FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

CURSOS INSTITUCIONALES

PROGRAMACIÓN DE MACROS EN EXCEL

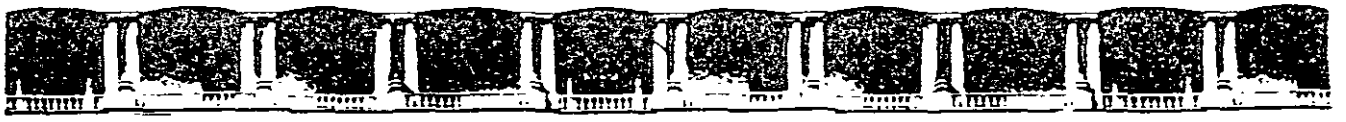
APUNTES GENERALES

CI - 244

Instructor: Ing. Rodolfo González Maldonado
LUZ Y FUERZA DEL CENTRO

NOVIEMBRE DE 2005

Palacio de Minería, Calle de Tacuba No. 5, Primer piso, Delegación Cuauhtémoc, CP 06000, Centro Histórico, México D.F.,
APDO Postal M-2285 ■ Tels: 5521.4021 al 24, 5623.2910 y 5623.2971 ■ Fax: 5510.0573



FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

CURSOS INSTITUCIONALES

PROGRAMACIÓN DE MACROS EN EXCEL

APUNTES GENERALES

Ci - 244

Instructor: Ing. Rodolfo González Maldonado
LUZ Y FUERZA DEL CENTRO

NOVIEMBRE DE 2005

Palacio de Minería, Calle de Tacuba No 5, Primer piso, Delegación Cuauhtémoc, CP 06000, Centro Histórico, México D.F.,
APDO Postal M-2285 • Tels: 5521.4021 al 24, 5623.2910 y 5623 2971 • Fax: 5510.0573

INDICE

INTRODUCCIÓN

MACRO

Una macro consiste en una serie de comandos y funciones que se almacenan en un módulo de Visual Basic y que está disponible siempre que sea necesario ejecutar la tarea. Una macro se graba igual que se graba música en un casete; a continuación, se ejecuta la macro para que repita los comandos.

Antes de grabar o escribir una macro, planifique los pasos y los comandos que desea que ejecute la macro. Si se comete algún error mientras se graba la macro, también se grabarán las correcciones que se realicen. Cada vez que se grabe una macro, ésta se almacenará en un nuevo módulo adjunto a un libro.

Con el Editor de Visual Basic, se pueden modificar macros, copiar macros de un módulo en otro, copiar macros entre diferentes libros, cambiar de nombre a los módulos que almacenan las macros o cambiar de nombre a las macros.

FORMAS DE HACER UNA MACRO EN EXCEL

Se puede hacer una Macro de dos maneras:

Grabación de Macros

Se refiere a la grabación de todos los pasos que se van realizando, al mismo tiempo que Excel genera código de los pasos ejecutados.

Creación de Macros

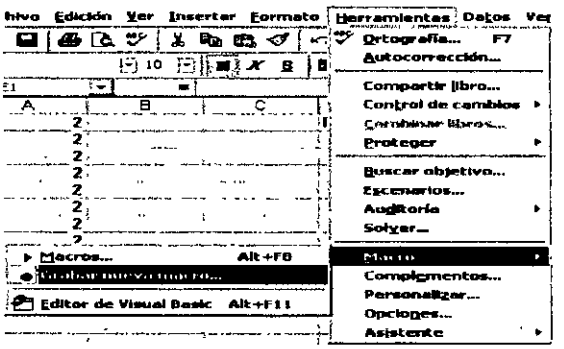
Se refiere a la codificación de una macro, se escribe el programa mediante el uso del editor de Visual Basic.

NOTAS:

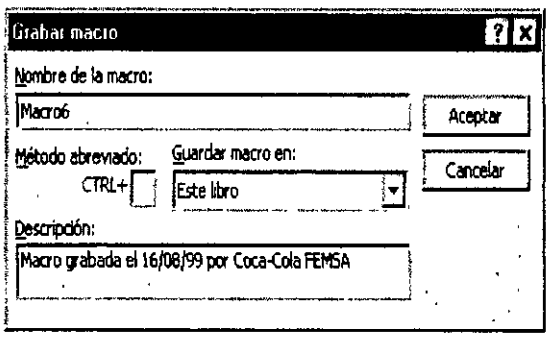
CAPITULO 1. MACROS DE GRABACIÓN

GRABAR UNA MACRO

1. Seleccione Macro en el menú Herramientas y, a continuación, haga clic en Grabar.



2. En el cuadro Nombre de la macro, escriba un nombre para la macro.



3. El primer carácter del nombre de la macro debe ser una letra. Los demás caracteres pueden ser letras, números o caracteres de subrayado. No se permiten espacios en un nombre de macro; puede utilizarse un carácter de subrayado como separador de palabras. Para ejecutar la macro presionando un método abreviado, escriba una letra en el cuadro Tecla de método abreviado. Puede utilizarse CONTROL+ letra (para letras *minúsculas*) o CONTROL+ MAYÚS + *letra (para* letras mayúsculas), donde letra es cualquier tecla del teclado. La tecla de método abreviado que se utilice no puede ser ni un número ni un carácter especial. La tecla de método abreviado suplantará a cualquier tecla de método abreviado predeterminada en Microsoft Excel mientras esté abierto el libro que contiene la macro.

4. En el cuadro Guardar macro en, haga clic en la ubicación en que desea almacenar la macro. Si desea que la macro esté disponible siempre que se utilice Microsoft Excel, almacene la

NOTAS:

macro en el libro de macros personales en la carpeta INICIAR. Para incluir una descripción de la macro, escriba la descripción en el cuadro Descripción.

5. Haga clic en Aceptar.

Referencias Absolutas y Relativas

Si se seleccionan celdas mientras se está ejecutando una macro, ésta seleccionará las mismas celdas independientemente de la celda que se haya seleccionado en primer lugar, ya que graba referencias absolutas de celda. Si desea tener una macro para seleccionar celdas independientemente de la posición que tenga la celda activa cuando se ejecute la macro, configure el grabador de macros para que grabe referencias relativas de celda. En la barra de herramientas Detener grabación, haga clic en Referencia 

Microsoft Excel continuará grabando macros con referencias relativas hasta que termine la sesión con Microsoft Excel o hasta que haga clic otra vez en Referencias relativas  .

6. Ejecute las acciones que desee grabar.

7. En la barra de herramientas Detener grabación, haga clic en Detener grabación  .

 Si desea tener una macro para seleccionar una celda determinada, ejecute una acción y seleccione otra celda relativa a la celda activa; cuando se grabe una macro, pueden mezclarse referencias relativas y absolutas. Para grabar una macro utilizando referencias, compruebe que está activada la opción Referencias relativas . Para grabar una macro utilizando referencias absolutas (cuando se hace referencia a una posición fija, sin importar la posición de la celda activa), compruebe que la opción Referencias relativas no está habilitada.

