentro del control y la distancia a la que se encuentra de su texto asociado no se pueden ajustar.

Formas de Optimizar las Macros

Esta información es muy útil para quienes manejen el tema de programación de macros excel. ¿Tus macros van lentas? ¿Problemas a la hora de ejecutarlas? ¿Cuáles son las técnicas recomendadas?

Cuando de programación de macros excel se trata, el tema de la eficiencia y la velocidad es clave. Hay 2 leyes fundamentales que hay que recordar:

- Cuanto menos código tiene una macro mejor…¿por qué?
  Ayuda a que la macro se ejecute mucho más rápido
  Simplifica la tarea a la hora de modificar/ampliar/reparar la macro.

- Cuanto más rápido se ejecuta una macro mejor!…¿por qué?
  Mejora la experiencia del usuario
  No mantiene la PC ocupada tanto tiempo

Formas de Optimizar las Macros

Respecto de usar menos código dependerá de las habilidades del **programador excel** en cuestión. Hemos visto infinidad de casos donde 30 o 40 líneas de código VBA se pueden resumir en 5 o 6 líneas (algo similar pasa con las fórmulas excel). Siempre hay macros o fórmulas que hacen la tarea de forma más directa y sin dar tantas vueltas!

Otra recomendación clave es **invertir mucho tiempo inicial en planificar y analizar la lógica del trabajo.** Esto nos va a ahorrar muchos problemas y dolores de cabeza posteriores!

Hay algunas instrucciones puntuales que siempre conviene usar y que **van a acelerar y optimizar nuestras macros en todos los casos**. Vamos a ver repasar algunas técnicas puntuales que podemos usar al comienzo, durante y al final de nuestras macros.

Formas de Optimizar las Macros

AL COMIENZO DE LAS MACROS

1. Apagar el parpadeo de pantalla
Lo hacemos con la instrucción: *Application.screenupdating=False*
***E*vita los movimientos de pantalla que se producen al seleccionar celdas, hojas y libros.**

2. Apagar los cálculos automáticos
Lo hacemos con la instrucción: *Application.calculation=xlCalculationManual*
Evita que se recalcule todo cada vez que se pegan o modifican datos

3. Apagar los eventos automáticos
Lo hacemos con la instrucción: *Application.EnableEvents=False*
Evita que se disparen macros de evento si las hubiere

Formas de Optimizar las Macros

AL FINAL DE LAS MACROS

4. Apagar visualización de saltos de página
Lo hacemos con la instrucción: *ActiveSheet.DisplayPageBreaks = False*
Sirve para evitar algunos problemas de compatibilidad entre macros Excel 2003 vs. 2007/2010
En resumen, siempre debemos comenzar las macros así:
Application.screenupdating=False
Application.calculation=xlCalculationManual
Application.EnableEvents=False
ActiveSheet.DisplayPageBreaks

5. Borrar contenido de portapapeles
Lo hacemos con la instrucción: *Application.CutCopyMode = False*
Permite limpiar el portapapeles en caso de haber copiado datos
Además debemos volver a su estado original las instrucciones con las que comenzamos la macro.

Formas de Optimizar las Macros

OTRAS TECNICAS UTILES

6. Usar la instrucción WITH

Se usa para evitar tener que referenciar un mismo objeto muchas veces

Ejecución leeenta…

```
Sheets(1).Range("A1:Z1").Font.Italic = True
Sheets(1).Range("A1:Z1").Font.Interior.Color = vbRed
Sheets(1).Range("A1:Z1").MergeCells = True
```

Ejecución rápida!

```
With Sheets(1).Range("A1:Z1")
.Font.Italic = True
.Font.Interior.Color = vbRed
.MergeCells = True
End With
```

Formas de Optimizar las Macros

7. Evitar la instrucción SELECT

Se genera sobre todo en las macros grabadas
La mayoría de las veces no es necesario seleccionar para cumplir el objetivo

Ejecución leeenta…
Range("E1").Select
Selection.Copy
Range("D10").Select
ActiveSheet.Paste

Ejecución rápida!
Range("E1").Copy Range("D10")

Formas de Optimizar las Macros

8. Evitar loops FOR EACH

Tener que ir celda por celda consume mucho tiempo
Se puede resolver el problema de forma más directa!

Ejecución leeenta…

```
For Each cell In Range(“A1:A10000″)
If cell = Empty Then cell = 0
Next cell
```

* Los loops siempre son leeentos
* En este caso recorre 10.000 celdas!

Ejecución rápida!

Existen diversas formas de evitar los loops. La solución dependerá del caso concreto en cuestión. Generalmente se usan algunas de estas técnicas: agrupar, ir a especial, filtros, filtros avanzados. La idea es poder realizar la acción sobre todos los elementos al mismo tiempo, en lugar de tener que ir uno a uno!

Formas de Optimizar las Macros

9. Usar las funciones nativas de Excel

No quieras reinventar la rueda. Quizás ya exista una función Excel que lo haga!

Las macros siempre ejecutan más rápido las funciones nativas de Excel

Ejecución leeenta…
mProducto = 1
For i = 1 to 100
*mProducto = mProducto * Cells(3,i)*
Next
Ejecución rápida!
mProducto = Application.WorkSheetFunction.Product(Range("C1:C100"))

Formas de Optimizar las Macros

10. Forzar la declaración de variables

En el editor VBA, menú Herramientas > Opciones > pestaña Editor > marcar "Requerir declaración de variables"
Luego usar la variable correcta: si es fecha usar Date, si es texto usar String, si es valor usar Long…
Evitar el uso de la variable Variant ya que insume más recursos…
Usar nombres de variables que nos digan algo (por ej. "UltimaFila" o "FilaZ" en lugar de "f" o "uf")

11. Escribir las macros en módulos y no en hojas

Las hojas pueden ser borradas o copiadas y esto generaría problemas inesperados

Formas de Optimizar las Macros

12. Separar el proceso en varias macros (divide y conquistarás)

Si tu macro hace muchas cosas conviene separarla en muchas macros pequeñas y luego unirlas
Es más fácil para controlar, auditar, etc…
Además te permite luego poder rehusar alguna parte del proceso en otras macros

Macro muy laaarga…
Sub MegaMacro()
'Codigo limpia datos
'Codigo carga datos
'Código arregla datos
'Código arma reporte
End Sub()

Formas de Optimizar las Macros

Mejor dividir en diferentes macros para cada proceso

```
Sub LimpiaDatos()
'Codigo…
End Sub Sub

CargaDatos()
'Codigo…
End Sub

Sub ArreglaDatos()
'Codigo…
End Sub

Sub ArmaReporte()
'Codigo…
End Sub
```

Formas de Optimizar las Macros

Finalmente podemos unir todos los procesos
Sub ProcesoCompleto()
Call LimpiaDatos
Call CargaDatos
Call ArreglaDatos
Call ArmaReporte
End Sub()

13. Ser cuidadoso con la instrucción ON ERROR RESUME NEXT

Esta instrucción hace que la macro siga avanzando aunque encuentre un error
En algunos casos esto hará que se ignoren errores que no deberían ser ignorados
Podrías tener errores (bugs) y no enterarte!