NOTAS:

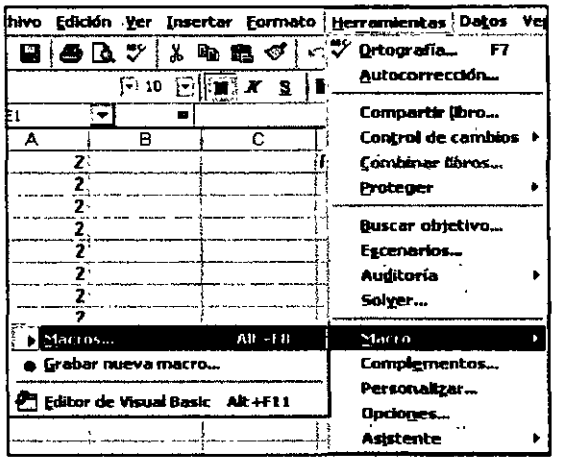
EJECUCIÓN DE UNA MACRO

Una vez grabada, una macro puede ejecutarse en Microsoft Excel o en el Editor de Visual Basic. Normalmente, se ejecutará la macro en Microsoft Excel; sin embargo, puede ejecutarse desde el Editor de Visual Basic, mientras se realiza la macro. Para interrumpir la macro antes de que finalice las acciones que se han grabado, presione ESC.

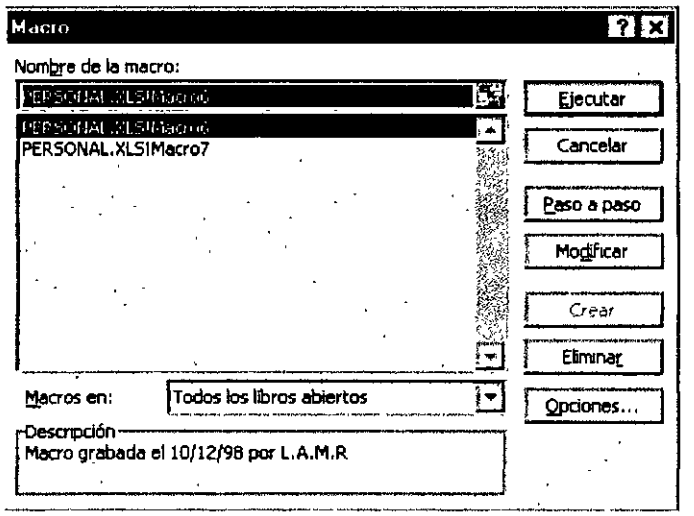
1. Abra el libro que contiene la macro.

2. Seleccione Macro en el *Menú* Herramientas y, a continuación, haga clic en Macros.

3. En el cuadro Nombre de la macro, escriba el nombre de la macro que desea ejecutar.



8. Haga clic en Ejecutar.



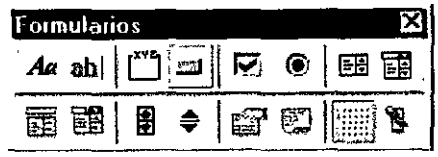
Nota: Para interrumpir una macro antes de que finalice las acciones, presione ESC.

NOTAS:

ASIGNAR MACRO A UN BOTÓN DE CONTROL

Si desea ejecutar una macro existente desde un control de la hoja de cálculo, asegúrese de que el libro que contenga la macro está abierto.

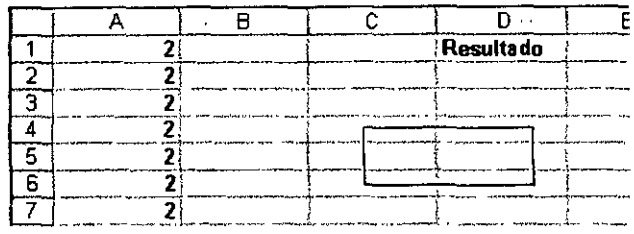
1. Abra la hoja de cálculo a la que desee agregar controles.

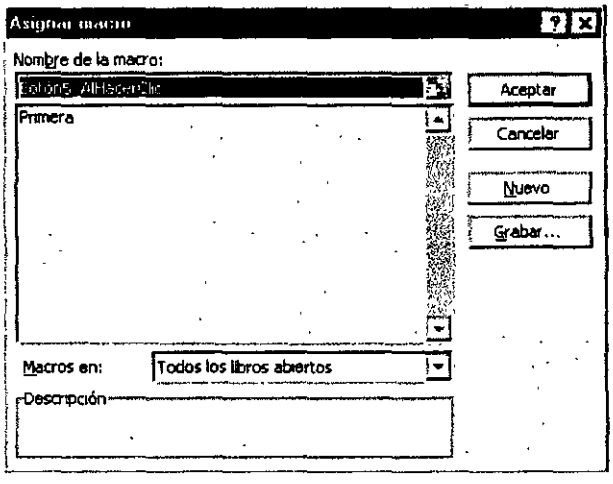


Asegúrese de que aparece la barra de herramientas Formularios.

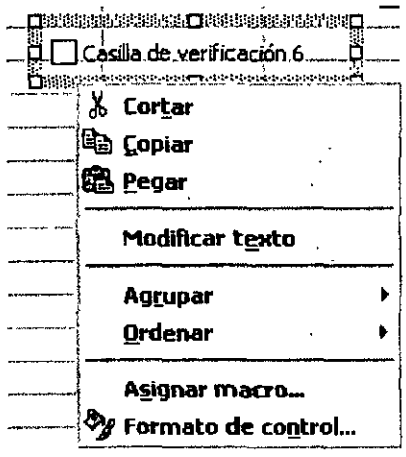
2. En la barra de herramientas Formularios, haga clic en el botón del control que desee agregar.

3. En la hoja de cálculo, arrastre el control hasta que tenga el tamaño que desee.





4. Si agrega un botón, seleccione la macro que desee ejecutar cuando se haga clic en el botón del cuadro Nombres de macro.



Si agrega un control que no sea un botón, haga clic con el botón secundario en el control y, a continuación, haga clic en Asignar macro en el *Menú* contextual.

NOTAS:

NOTAS:

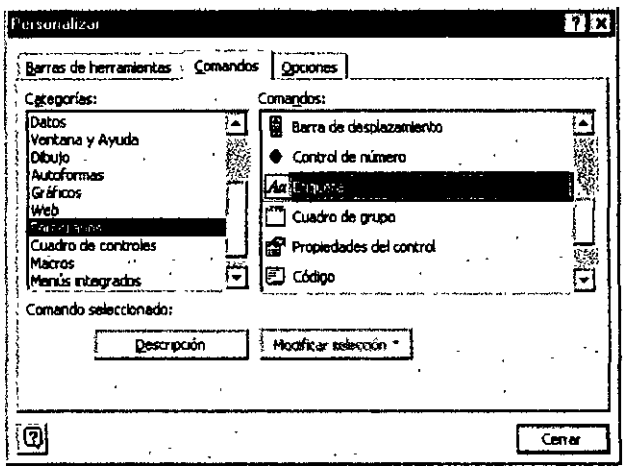
ASIGNAR MACRO A UN *MENÚ*

Pasos a seguir:

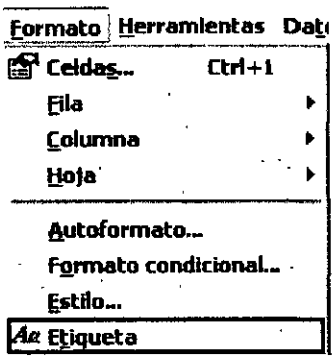
Si desea ejecutar una macro existente desde un *Menú* de la hoja de cálculo, asegúrese de que el libro que contenga la macro está abierto.

Seleccione en el *Menú* Herramientas la opción personalizar

1. En la ficha Comandos, seleccione la categoría de formularios y el comando Etiqueta.

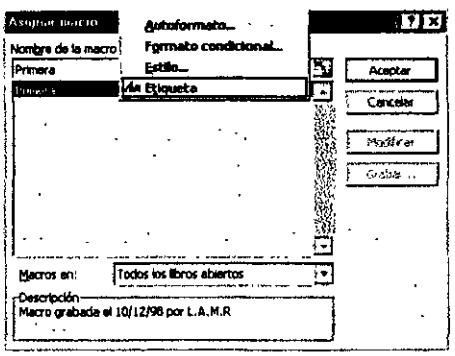


2. Arrastre el comando Etiqueta al *Menú* donde desea agregar la macro.



3. De un clic con el botón derecho del mouse sobre la etiqueta que acaba de agregar y seleccione el comando Asignar macro del *Menú* contextual.

4. Del cuadro Asignar macro seleccione la que desea asignar a la etiqueta y de un clic en Aceptar y luego Cerrar para finalizar la asignación.



Nota: Desde el *Menú* contextual podrá cambiarle el texto y el icono a la etiqueta.

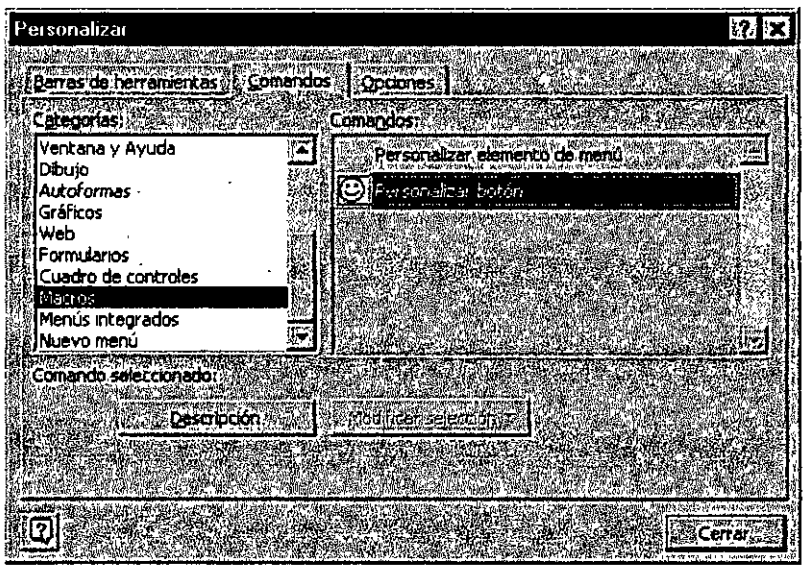
NOTAS:

ASIGNAR MACRO A UN BOTÓN DE LA BARRA DE HERRAMIENTAS

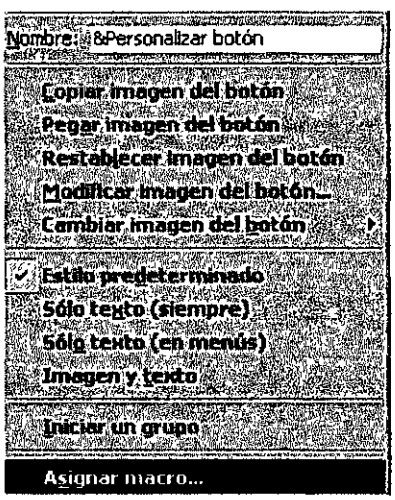
1. En Menú Herramientas, seleccionar la opción Personalizar

2. Seleccionar la ficha Comandos

3. En categorías seleccione Macro y en Comandos seleccione Personalizar botón **y colóquelo** arrastrando con el mouse en la barra de herramientas deseada.



5. Suelte el botón y de clic con el botón secundario del mouse, seleccione la opción Asignar Macro



6. Seleccione la Macro que desee asignar al botón y de clic en Aceptar.

Si desea modificar el dibujo del botón. De clic en el botón secundario del mouse sobre el botón creado, de clic en Cambiar la imagen del botón.

Nota: Para cambiar la imagen del botón cuantas veces queramos o asignar una macro diferente a un botón debemos tener abierta la ventana Personalizar/Categoría: Macros/Comando: Personalizar botón.

NOTAS:

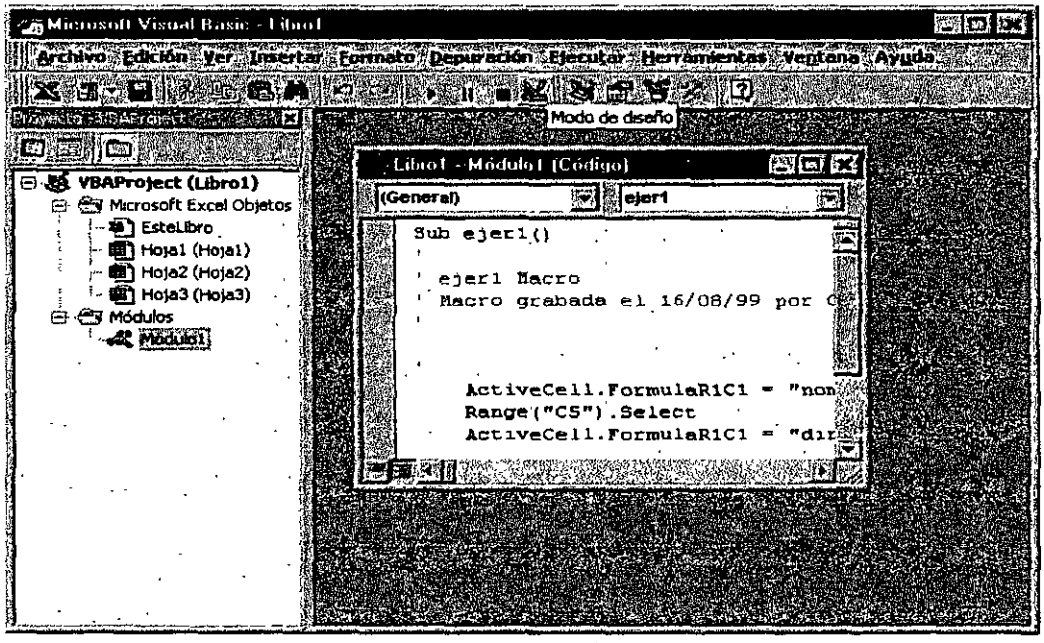
MODIFICAR UNA MACRO

Antes de modificar una macro, deberá familiarizarse con el Editor de Visual Basic. Puede utilizarse el Editor de Visual Basic para escribir y modificar las macros adjuntas a los libros de Microsoft Excel.