14. Comentar bien las macros

¿Qué pasaría si tuvieras que volver a revisar/arreglar/ampliar tu código 8 meses después?
Añadir comentarios te ayudará a describir y recordar la lógica y te ahorrará mucho tiempo!

MSGBox N°2

FUNCIÓN
MSGBOX
Muestra un mensaje en un cuadro de diálogo, espera a que el usuario haga clic en un botón y devuelve un entero que indica el botón utilizado.
Parámetros
Prompt
Obligatorio. Expresión de tipo **String** que se muestra como mensaje en el cuadro de diálogo. La longitud máxima de Prompt es de aproximadamente 1024 caracteres, según el ancho de los caracteres utilizados. Si Prompt consta de más de una línea, puede separar las líneas mediante un carácter de retorno de carro (**Chr(**13**)**), un carácter de avance de línea (**Chr(**10**)**) o una combinación de caracteres de retorno de carro/avance de línea (**Chr(**13**)** y **Chr(**10**)**) entre cada línea.
Buttons
Opcional. Expresión numérica que corresponde a la suma de los valores que especifican el número y tipo de botones que se han de mostrar, el estilo de icono que se va a usar, la identificación del botón predeterminado y la modalidad del cuadro de mensaje. Si se omite Buttons, el valor predeterminado será cero.

MSGBox N°2

Title
Opcional. Expresión de tipo **String** que se muestra en la barra de título del cuadro de diálogo. Si se omite Title, en la barra de título aparecerá el nombre de la aplicación.
Valores
En la siguiente tabla se incluyen los valores de enumeración de **MsgBoxStyle**:

Miembro	Valor	Descripción
OKOnly	0	Muestra sólo el botón Aceptar.
OKCancel	1	Muestra los botones Aceptar y Cancelar.
AbortRetryIgnore	2	Muestra los botones Anular, Reintentar y Omitir.
YesNoCancel	3	Muestra los botones Sí, No y Cancelar.
YesNo	4	Muestra los botones Sí y No.
RetryCancel	5	Muestra los botones Reintentar y Cancelar.

MSGBox N°2

Critical	**16**	**Muestra el icono Mensaje crítico.**
Question	32	Muestra el icono Consulta de advertencia.
Exclamation	48	Muestra el icono Mensaje de advertencia.
Information	64	Muestra el icono Mensaje de información.
DefaultButton1	0	El primer botón es el predeterminado.
DefaultButton2	256	El segundo botón es el predeterminado.
DefaultButton3	512	El tercer botón es el predeterminado.
ApplicationModal	0	Aplicación modal: el usuario debe responder al cuadro de mensaje antes de continuar trabajando en la aplicación actual.
SystemModal	4096	Sistema modal: se suspenden todas las aplicaciones hasta que el usuario responda al cuadro de mensaje.
MsgBoxSetForeground	65536	Especifica la ventana del cuadro de mensaje como ventana de primer plano.
MsgBoxRight	524288	Texto alineado a la derecha.
MsgBoxRtlReading	1048576	Especifica que el texto debe aparecer para ser leído de derecha a izquierda en los sistemas árabe y hebreo.

MSGBox N°2

El primer grupo de valores (0-5) describe el número y tipo de botones mostrados en el cuadro de diálogo. El segundo grupo (16, 32, 48, 64) describe el estilo de icono. El tercer grupo (0, 256, 512) determina qué botón es el valor predeterminado. El cuarto grupo (0, 4096) determina la modalidad del cuadro de mensaje y el quinto grupo especifica si el cuadro de mensaje es la ventana de primer plano, junto con la alineación y la dirección del texto. A la hora de sumar números para crear el valor final del argumento Buttons, se deberá utilizar únicamente un número de cada grupo.
Valor devuelto

Constante	Valor
OK	1
Cancel	2
Abort	3
Retry	4
Ignore	5
Yes	6
No	7

MSGBox N°2

Excepciones

Tipo de excepción	Número de error	Condición
ArgumentException	5	Prompt no es una expresión String o Title no es válido.
InvalidOperationException	5	El proceso no se está ejecutando en modo interactivo de usuario.
InvalidEnumArgumentException	5	Uno o más parámetros no son miembros de la enumeración MsgBoxResult o MsgBoxStyle.

MSGBox N°2

FUNCIÓN
INPUTBOX
Muestra un mensaje en un cuadro de diálogo, espera a que el usuario escriba un texto o haga clic en un botón y devuelve una cadena con el contenido del cuadro de texto.
Parámetros
Prompt
Requerido. Expresión de tipo **String** que se muestra como mensaje en el cuadro de diálogo. La longitud máxima de Prompt es de aproximadamente 1024 caracteres, según el ancho de los caracteres utilizados. Si Prompt incluye más de una línea, puede separar las líneas mediante un carácter de retorno de carro (**Chr(**13**)**), un carácter de salto de línea (**Chr(**10**)**) o una combinación de retorno de carro y salto de línea (**Chr(**13**)** & **Chr(**10**)**) que inserta entre cada línea.
Title
Opcional. Expresión de tipo **String** que se muestra en la barra de título del cuadro de diálogo. Si se omite Title, en la barra de título aparecerá el nombre de la aplicación.
DefaultResponse
Opcional. Expresión de tipo **String** que se muestra en el cuadro de texto como respuesta predeterminada en caso de que no se suministre otra entrada. Si se omite DefaultResponse, el cuadro de texto se mostrará vacío.

MSGBox N°2

XPos

Opcional. Expresión numérica que especifica, en píxeles, la distancia entre el borde izquierdo del cuadro de diálogo y el borde izquierdo de la pantalla. Si omite XPos y YPos, el cuadro de diálogo se centra en la pantalla.

YPos

Opcional. Expresión numérica que especifica, en píxeles, la distancia entre el borde superior del cuadro de diálogo y el borde superior de la pantalla. Si omite XPos y YPos, el cuadro de diálogo se centra en la pantalla.

Ejemplo:

```
Private Sub Command1_Click()
Dim Mensaje, Estilo, Título, Respuesta, MiCadena
Mensaje = "¿Estas totalmente seguro de lo que vas a hacer?"
Estilo = vbYesNo + vbCritical
Título = "error grave (mentira)"
Respuesta = MsgBox(Mensaje, Estilo, Título)
If Respuesta = vbYes Then
                        MiCadena = "Sí"
Else
                        MiCadena = "No"
                        Form1.Visible = False
End If
End Sub
```

Manejo de Variables

TRABAJO CON VARIABLES

Para declarar variables se utiliza normalmente una instrucción **Dim**. La instrucción de declaración puede incluirse en un procedimiento para crear una variable de nivel de procedimiento. O puede colocarse al principio de un módulo, en la sección Declarations, para crear una variable de nivel de módulo.

El siguiente ejemplo crea la variable NombreTexto y específicamente le asigna el tipo de datos String.

Dim NombreTexto As String

Si esta instrucción aparece dentro de un procedimiento, la variable NombreTexto se puede usar sólo en ese procedimiento. Si la instrucción aparece en la sección Declarations del módulo, la variable NombreTexto estará disponible en todos los procedimientos dentro del módulo, pero para los restantes módulos del proyecto. Para hacer que esta variable esté disponible para todos los procedimientos de un proyecto, basta con comenzar la declaración con la instrucción Public, tal y como muestra el siguiente ejemplo:

Public NombreTexto As String

Si desea más información sobre cómo dar nombre a sus variables, puede consultar la sección "Visual Basic Naming Rules" en la Ayuda de Visual Basic.

Manejo de Variables

Las variables se pueden declarar como de uno de los siguientes tipos de datos: Boolean, Byte, Integer, Long, Currency, Single, Double, Date, String (para cadenas de longitud variable), String * longitud (para cadenas de longitud fija), Object, o Variant. Si no se especifica el tipo de datos, el tipo de datos Variant es el predefinido. También es posible crear un tipo definido por el usuario empleando la instrucción Type

Se pueden declarar varias variables en una instrucción. Para especificar el tipo de datos se debe incluir un tipo de datos para cada variable. En la siguiente instrucción se declaran las variables intX, intY, e intZ como del tipo Integer.

Dim intX As Integer, intY As Integer, intZ As Integer

En la siguiente instrucción, intX e intY se declaran como del tipo Variant; y sólo intZ se declara como del tipo Integer.

Dim intX, intY, intZ As Integer

No es necesario especificar el tipo de datos en la instrucción de declaración. Si se omite, la variable será del tipo Variant.

Utilizar la instrucción Public

La instrucción Public se puede utilizar para declarar variables públicas de nivel de módulo.

Manejo de Variables

Public NombreTexto As String
Las variables públicas se pueden usar en cualquier procedimiento del proyecto. Si una variable pública se declara en un módulo estándar o en un módulo de clase, también se podrá usar en los proyectos referenciados por el proyecto en que se declara la variable pública.
Utilizar la instrucción Private
La instrucción Private se puede usar para declarar variables privadas de nivel de módulo
Private MiNombre As String
Las variables Private pueden ser usadas únicamente por procedimientos pertenecientes al mismo módulo.
Nota: Cuando se utiliza a nivel de módulo, la instrucción Dim es equivalente a la instrucción Private. Sería aconsejable usar la instrucción Private para facilitar la lectura y comprensión del código.
Utilizar la instrucción **Static**
Cuando se utiliza la instrucción Static en lugar de la instrucción Dim, la variable declarada mantendrá su valor entre llamadas sucesivas.
Utilizar la instrucción **Option Explicit**

Manejo de Variables

En Visual Basic se puede declarar implícitamente una variable usándola en una instrucción de asignación. Todas las variables que se definen implícitamente son del tipo Variant. Las variables del tipo Variant consumen más recursos de memoria que la mayor parte de las otros tipos de variables. Su aplicación será más eficiente si se declaran explícitamente las variables y se les asigna un tipo de datos específico. Al declararse explícitamente las variables se reduce la posibilidad de errores de nombres y el uso de nombres erróneos
Si no desea que Visual Basic realice declaraciones implícitas, puede incluir en un módulo la instrucción Option Explicit antes de todos los procedimientos. Esta instrucción exige que todas las variables del módulo se declaren explícitamente. Si un módulo incluye la instrucción Option Explicit, se producirá un error en tiempo de compilación cuando Visual Basic encuentre un nombre de variable que no ha sido previamente declarado, o cuyo nombre se ha escrito incorrectamente.
Se puede seleccionar una opción del entorno de programación de Visual Basic para incluir automáticamente la instrucción Option Explicit en todos los nuevos módulos.

Trabajo de Variables

TIPOS DE VARIABLES

Las variables pueden ser de los siguientes tipos: (El número indicado en segundo lugar indica el número de Bytes que ocupa en memoria.)

Booleana (2) Admite los valores 0 y 1, o True (verdadero) y False (falso)
Byte (1) Números enteros, en el rango de 0 a 255
Integer (2) Números enteros en el rango de -32768 a 32767
Long (4) Números enteros en el rango de -2147483648 a 2147483647
Single (4) Punto flotante, simple precisión
Doble (8) Punto flotante, doble precisión.
Currency (8) Entero, con punto decimal fijo (Típico de monedas)
String (*) Cadenas alfanuméricas de longitud variable o fija
Date (8) Fechas
Objet (4) Referencia a objetos
Variant (**) Otros tipos de datos

(*) Una variable tipo String ocupa el mismo número de bytes que caracteres tenga la cadena.

(**) Una variable tipo Variant ocupa 16 bytes si se trata de un número y 22 bytes + longitud de la cadena si se trata de un dato tipo cadena de caracteres.