1. Seleccione Macro en el menú Herramientas y, a continuación, haga clic en Macros.

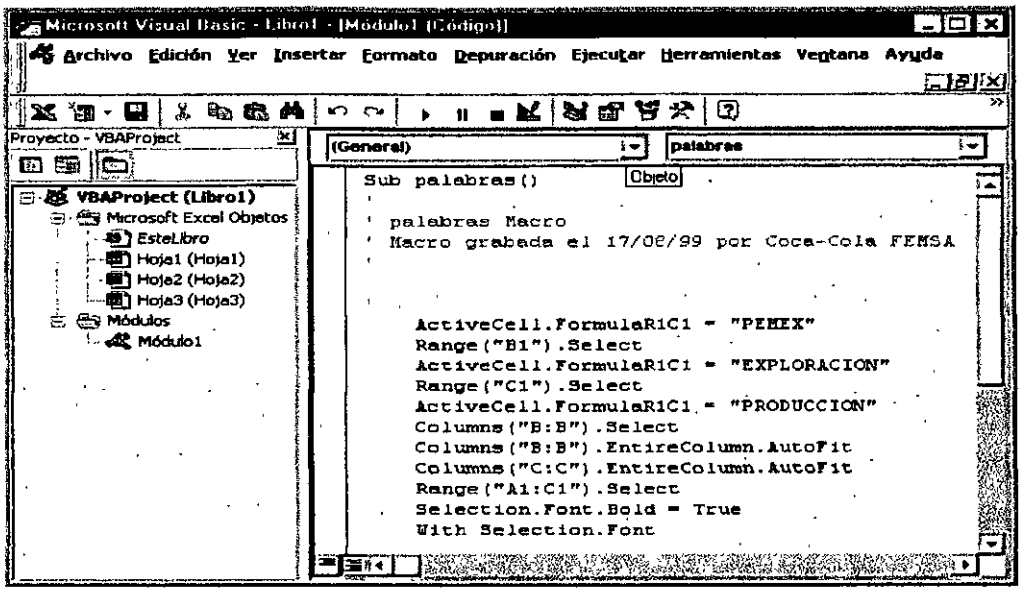
2. En el cuadro Nombre de la macro, seleccione la macro a modificar.

3. Haga clic en Modificar. Y aparece la ventana del Editor de visual Basic mostrando el código de la macro, listo para ser modificado



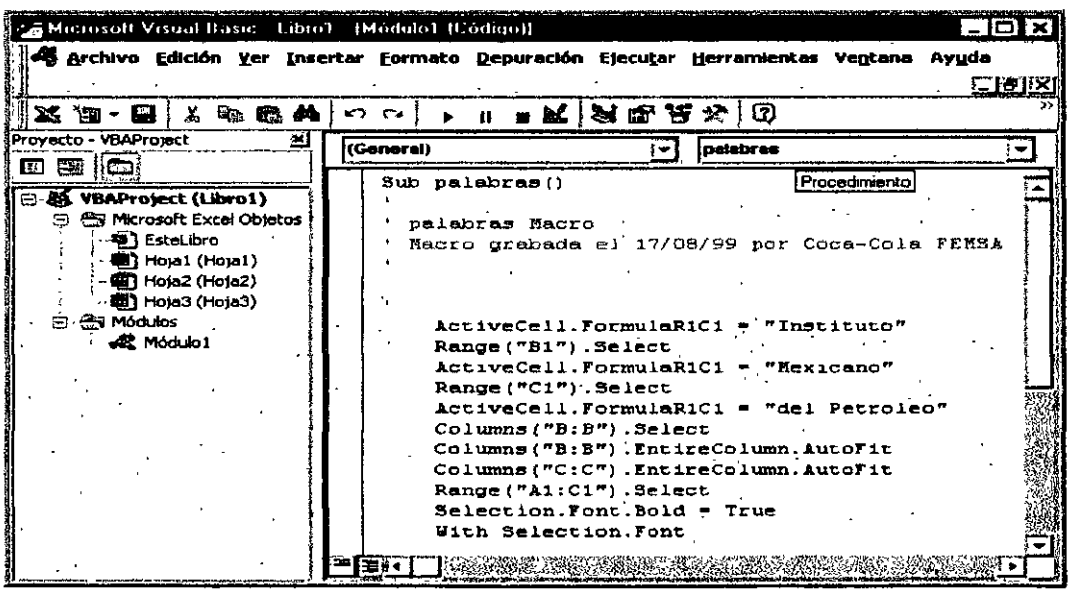
Por ejemplo: Deseamos modificar una macro que escribe las palabras: PEMEX, EXPLORACIÓN, PRODUCCIÓN, cambiándola por Instituto, Mexicano, del Petróleo.

NOTAS:



Si deseamos modificar las palabras escritas en la macro, solo basta sustituirlas por las que queremos , en este caso será por: Instituto, Mexicano, del Petróleo.

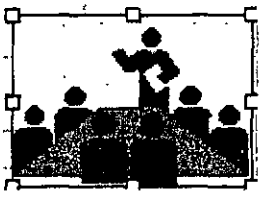
Cerramos el editor y ejecutamos de nuevo la macro, para ver los cambios.



NOTAS:

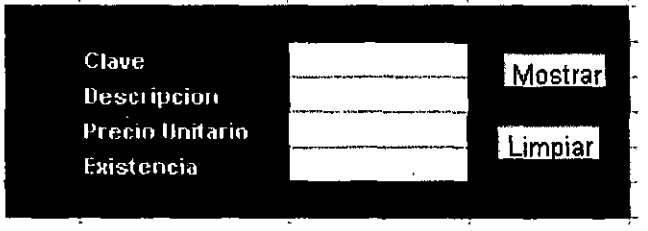
PRÁCTICA 1:

1. Diseñar una macro que active una hoja al dar clic sobre la imagen.



2. Diseñar una macro que visualice los datos de un producto a partir de una clave

A	B	C	D
Clave	Descripción	Precio Unitario	Existencia
12	Comedor	3500	20
23	Sofá	650	30
14	Sillón	350	30
16	Silla	150	10
25	Cama	1000	20



3. Agregar un botón para facilitar la búsqueda, de tal forma que se capture el dato y al dar clic en el botón comience la búsqueda en la base de datos visualizando la información del registro.