Estructura Repetitiva

ESTRUCTURA REPETITIVA

Hasta el momento se ha encontrado que cada una de las instrucciones que conforman el algoritmo se ejecuta una, y solo una vez, en el mismo orden en que aparecían. Los algoritmos de este tipo son realmente simples, ya que no incluyen una estructura que permita que un grupo de instrucciones se ejecute varias veces, como resultado de una determinada condición.

La mayoría de los problemas dentro de la programación exigen que un grupo de instrucciones que hacen un cálculo determinado no se haga para un ente específico, sino que sea aplicado a muchos para realizar el mismo cálculo.

La estructura repetitiva, también conocida como estructura MIENTRAS o MIENTRAS – QUE, permite ordenar la realización de una o más instrucciones (secuencia), cero o más veces, con base en el valor de verdad que arroje la evaluación de una expresión de tipo lógico. Esta expresión le permite al algoritmo tomar la decisión de repetir o dejar de ejecutar el grupo de instrucciones.

La estructura está formada por dos partes: la expresión de tipo lógico que es evaluada cada vez que se intenta repetir el proceso y, el grupo de instrucciones donde debe haber, por lo menos, una que permita modificar el resultado de la expresión lógica. De lo contrario, nunca se terminaría la repetición de la ejecución de las instrucciones y sería un proceso infinito.

Estructura Repetitiva

REPRESENTACIÓN:

MIENTRAS <expresión lógica> **HAGA**

<secuencia>

FIN_MIENTRAS

FUNCIONAMIENTO

Cuando se llega a la estructura se evaluar la expresión lógica; sí ésta es:

Falsa, no se ejecuta la secuencia de instrucciones y continuará hacia abajo si hay más instrucciones o, terminará la ejecución si no hay es decir, la secuencia se repetirá cero veces.

Si es verdadera, se ejecuta la secuencia una vez y automáticamente (no hay que decírselo) regresa a evaluar la expresión; si nuevamente es verdadera se repite la ejecución de la secuencia y vuelve a evaluar la expresión. Este proceso es cíclico porque se repite siempre que la condición sea verdadera; también se puede decir que la secuencia no se repite o deja de repetirse cuando la condición es falsa.

VARIABLES DE TIPO CONTADOR.

Mucha veces en los proceso repetitivos es necesario hacer el conteo de sucesos o acciones internas del ciclo; este proceso de conteo se hace con una variable que se va incrementando cada vez que el ciclo se repite.

Estructura Repetitiva

El contador es una variable que se incrementa o disminuye en un valor constante cada que ocurre una acción o suceso. La forma general de los contadores es la siguiente.

CONTADOR = CONTADOR + <valor constante>

VARIABLE DE TIPO ACUMULADOR.

Un acumulador totalizador es una variable cuya misión es almacenar cantidades de variables resultantes de procesos sucesivos. La diferencia con el contador radica en que el incremento o disminución de cada suma es variable en lugar de constante, como en el caso del contador.

La forma general del acumulador es:

ACUMULADOR = ACUMULADOR + <expresión>

VARIABLES TIPO BANDERA O SWITCHE

La bandera es una variable que generalmente usa dos valores excluyentes o diferentes, su contenido es uno cualquiera de dos valores definidos por el programador, el cual ordena cuando cambia su contenido.

La bandera es utilizada dentro de la programación como u seleccionador de una de dos alternativas a seguir dentro del algoritmo. Antes de llegar a la expresión que se utilice para bifurcar la ejecución, la bandera debe tener asignado uno de los dos valores.

Los valores escogidos por la bandera pueden ser de cualquier tipo de dato, por ejemplo:

Estructura Repetitiva

RUPTURA DE CICLOS

Dentro de la programación, algunas veces, es necesario hacer que un ciclo se detenga abruptamente (aborte), porque ya se cumplió algo que se necesitaba o se estaba buscando, por lo que, posiblemente, no se alcance a satisfacer completamente en una forma normal la culminación de la cantidad de veces que debe ejecutarse o repetirse un ciclo. La ruptura se hace cambiando el sentido de la expresión lógica que controla el ciclo, para que esta sea falsa y no se continúe ejecutando la secuencia de instrucciones.

ROMPIMIENTO DE CONTROL DE EJECUCIÓN

Los rompimientos de control en la ejecución de un algoritmo se presentan cuando es necesario detener el proceso porque se ha producido un cambio y se debe dar reportes de lo que ha acontecido antes de la ruptura. En estos casos los registros a procesar están agrupados por lotes; cada uno de ellos pertenece una determinada entidad, persona o cosa, y tiene un campo cuyo contenido cambia cuando se pasa de un lote a otro. Para detectar el cambio de un lote de registros a otro se hace uso de una variable auxiliar, que almacene el valor que tenía el campo antes del cambio de contenido. Al leer un registro y el valor del campo no cambia, se sigue el proceso normal; si su valor es diferente al de la variable auxiliar, se detiene el control de la ejecución para efectuar procesos distintos al de la rutina o secuencia.

TIPO DE DATO	VALORES DE LA BANDERA
Numérico	0 y 1 , 1 y 2
Lógico	V , F
Caracter	S y N , “encontrado” y “no encontrado”

Estructura Repetitiva

ESTRUCTURAS DE LOS CICLOS EN VBA FOR APPLICATION

Bucle **MIENTRAS** se cumple una condición. Ciclo **CUALITATIVO**

ESTRUCTURA

WHILE

<Instrucciones>

WEND

Bucle **PARA** una variable iniciada hasta su finalización. Ciclo **CUANTITATIVO**

ESTRUCTURA

For "linicio" to "Final"

<Instrucciones>

Next

Bucle **MIENTRAS** se cumple una condición, a diferencia del primero este ciclo lleva a que las instrucciones se ejecuten mínimo una vez.

ESTRUCTURA

Do

<Instrucciones>

Loop While "condición"

Instrucción Select

INSTRUCCIÓN SELECT

Ejecuta uno de varios grupos de instrucciones, según el valor de una expresión.

EstructuraSelect [Case] testexpression [Case expressionlist [statements]] [Case Else [elsestatements]]End Select

testexpression

Obligatorio. Expresión. Debe evaluarse en uno de los tipos de datos elementales (**Boolean**, **Byte**, **Char**, **Date**, **Double**, **Decimal**, **Integer**, **Long**, **Object**, **SByte**, **Short**, **Single**, **String**, **UInteger**, **ULong** y **UShort**).

expressionlist

Requerido en una instrucción **Case**. Lista de cláusulas de expresiones que representan valores que coinciden para testexpression. Las cláusulas de varias expresiones se separan mediante comas. Cada cláusula puede tomar una de las siguientes formas:

- *expression1* **To** *expression2*
- [**Is**] *comparisonoperator expression*
- *expression*

Utilice la palabra clave **To** para especificar los límites de un intervalo de valores que coinciden para testexpression. El valor de expression1 debe ser menor o igual que el valor de expression2.

Utilice la palabra clave **Is** con un operador de comparación (**=**, **<>**, **<**, **<=**, **>** o **>=**) para especificar una restricción de los valores coincidentes para testexpression. Si no se escribe, la palabra clave **Is** se insertará automáticamente antes de *comparisonoperator*.

La forma de especificar sólo expression se trata como un caso especial de la forma **Is** donde *comparisonoperator* es el signo igual (**=**). Esta forma se evalúa como testexpression = expression.

Instrucción Select

Las expresiones contenidas en expressionlist pueden ser de cualquier tipo de datos, siempre que sean implícitamente convertibles al tipo de testexpression y el correspondiente comparisonoperator sea válido para los dos tipos con los que se utilice.

statements

Opcional. Una o más instrucciones posteriores a **Case** que se ejecutan si testexpression coincide con cualquier cláusula de expressionlist.

elsestatements

Opcional. Una o más instrucciones posteriores a **Case Else** que se ejecutan si testexpression no coincide con ninguna cláusula de expressionlist de cualquiera de las instrucciones **Case**.

End Select

Comentarios

Si testexpression coincide con cualquier cláusula de **Case** expressionlist, se ejecutan las instrucciones situadas a continuación de **Case** hasta la siguiente instrucción **Case**, **Case Else** o **End Select**. El control pasa después a la instrucción que sigue a **End Select**. Si testexpression coincide con una cláusula expressionlist en más de una cláusula **Case**, sólo se ejecutan las instrucciones situadas después de la primera coincidencia.

La instrucción **Case Else** se utiliza para introducir las elsestatements que se deben ejecutar si no se encuentra ninguna coincidencia entre las cláusulas testexpression y expressionlist de cualquiera de las demás instrucciones **Case**. Aunque no es necesario, es aconsejable tener una instrucción **Case Else** en su construcción **Select Case** para controlar los valores testexpression imprevistos. Si no coincide ninguna cláusula **Case** expressionlist con testexpression y no hay ninguna instrucción **Case Else**, el control pasa a la instrucción **End Select** siguiente.

Instrucción Select

Se pueden utilizar varias expresiones o intervalos en cada cláusula **Case**. Por ejemplo, la línea siguiente es válida.

```
Case 1 To 4, 7 To 9, 11, 13, Is > maxNumber
```

```
Dim number As Integer = 8
Select Case number
    Case 1 To 5
        Debug.WriteLine("Between 1 and 5, inclusive")
        ' The following is the only Case clause that evaluates to True.
    Case 6, 7, 8
        Debug.WriteLine("Between 6 and 8, inclusive")
    Case 9 To 10
        Debug.WriteLine("Equal to 9 or 10")
    Case Else
        Debug.WriteLine("Not between 1 and 10, inclusive")
End Select
```

Funciones en VBA por App

FUNCIONES EN VBA FOR APPLICATIONS

Si utiliza con frecuencia un cálculo complejo en Excel, no es necesario que escriba una fórmula larga y complicada repetida veces. Puede crear su propia función de hoja de cálculo para realizar los cálculos. Después, puede utilizar esa función para crear fórmulas más sencillas de introducir y mantener.
Para crear sus propias funciones personalizadas, debe trabajar con Microsoft Visual Basic® para Aplicaciones (VBA).
Las funciones en el entorno de Visual Basic son como los Procedimientos o subrutinas, es decir como el click de un botón o el doble click en un formulario, pero con la diferencia de que estas, devuelven un resultado y los procedimientos NO, el click de un botón puede mostrar el resultado de una cuenta en la pantalla, pero no puede usarse el procedimiento para agregarle (por ejemplo) un 2 y multiplicarlo por un 3, los botones y formularios y todos los Objetos NO PUEDEN OPERARSE, es decir no se los puede restar sumar, multiplicar o dividir o lo que se les ocurra.
Para salvar esta imposibilidad, podemos crear nosotros mismos una función.
Una función se escribe de una forma muy parecida a una Subrutina o procedimiento pero cambia la ***Sintaxis*** o ***Declaración***

Funciones en VBA por App

El código para una UDF se deben colocar en un módulo de código estándar, no es uno de los módulos de hoja y no en el módulo ThisWorkbook.. Un módulo puede contener cualquier número de funciones.
Estructura
Function name [(Of typeparamlist)] [(parameterlist)] [As returntype]
End Function
NAME
Nombre del procedimiento. Obligatorio.
REGLAS
Un nombre de elemento en Visual Basic debe observar las reglas siguientes:
•Debe comenzar por un carácter alfabético o un signo de subrayado (_).
•Sólo puede contener caracteres alfabéticos, dígitos decimales y signos de subrayado.
•Debe contener por lo menos un carácter alfabético o un dígito decimal, si empieza con un signo de subrayado.
•No puede superar los 1023 caracteres de longitud.
El límite de longitud de 1023 caracteres también se aplica a la cadena completa de un nombre completo.

```
aB123__45
_567
```

El ejemplo siguiente muestra algunos nombres de elementos no válidos. El primero contiene sólo un subrayado, el segundo comienza con un dígito decimal y el tercero contiene un carácter no válido ($).

```
' Three INVALID element names

_
12ABC
xyz$wv
```

Returntype
Obligatorio:Tipo de datos del valor devuelto por este procedimiento.

Funciones en VBA por App

COMENTARIOS

El código ejecutable debe estar en un procedimiento. Cada procedimiento se declara a su vez dentro de una clase, estructura o módulo, que se denominan clase *contenedora*, estructura o módulo.

Utilice un procedimiento **Function** cuando necesita devolver un valor al código de llamada.