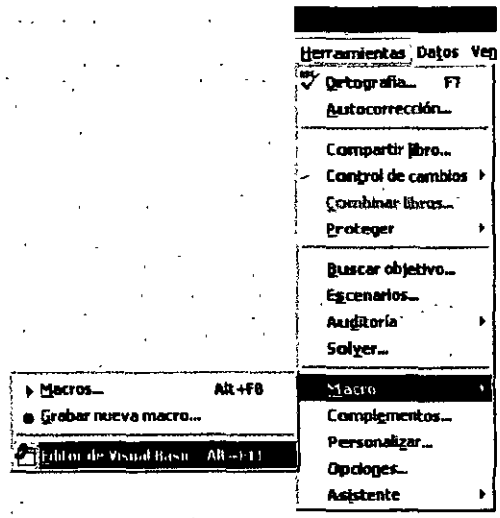
NOTAS:

NOTAS:

CAPÍTULO 2. EDITOR DE VISUAL BASIC

Para poder mostrar el entorno de Visual Basic:

1. Seleccione Macro en el menú Herramientas y, a continuación, haga clic en Editor de Visual Basic



2. Aparece la ventana del Editor el cual consta de las siguientes partes:

Barra de Título

Es la barra horizontal situada en la parte superior de la pantalla, contiene el nombre de la aplicación.

Barra de Menús

Proporciona las herramientas necesarias para desarrollar, probar y archivar la aplicación.

La forman los siguientes elementos:

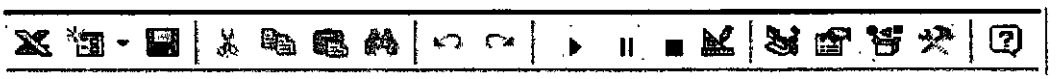
NOTAS:

Barra de Herramientas:

Contiene botones que tienen accesos directos a algunos elementos de menú utilizados con frecuencia.

Puede hacer clic una sola vez en un botón de la barra de herramientas para realizar la acción representada por el botón. Puede seleccionar la opción Información sobre herramientas de la ficha General del cuadro de diálogo Opciones si desea mostrar información sobre los botones de la barra de herramientas.

Botones de la barra de herramientas

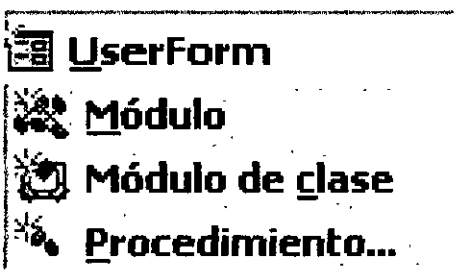


Ver <aplicación principal >

Alterna entre la aplicación principal y el documento de Visual Basic activo.

Insertar User Form

Abre un menú para que se pueda insertar uno de los objetos siguientes en el proyecto activo. El icono cambia al último objeto agregado. El objeto predeterminado es el formulario.



NOTAS:

Guardar <nombre de documento principal >

Guarda el documento principal, incluidos el proyecto y todos sus componentes: formularios y módulos

Cortar

Quita el control o texto seleccionado y lo coloca en el Portapapeles.

Copiar

Copia el control o texto seleccionado en el Portapapeles.

Pegar

Inserta el contenido del Portapapeles en la ubicación actual del cursor.

Buscar

Abre el cuadro de diálogo Buscar y busca el texto especificado en el cuadro Buscar.

Deshacer

Deshace la última acción de edición.

Rehacer

Restaura las últimas acciones descartadas de edición de texto si no se han realizado otras acciones desde la última operación de Deshacer.

Ejecutar Sub/UserForm o Ejecutar macro

Ejecuta el procedimiento actual si el cursor está en un procedimiento, ejecuta el UserForm si un UserForm está activo actualmente o ejecuta una macro si no está activa la ventana Código ni un UserForm.

NOTAS:

nterrumpir

Detiene la ejecución de un programa y cambia al modo de interrupción.

Restablecer <proyecto>

Borra las variables de nivel de módulo de la pila de ejecución y restablece el proyecto.

Modo de diseño

Activa y desactiva el modo de diseño.

Explorador de proyectos

Abre el Explorador de proyectos que muestra una lista jerárquica de los proyectos abiertos actualmente y su contenido.

Ventana de Propiedades

Abre la ventana de Propiedades para que puedan verse las propiedades del control seleccionado.

Examinador de objetos

Muestra el Examinador de objetos, que presenta una lista de bibliotecas de objetos, biblioteca de tipos, clases, métodos, propiedades, eventos y constantes que se pueden utilizar en código, así como los módulos y procedimientos definidos para el proyecto.

Cuadro de herramientas

Muestra u oculta el cuadro de herramientas que contiene todos los controles y los objetos insertables (Como un gráfico de Microsoft Excel) disponibles para la aplicación. Sólo está disponible cuando está activo un UserForm.

Asistente de Office

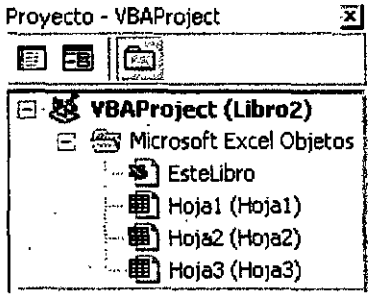
Abre el Asistente de Office donde puede obtener ayuda sobre la ventana, el comando o el cuadro de diálogo que esté activo.

NOTAS:

Explorador de Proyectos

El Explorador de proyectos muestra una lista jerárquica de los proyectos y todos los elementos contenidos o que hace referencia en cada proyecto.

1. En el menú Ver, elija Explorador de proyectos (CTRL+R) o utilice el cuadro de herramientas abreviado: 



ELEMENTOS DE LA VENTANA

Ver Código

Muestra la ventana Código para que pueda escribir y editar código asociado al elemento seleccionado.

Ver Objeto

Muestra la ventana Objeto correspondiente al elemento seleccionado, un módulo, libro, hoja o UserForm seleccionado.

Alternar Carpetas

Oculta y muestra las carpetas de objetos a la vez que muestra los elementos individuales contenidos en dichas carpetas.

NOTAS:

VENTANA DE LISTA

Presenta todos los proyectos cargados y los elementos incluidos en cada proyecto.

Proyecto

El proyecto y los elementos contenidos en él.

Formularios de usuario

Todos los archivos .frm asociados con el proyecto.

Libro

El libro asociado con el proyecto. Por ejemplo, en Microsoft Excel, es el libro y las hojas que lo contienen.

Módulos

Todos los módulos .bas para el proyecto.

Módulos de clase

Todos los archivos .cls del proyecto.

Ventana de propiedades

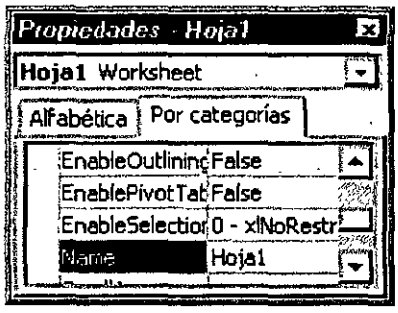
Enumera las propiedades de tiempo de diseño correspondientes a los objetos seleccionados y su configuración actual. Puede cambiar estas propiedades en tiempo de diseño. Cuando seleccione múltiples controles, la ventana de Propiedades contiene una lista de las propiedades comunes a todos los controles seleccionados.

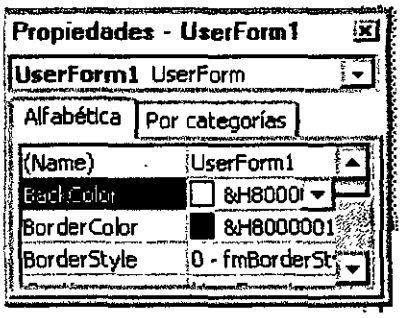
Se puede llamar a la ventana de propiedades usando el icono correspondiente de la barra de herramientas estándar u oprimir la tecla <F4>

lista de las propiedades comunes a todos los controles seleccionados.

NOTAS:

ELEMENTOS DE LA VENTANA





Cuadro Objeto

Presenta el objeto seleccionado actualmente. Sólo están visibles los objetos del formulario activo. Si selecciona múltiples objetos, las propiedades comunes a los objetos y su configuración, en función del primer objeto seleccionado, aparecen en las fichas de Lista de propiedades.

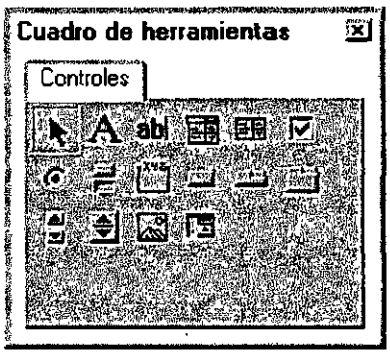
Fichas de Lista de propiedades

Ficha Alfabética — Relaciona alfabéticamente todas las propiedades del objeto seleccionado que se pueden cambiar en el tiempo de diseño, así como su configuración actual. Puede cambiar la configuración de la propiedad seleccionando el nombre de la propiedad y escribiendo o seleccionando la configuración nueva.

Ficha Por categorías — Enumera todas las propiedades del objeto seleccionado por categoría. Por ejemplo, Color de fondo, Título y Color de primer plano están en la categoría Apariencia. Puede contraer la lista para que pueda ver las categorías, o expandir una categoría para ver las propiedades. Cuando expande o contrae la lista, verá un icono con el signo más (+) o menos (-) situado a la izquierda del nombre de la categoría.

NOTAS:

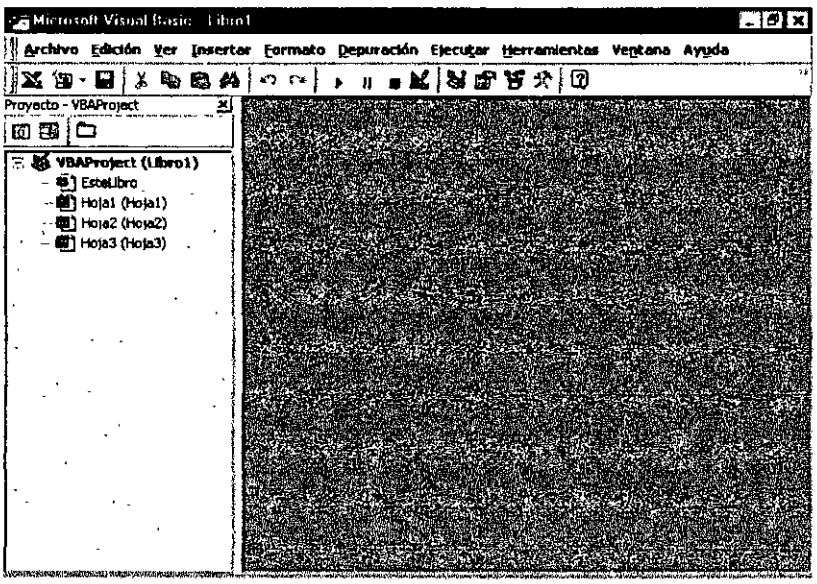
Cuadro de Herramientas



Muestra los controles estándar de Visual Basic junto con los controles ActiveX y los objetos que se pueden insertar que se han agregado al proyecto.

La ventana del formulario inicial

Ocupa la mayor arte del centro de la pantalla. En ella es donde se personaliza la ventana que verán los usuarios. La documentación de Visual Basic utiliza el término form(formulario) para una ventana personalizable.



NOTAS:

CAPÍTULO 3. VISUAL BASIC

PROCEDIMIENTOS

Definición

son una secuencia con nombre de instrucciones que se ejecutan como una unidad. Por ejemplo, Function, Property y Sub son todos tipos de procedimientos. Un nombre de procedimiento siempre se define a nivel de módulo. Todo el código ejecutable debe estar contenido en un procedimiento. Los procedimientos no se pueden anidar dentro de otros procedimientos.

Procedimiento Sub

Un procedimiento Sub es una serie de instrucciones Visual Basic, encerradas entre un par de instrucciones Sub y End Sub, que realizan acciones específicas pero no devuelven ningún valor. Un procedimiento Sub puede aceptar argumentos, como constantes, variables o expresiones que le pasa el procedimiento que ha efectuado la llamada. Si un procedimiento Sub no tiene argumentos, la instrucción Sub debe incluir un par de paréntesis vacío.

Sintaxis

```
Sub

[Private | Public] [Static] Sub nombre [(listaargumentos)]

        [instrucciones]

        [Exit Sub]

        [instrucciones]

End Sub
```

NOTAS:

La sintaxis de la instrucción Sub consta de las siguientes partes:

Public (opcional)

Indica que el procedimiento Sub es accesible a todos los otros procedimientos en todos los módulos. Si se usa en un módulo privado (uno que contiene una instrucción Option Private) el procedimiento no está disponible fuera del proyecto.

Private (Opcional)

Indica que el procedimiento Sub es accesible sólo a otros procedimientos en el módulo donde es declarado.

Static (Opcional)

Indica que las variables locales del procedimiento Sub se conservan entre llamadas. El atributo Static no afecta variables que se declaran fuera de Sub, aún si ellas se usan en el procedimiento.

Nombre (Requerido)

Nombre del Sub; sigue las convenciones de nombres estándar de variables.

Listaargumentos (Opcional)

Lista de variables que representan los argumentos que son pasados al procedimiento Sub cuando es llamado. Las variables múltiples se separan con puntos y coma.

Instrucciones (Opcional)

Cualquier grupo de instrucciones que se ejecutan dentro del cuerpo del procedimiento Sub.

NOTAS:

Ejemplo de la Instrucción Sub

En este ejemplo se utiliza la instrucción Sub para declarar el nombre, argumentos y código que forman el cuerpo del procedimiento Sub.

```
' Definición del procedimiento Sub.
' Sub con dos argumentos.
Sub SubAreaPC(Largo, Ancho)
    Dim Area As Double  ' Declara la variable local.
    If Largo = 0 Or Ancho = 0 Then
    ' Si cualquier argumento = 0.
        Exit Sub        ' Salir inmediatamente de Sub.
    End If
    Area = Largo * Ancho' Calcula el área del rectángulo.
    Debug.Print Area    ' Imprime Area en la ventana de depuración.
End Sub
```

Procedimiento Function

Un procedimiento Function es una serie de instrucciones de Visual Basic encerradas entre dos instrucciones Function y End Function. Un procedimiento Function es similar a un procedimiento Sub, aunque una función puede devolver además un valor. Un procedimiento Function acepta argumentos, como pueden ser constantes, variables o expresiones que le pasa el procedimiento que efectúa la llamada. Si un procedimiento Function no tiene argumentos, la instrucción Function debe incluir un par de paréntesis vacíos. Una función devuelve un valor asignándolo a su nombre en una o más instrucciones del procedimiento.