Utilice un procedimiento **Sub** cuando no necesita devolver un valor.

Sólo puede utilizar **Function** en el nivel de módulo. Esto significa que el *contexto de la declaración* para una función debe ser una clase, estructura, módulo o interfaz y no puede ser un archivo de código fuente, espacio de nombres, procedimiento o bloque.

Se puede utilizar un procedimiento **Function** en el lado derecho de una expresión cuando desee utilizar el valor devuelto por la función. El procedimiento **Function** se utiliza de la misma manera que utiliza cualquier función de la biblioteca como por ejemplo **Sqrt**, **Cos** o **ChrW**.

Puede llamar a un procedimiento **Function** con el nombre del procedimiento, seguido por la lista de argumentos entre paréntesis, en una expresión. Si no se proporcionan argumentos, se pueden omitir los paréntesis. Sin embargo, el código es más legible si se incluyen siempre los paréntesis.

Se puede llamar una función también mediante la instrucción **Call**. En cualquier caso, el valor devuelto se pasará por alto.

Funciones en VBA por App

Reglas

Tipo de valor devuelto. La instrucción **Function** puede declarar el tipo de datos del valor que devuelve. Puede especificar cualquier tipo de datos o el nombre de una enumeración, estructura, clase o interfaz.

Si no especifica returntype, el procedimiento devuelve un tipo **Object**.

Implementación. Si este procedimiento utiliza la palabra clave **Implements**, la clase contenedora o la estructura también deben incluir una instrucción **Implements** inmediatamente después de la instrucción **Class** o **Structure**. La instrucción **Implements** debe incluir cada interfaz especificada en implementslist. Sin embargo, el nombre con el que una interfaz define **Function** (en definedname) no tiene que ser el mismo que el nombre de este procedimiento (en name).

Comportamiento

Volver de un procedimiento. Cuando el procedimiento **Function** vuelva al código de llamada, la ejecución continúa con la instrucción que sigue a la instrucción que lo llamó. Las instrucciones **Exit Function** y **Return** provocan una salida inmediata de un procedimiento **Function**. Puede aparecer cualquier número de instrucciones **Exit Function** y **Return** en cualquier parte del procedimiento y puede combinar instrucciones **Exit Function** y **Return**.

Funciones en VBA por App

•**Valor devuelto.** Para devolver un valor de una función, se puede asignar el valor al nombre de función o incluirlo en una instrucción **Return**. El ejemplo siguiente asigna el valor devuelto al nombre de función myFunction y, a continuación, utiliza la instrucción **Exit Function** para volver:

```
Function myFunction(ByVal j As Integer) As Double
    myFunction = 3.87 * j
    Exit Function
End Function
```

Si utiliza **Exit Function** sin asignar un valor a name, el procedimiento devuelve el valor predeterminado del tipo de datos especificado en returntype. Si no se especifica returntype, el procedimiento devuelve **Nothing**, el valor predeterminado de **Object**.

La instrucción **Return** asigna el valor devuelto y sale de la función. Esto se muestra en el siguiente ejemplo.

```
Function myFunction(ByVal j As Integer) As Double
    Return 3.87 * j
End Function
```

Arrays en VBA por App

Arreglos
vectores

Los Arreglos se utilizan para almacenar un conjunto de variables, que sean del mismo tipo de dato, y todas estas bajo un mismo nombre.

Por ejemplo imaginemos que tenemos 20 variables de tipo String que almacenan nombres (nombre1, nombre2, etc..). si yo ahora quisiera pasar todas estas cadenas a minúsculas tendría que utilizar la función Lcase con cada variable: nombre1 = Lcase(nombre1), nombre2 = Lcase(nombre2), etc..

Importante: los arreglos se dividen en 2 grupos, los **vectores** y las **matrices**. Los vectores son arreglos que contienen **una sola dimensión** y las **matrices 2 o mas dimensiones**.

Ejercicio práctico.

Arrays en VBA por App

Utilizar una estructura Type o UDT en un arreglo

Como vimos, en un arreglo podemos almacenar datos de cualquier tipo pero no mezclarlos, es decir podemos crear arreglos de tipo string, de tipo Integer etc,,, pero sin duda que lo mas importante y de mas utilidad a la hora de programar es la utilización de datos definidos por nosotros mediante una estructura Type

Por ejemplo:

```
Option Explicit

' Estructura de dato para el vector
Private Type agenda

    nombre As String
    apellido As String
    cpostal As Integer

End Type

' Declaramos el vector
Dim personas(1 To 3) As agenda

Private Sub Form_Load()

    ' Llenamos con datos para el elemento 1 del arreglo

    personas(1).nombre = "carlos"
    personas(1).apellido = "Martínez"
```

Arrays en VBA por App

```
personas(1).cpostal = 1900

    ' Llenamos con datos para el elemento 2 del arreglo

    personas(2).nombre = "Héctor"
    personas(2).apellido = "rosales"
    personas(2).cpostal = 1898

    ' Llenamos con datos para el elemento 3 del arreglo

    personas(3).nombre = "Albert"
    personas(3).apellido = "Einstein"
    personas(3).cpostal = 1324

End Sub
```

Para utilizar una estructura definida por nosotros en vectores o matrices, se hace de la forma habitual, con la diferencia que debemos declarar el arreglo utilizando el tipo de dato Type que hayamos creado, en este caso Dim personas(1 to 3) as agenda

Arrays en VBA por App

Las matrices

las matrices son arreglos de mas de 1 dimensión (2 o mas), a diferencia de los vectores que poseen una sola dimensión.
Podemos imaginar una matriz bidimensional (2 dimensiones) , como una cuadrícula con filas y columnas, donde las filas representarían las coordenadas x y las columnas las coordenadas y.
A una matriz de 3 dimensiones o tridimensional se la puede imaginar con las coordenadas x, y, z, y esta es ideal para representar figuras en el espacio por ejemplo.
Las matrices se declaran en el código de manera igual que los vectores, con la diferencia que debemos **indicar mas subíndices de acuerdo a la cantidad de dimensiones que posea l**

la matriz.

por lo general no se suele utilizar matrices de mas de 3 dimensiones..
Ejemplo de matriz de 2 dimensiones
Matriz bidimensionales de 6 x 8 (de 2 dimensiones).
Dim personas (1 to 6, 1 to 8) as string
Si luego quisiera acceder a los datos de la misma basta con referirnos a los subíndices

Arrays en VBA por App

Por ejemplo:
personas (1, 1) = "Natalia"
personas (2, 1) = "pedro"
personas (1, 7) = "valeria"
personas (1, 8) = "josé"
personas (2, 2) = "carolina"
personas (4, 1) = "raquel"
personas (6, 2) = "eustaquio"
personas (6, 5) = "maria"
personas (6, 8) = "mariana"
El total de índices posibles para almacenar datos o valores en el ejemplo anterior es de 48 datos, ya que si multiplicamos 6 x 8 nos da como total 48 valores posibles para utilizar en la matriz bidimensional.
En este ejemplo creamos una matriz de 3 dimensiones de 3 x 3 x 3
Dim cubo (1 to 3, 1 to 3, 1 to 3) as integer
para acceder a los datos sería exactamente de la misma manera pero debemos utilizar un índice mas.

Arrays en VBA por App

Ejemplo:
cubo (1, 1 , 1) = 50
cubo (1, 1 , 2) = 50
cubo (1, 1 , 3) = 50
cubo (1, 2 , 1) = 50
cubo (1, 2 , 2) = 50
cubo (1, 2 , 3) = 50
cubo (1, 3 , 1) = 50
cubo (1, 3 , 2) = 50
cubo (1, 3 , 3) = 50
cubo (2, 1 , 1) = 50
cubo (2, 1 , 2) = 50
cubo (2, 1 , 3) = 50
cubo (2, 2 , 1) = 50
cubo (2, 2 , 2) = 50
cubo (2, 2 , 3) = 50
cubo (2, 3 , 1) = 50
cubo (2, 3 , 2) = 50
cubo (2, 3 , 3) = 50
cubo (3, 1 , 1) = 50
cubo (3, 1 , 2) = 50
cubo (3, 1 , 3) = 50
cubo (3, 2 , 1) = 50
cubo (3, 2 , 2) = 50
cubo (3, 2 , 3) = 50
cubo (3, 3 , 1) = 50
cubo (3, 3 , 2) = 50
cubo (3, 3 , 3) = 50
En el ejemplo anterior, que es un poco extenso, es para que veas todos los posibles valores que puedes almacenar en una matriz de 3 x 3 x 3, y que da como resultado un arreglo de 27 valores posibles.

Códigos VBA para APP

TRABAJANDO CON LIBROS

ABRIR UN SEGUNDO LIBRO:

Application.Workbooks.Open "C:\Mis docu\Libro1.xls" o Workbooks.Open "C:\....."

ACTIVAR UN SEGUNDO LIBRO:

Workbooks("Libro2.xls").Worksheets("Hoja3").Activate o Workbooks(2).Sheets(3).Activate

CERRAR UN LIBRO (SIN GUARDAR):

Workbooks("Libro1.xls").Close False o ActiveWorkbook.Close False

CERRAR UN LIBRO (GUARDANDO LOS CAMBIOS):

ActiveWorkbook.Save

ActiveWorkbook.Close

GUARDAR UN LIBRO CON OTRO NOMBRE:

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

Estas son algunas de las opciones. Si se omiten, escribir la coma.

Ejemplo:

ActiveWorkbook.SaveAs Filename:="C:\Mis doc\Libro.xls",,, ReadOnlyRecommended:=False

Códigos VBA para APP

GUARDAR UN LIBRO CUYO NOMBRE SERÁ EL VALOR DE UNA VARIABLE:

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

PARA NO MOSTRAR AVISO AL SALIR, O AL ELIMINAR UNA HOJA, O CUALQUIER AVISO QUE QUEREMOS OBVIAR:

```
Application.DisplayAlerts= False          'volverla a True al finalizar la macro
```

DESHABILITAR LA OPCIÓN DE ACTUALIZAR VÍNCULOS AL ABRIR UN LIBRO:

```
Application.DisplayAlerts= False          'volverla a True al finalizar la macro
WorkBooks.Open Filename:= "C:\Mis docu\pruebas.xls", UpdateLinks:= 0
```

PARA NO MOSTRAR LA EJECUCIÓN DE LA MACRO:

```
Application.ScreenUpdating = False        'volverla a True al finalizar la macro
```

Códigos VBA para APP

EJECUTAR UNA MACRO AL ABRIR UN LIBRO:

Crear una rutina pública o insertar una rutina en un módulo:

Ejemplo:

```
Sub Nuevamacro ()
'instrucciones
End Sub
```

Y en el evento Open del objeto ThisWorkbook:

```
Private Sub Workbook_Open ()
        Nuevamacro
End Sub
```

Macro que permite abrir un libro de excel llamado "Libro 1" ubicado en la misma carpeta del libro donde se está ejecutando la macro.

```
Sub abrir_libro_en_la_misma_carpeta()

Workbooks.Open ThisWorkbook.Path & "\Libro1.xlsx"

End Sub
```

Códigos VBA para APP

Macro que permite abrir un libro de excel almacenado en una variable

```
Sub abrir_libro_variable()

'Colocamos el nombre del libro en un inputbox

AbrirLibro = InputBox("Teclea el libro que desea abrir", "Libro para abrir")

'Abrimos el libro con extensión xls

Workbooks.Open ThisWorkbook.Path & "\" & "" & AbrirLibro

End Sub
```

TRABAJANDO CON HOJAS
ACTIVAR O SELECCIONAR OTRAS HOJAS:

```
Sheets("Hoja2").Activate    o   Sheets(2).Select
```

SELECCIONAR LA HOJA ANTERIOR O SIGUIENTE:

```
ActiveSheet.Previous.Select          'hoja anterior a la activa
ActiveSheet.Next.Select              'hoja posterior a la activa
```

Códigos VBA para APP

DATOS DE LA HOJA:

```
ActiveSheet.Name              'nombre de la hoja
ActiveSheet.Index             'número de hoja
```

COPIAR DATOS DE UNA HOJA A OTRA:

```
Selection.Copy                'previamente se habrá seleccionado algo
ActiveSheet.Paste Destination:=ActiveSheet.Next.Cells(1,4)
Application.CutCopyMode= False
```

OCULTAR FILAS O COLUMNAS:

```
ActiveCell.EntireRow.Hidden=True    o    ActiveCell.EntireColumn.Hidden= True
```