NOTAS:

En el siguiente ejemplo, la función Celsius calcula grados centígrados a partir de grados Fahrenheit. Cuando se llama a la función desde el procedimiento Principal, se le pasa una variable que contiene el valor del argumento. El resultado de los cálculos se devuelve al procedimiento que efectúo la llamada y se presenta en un cuadro de mensaje.

```
Sub Principal()

    temp = Application.InputBox(Texto:= _

        "Por favor, introduzca la temperatura en grados F.", Tipo:=1)

    MsgBox "La temperatura es " & Celsius(temp) & " grados C."

End Sub

Function Celsius(GradosF)

    Celsius = (GradosF - 32) * 5 / 9

End Function
```

OBJETOS

Un objeto representa un elemento de una aplicación, como una hoja de cálculo, una celda, un diagrama, un formulario o un informe. En código de Visual Basic, un objeto debe identificarse antes de se pueda aplicar uno de los métodos del objeto o cambiar el valor de una de sus propiedades.

Una colección es un objeto que contiene varios objetos que normalmente, pero no siempre, son del mismo tipo.

En Microsoft Excel, por ejemplo, el objeto Workbooks contiene todos los objetos Workbook abiertos. En Visual Basic, la colección Forms contiene todos los objetos Form existentes en una aplicación.

Los elementos de una colección se pueden identificar mediante su número o su nombre. Por ejemplo, en el siguiente procedimiento, Libro(1) identifica al primer objeto Workbook abierto.

NOTAS:

```
Sub CierraPrimero()

    Libro(1).Close

End Sub
```

El siguiente procedimiento utiliza un nombre especificado como cadena para identificar un objeto Form.

```
Sub CierraForm()

    Forms("MiForm.frm").Close

End Sub
```

También es posible operar al mismo tiempo sobre toda una colección de objetos siempre que los objetos compartan métodos comunes. Por ejemplo, el siguiente procedimiento cierra todos los formularios abiertos.

```
Sub CierraTodos()

    Forms.Close

End Sub
```

Métodos

Método es toda acción que puede realizar un objeto. Por ejemplo, Add es un método del objeto ComboBox ya que sirve para añadir un nuevo elemento a un cuadro combinado.

El siguiente procedimiento utiliza el método Add para añadir un nuevo elemento a un ComboBox.

```
Sub AñadeElemen(nuevoElemento as String)

    Combo1.Add nuevoElemento
```

NOTAS:

```
End Sub
```

Propiedades

Propiedad es un atributo de un objeto que define una de las características del objeto, tal como su tamaño, color o localización en la pantalla, o un aspecto de su comportamiento, por ejemplo si está visible o activado. Para cambiar las características de un objeto, se cambia el valor de sus propiedades

Para dar valor a una propiedad, hay que colocar un punto detrás de la referencia a un objeto, después el nombre de la propiedad y finalmente el signo igual (=) y el nuevo valor de la propiedad. Por ejemplo, el siguiente procedimiento cambia el título de un formulario de Visual Basic dando un valor a la propiedad Caption.

```
Sub CambiaNombre(nuevoTitulo)

    miForm.Caption = nuevoTitulo

End Sub
```

Hay propiedades a las que no se puede dar valor. El tema de ayuda de cada propiedad indica si es posible leer y dar valores a la propiedad (lectura/escritura), leer sólo el valor de la propiedad (sólo lectura) o sólo dar valor a la propiedad (sólo escritura).

Se puede obtener información sobre un objeto devolviendo el valor de una de sus propiedades. El siguiente procedimiento utiliza un cuadro de diálogo para presentar el título que aparece en la parte superior del formulario activo en ese momento.

```
Sub NombreFormEs()

    formNonmbre = Screen.ActiveForm.Caption

    MsgBox formNombre

End Sub
```

NOTAS:

Evento

Es toda acción que puede ser reconocida por un objeto, como puede ser el clic del mouse o la pulsación de una tecla y para la que es posible escribir código como respuesta. Los eventos pueden ocurrir como resultado de una acción del usuario o del código de l programa, también pueden ser originados por el sistema.

NOTAS:

CAPÍTULO 4. OBJETOS DE EXCEL

OBJETO APPLICATION

Representa la aplicación completa Microsoft Excel. El objeto Application contiene:

- Valores y opciones de toda la aplicación (por ejemplo, muchas de las opciones del cuadro de diálogo Opciones, en el menú Herramientas).
- Métodos que devuelven objetos de nivel superior, como ActiveCell, ActiveSheet, etc.

Propiedades del Objeto Application

ActiveCell	Devuelve un objeto Range que representa la celda activa de la ventana activa (la ventana superior) o de la ventana especificada. Si la ventana no contiene una hoja de cálculo, esta propiedad fallará. Es de sólo lectura.
ActivePrinter	Devuelve o establece el nombre de la impresora activa. String de Lectura/Escitura.
ActiveSheet	Devuelve un objeto que representa la hoja activa (la hoja en primer plano) del libro activo o de la ventana o el libro especificado. Devuelve Nothing si no hay ninguna hoja activa. Es de sólo lectura.
Caption	Devuelve nombre que aparece en la barra de título de la ventana principal de Microsoft Excel.
Cursor	Devuelve o establece el aspecto del puntero del mouse (ratón) en Microsoft Excel. Long de Lectura/Escritura.
DisplayAlerts	True si Microsoft Excel muestra ciertos mensajes de alerta durante la ejecución de una macro. Boolean de Lectura/Escritura.
Height	Devuelve o establece el alto de un objeto, en puntos. Long de Lectura/Escritura.

NOTAS:

Visible	True si el objeto está visible. En un gráfico o una hoja de cálculo, esta propiedad puede establecerse como xlVeryHidden. Oculta el objeto de tal manera que el único modo de volver a hacerlo visible es estableciendo esta propiedad como True (el usuario no puede hacerlo visible). Boolean o Long de Lectura/Escritura.

Métodos del Objeto Application

Quit	Sale de Microsoft Excel. No ejecuta ninguna macro Auto_cerrar antes de salir. Sintaxis : expresión.Quit
Wait	Realiza una pausa, durante el tiempo especificado, en una macro que se está ejecutando. Sintaxis : expresión. Wait(Time)
InputBox	Muestra un cuadro de diálogo para que el usuario introduzca información. Devuelve la información introducida en el cuadro de diálogo. Sintaxis: expresión.InputBox(Prompt, Title, Default, Left, Top, HelpFile, HelpContextId, Type)
FindFile	Muestra el cuadro de diálogo Abrir Sintaxis: expresión.FindFile

Eventos del Objeto Application

Los eventos de aplicación se producen cuando se crea o se abre un libro, o cuando cambia cualquier hoja de cualquier libro abierto. Para escribir procedimientos de evento para el objeto Application, debe crear un nuevo objeto utilizando la palabra clave WithEvents en un módulo de clase.