PROTEGER O DESPROTEGER UNA HOJA:

```
ActiveSheet.Protect "contraseña"          'proteger con contraseña
ActiveSheet.Unprotect  "contraseña"       'quitar la protección
```

INSERTAR FILAS O COLUMNAS:

```
Workbooks("Libro1").Sheets("Hoja2").Column(i).Select
Selection.EntireColumn.Insert
        'reemplazar Column por Row en caso de filas.
```

Códigos VBA para APP

ELIMINAR FILAS O COLUMNAS:

```
ActiveSheet.Row(n).Select
Selection.EntireRow.Delete
```

INSERTAR UNA IMAGEN EN UNA HOJA:

```
ActiveSheet.Pictures.Insert(ruta).Select    'la ruta entre comillas:  "C:\Mis docu\Foto1.jpg"
```

IMPRIMIR LA HOJA SELECCIONADA:

```
ActiveWindow.SelectedSheet.PrintOut Copies:=1, Collate:=True
```

TRABAJANDO CON CELDAS

FORMAS DE SELECCIONAR UNA CELDA O UN RANGO DE CELDAS:

```
Range("B7").Select                              'selecciona la celda B7
Range("B:B").Select                             'selecciona toda la columna B
Range("A4:A10, D10, B5:B20").Select       'selecciona rangos discontinuos
Range("A"&variable).Select          'selecciona la celda cuya fila será el valor de la variable
Si Rango=("B2"&":E"&variable)
entonces:  Range(rango).Select  'selecciona el rango B2:E hasta la fila indicada en la variable
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 4 hasta la 7.
```

Códigos VBA para APP

SELECCIONAR CELDAS A CIERTA DISTANCIA DE LA CELDA ACTIVA:

```
Sheets(1).Range("A1").Offset(2,3).Select    'selecciona la celda D3
ActiveCell.Offset(-10,1).Select      'selecciona la celda que se encuentra 10 filas por
encima
                                     'y 1 columna a la derecha de la celda activa.
```

FORMATO DE CELDAS:

```
Range("B2:D10").Select
With Selection
    .Font.Bold=True                              'formato negrita
    .Font.Italic=True                            'formato cursiva
    .Font.Underline=xlUnderlineStyleSingle       'subrayado simple
    .Font.Color = RGB(255,0,0)                   'color de fuente (para estos valores será rojo)
    .HorizontalAlignment=xlCenter                'alineación central (Right=derecha,
Left=izquierda)
End With
```

GUARDAR LA DIRECCIÓN DE UNA CELDA EN UNA VARIABLE:

```
lugar= ActiveCell.Address                        'guarda la referencia absoluta
lugar= ActiveCell.Address(False, False)          'guarda la referencia relativa
```

Códigos VBA para APP

COPIAR UN COMENTARIO EN OTRA CELDA:

```
ActiveCell.Offset(0,1).Value = ActiveCell.Comment.Text
'copia el comentario de la celda activa en la celda que se encuentra en la
                                                    'columna siguiente.
```

SELECCIONAR CELDAS Y BORRARLAS:

```
Range(rango). Select      o    Cells.Select
Selection.ClearContents
```

AMPLIAR UN RANGO SELECCIONADO

```
Selection.Resize(10,4).Select     ' (10 filas, 4 columnas) ' si el rango seleccionado fue A1:B5 ahora será:
A1:D10
```

COMBINAR CELDAS SELECCIONADAS:

```
Range("B1:E1").Select
Selection.Merge
```

SELECCIONAR EL RANGO DONDE SE ENCUENRA LA CELDA ACTIVA:

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

SELECCIONAR HASTA LA ÚLTIMA CELDA NO VACÍA:

```
Range("A2", Range("A2").End(xlDown)).Select       'seleciona desde A2 hacia abajo
Range("A2", Range("A2").End(xlToRight)).Select    'selecciona desde A2 hacia la derecha
Range("D2", Range("D2").End(xlToLeft)).Select     'selecciona desde D2 hacia la izquierda
Range("A20", Range("A20").End(xlUp)).Select       'selecciona desde A20 hacia arriba
```

En cambio, para seleccionar solo la última celda con datos será:

```
Range("A2").End(xlDown).Select
```

Códigos VBA para APP

TRABAJANDO CON UNA COLECCIÓN

EJEMPLO1: INTRODUCIR UN NOMBRE PARA CADA HOJA DEL LIBRO ACTIVO:

```
Dim MiNombre as String
Dim hoja as Worksheet
For Each hoja in Worksheets
   MiNombre = InputBox("Ingrese nombre de hoja: ")
next hoja
```

EJEMPLO2: INTRODUCIR VALORES PARA CADA CELDA DE UN RANGO

```
Dim celdita as Range
For Each celdita in ActiveSheet.Range("A1:B10")
   celdita.Value = InputBox("Ingrese valor: ")
next celdita
```

EJEMPLO3: INTRODUCIR LOS MISMOS VALORES EN CELDAS DE TODAS LAS HOJAS

```
Dim hoja as Sheets
For Each hoja in Sheets
   hoja.Range("E3").Value = Date
   hoja.Range("F3").Value = Time
next hoja
```

Códigos VBA para APP

TRABAJANDO CON OBJETOS
LLAMANDO A UN USERFORM, DESDE UN BOTÓN:
En una hoja de Excel, los botones que lanzan una acción, pueden ser colocados con la barra de Formularios o Cuadro de Controles.
Botón de formulario: se asigna una macro que previamente se escribió en un módulo en el Editor de Visual Basic. Ejemplo

```
Sub mostrando ()
UserForm1.Show        'nombre del Userform que se desea mostrar en la hoja
End Sub
```

Ejemplo:

```
Private Sub CommandButton1_Click()
UserForm1.Show
End Sub
```

VOLCAR DATOS DE UN USERFORM A LA HOJA:

```
Cells(fila, col).Value = TextBox1.Value

Sheets("Hoja1").Cells(fila, col).Offset(1, 0).Value = TextBox2.Value
```

CARGAR DATOS A UN LISTBOX DE UN USERFORM:

```
Private Sub UserForm_Activate()
Dim item As Variant
For Each item In Range("F1:F6")
ListBox1.AddItem item.Value
Next item
End Sub
```