NewWorkbook	Ocurre al crear un nuevo libro.
SheetActivate	Ocurre cuando se activa una hoja.

NOTAS:

WindowActivate	Se produce cuando se activa cualquier ventana de libro.
WorkbookActivate	Ocurre cuando se activa cualquier libro.
WorkbookBeforeClose	Ocurre antes de que se cierre cualquier libro.
WorkbookBeforePrint	Ocurre antes de que se imprima cualquier libro abierto.
WorkbookBeforeSave	Ocurre antes de que se guarde cualquier libro abierto.
WorkbookNewSheet	Ocurre al crear una hoja nueva en cualquier libro abierto.
WorkbookOpen	Ocurre al abrir un libro.

Ejemplo de la propiedad ActivePrinter

En este ejemplo se muestra el nombre de la impresora activa.

```
MsgBox "El nombre de la impresora activa es " & Application.ActivePrinter
```

Ejemplo de la propiedad Caption

Este ejemplo asigna un nombre personalizado que aparece en la barra de título de la ventana principal de Microsoft Excel

```
Sub titulo()
    Application.Caption = "Sistema de reservas Aerolíneas Cielo Azul"
End Sub
```

Ejemplo de la propiedad DisplayAlerts

Este ejemplo cierra Libro1.xls y no pregunta al usuario si desea guardar los cambios. No se guardarán los cambios efectuados en Libro1.xls.

```
Sub Desplegar_mensaje()

    Application.DisplayAlerts = False

    Workbooks("LIBRO1").Close
```

NOTAS:

```
    Application.DisplayAlerts = True
End Sub
```

Este ejemplo cierra Libro1.xls y pregunta al usuario si desea guardar los cambios.

```
Sub alerta()
Application.DisplayAlerts = True
Workbooks("LIBRO2").Close
Application.DisplayAlerts = False
End Sub
```

Ejemplo de la propiedad Cursor

Este ejemplo establece la forma del puntero del mouse (ratón) como I, realiza una pausa y, a continuación, lo cambia por el puntero predeterminado.

```
Sub CambiarCursor()
  Application.Cursor = xlIBeam
  For x = 1 To 1000
    For y = 1 To 1000
    Next y
  Next x
  Application.Cursor = xlDefault
End Sub
```

Ejemplo del método Quit

Este ejemplo guarda todos los libros abiertos y, a continuación, sale de Microsoft Excel.

```
For Each I In Application.Workbooks
```

NOTAS:

```
        l.Save
    Next l
    Application.Quit
```

Ejemplo del método Wait

Este ejemplo hace una pausa en una macro en ejecución hasta las 6:23 p.m. de la fecha actual.

```
Sub espera()
    nuevaHora = Hour(Now())
    nuevoMinuto = Minute(Now())
    nuevoSegundo = Second(Now()) + 10
    tiempoEspera = TimeSerial(nuevaHora, nuevoMinuto, nuevoSegundo)
    Application.Wait tiempoEspera
End Sub
```

Ejemplo del método InputBox

Este ejemplo solicita un número al usuario.

```
Sub Entrar()
miNum = Application.InputBox("Introduzca un número")
End Sub
```

Este ejemplo solicita al usuario que seleccione una celda de la Hoja1. El ejemplo usa el argumento Type para asegurarse de que valor devuelto es una referencia de celda válida (un objeto Range).

```
    Worksheets("Hoja1").Activate
```

NOTAS:

```
Set miCelda = Application.InputBox( _
        prompt:="Seleccione una celda", Type:=8)
```

Ejemplo del método FindFile

Este ejemplo muestra el cuadro de diálogo Abrir.

```
Sub Abrir()
    Application.FindFile
End  Sub
```

Ejemplo del evento NewWorkbook

Este ejemplo organiza las ventanas abiertas cuando se crea un nuevo libro.

```
Private Sub App_NewWorkbook(ByVal Wb As Workbook)
    Application.Windows.Arrange xlArrangeStyleTiled
End Sub
```

Ejemplo del evento SheetActivate

Este ejemplo muestra el nombre de cada hoja activada.

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
    MsgBox Sh.Name
End Sub
```

OBJETOS WORKBOOK

Representa un libro de Microsoft Excel. El objeto Workbook es un elemento del conjunto Workbooks. El conjunto Workbooks contiene todos los objetos Workbook abiertos actualmente en Microsoft Excel.

NOTAS:

Propiedades del objeto Workbook

ActiveSheet	Devuelve un objeto que representa la hoja activa (la hoja en primer plano) del libro activo o de la ventana o el libro especificado. Devuelve Nothing si no hay ninguna hoja activa. Es de sólo lectura.
Colors	Devuelve o establece los colores de la paleta del libro. La paleta tiene 56 elementos, representado cada uno por un valor RGB. Variant de Lectura/Escritura.
HasPassword	True si el libro está protegido con una contraseña. Boolean de sólo lectura.
Sheets	Devuelve un conjunto Sheets que representa todas las hojas del libro especificado. Es de sólo lectura.
Saved	True si no se han efectuado cambios en el libro especificado desde la última vez que se guardó. Boolean de Lectura/Escritura.
Worksheets	Devuelve un conjunto Sheets que representa todas las hojas de cálculo del libro especificado. Es de sólo lectura.

Ejemplo de la propiedad ActiveSheet

Este ejemplo muestra el nombre de la hoja activa.

```
Sub hoja_activa( )

    MsgBox "El nombre de la hoja activa es " & ActiveSheet.Name

End Sub
```

Ejemplo de la propiedad HasPassword

Este ejemplo muestra un mensaje si el libro activo está protegido por contraseña.

```
If ActiveWorkbook.HasPassword = True Then

    MsgBox "No olvide solicitar la contraseña del libro" & Chr(13) & _

        " al administrador de la red."
```

NOTAS:

```
End If
```

Ejemplo de la propiedad Sheets

Este ejemplo crea una nueva hoja de cálculo y, a continuación, sitúa una lista de los nombres de hoja del libro activo en la primera columna.

```
Sub nuevo()

Set nuevaHoja = Sheets.Add(Type:=xlWorksheet)

For i = 1 To Sheets.Count

    nuevaHoja.Cells(i, 1).Value = Sheets(i).Name

Next i

End Sub
```

Ejemplo de la propiedad Saved

Este ejemplo muestra un mensaje si el libro activo contiene cambios no guardados.

```
Sub guardardado()

If Not ActiveWorkbook.Saved Then

    MsgBox "Este libro contiene cambios no guardados."

End If

End Sub
```

Ejemplo de la propiedad Worksheets

```
Sub abrir()

MsgBox Worksheets("Hoja1").Range("A1").Value

End Sub
```

Este ejemplo muestra el nombre de cada una de las hojas de cálculo del libro activo.

NOTAS:

```
Sub nombres( )

    For Each hc In Worksheets

    MsgBox hc.Name

Next hc

End Sub
```

Este ejemplo agrega una nueva hoja al libro activo y, a continuación, establece el nombre de la hoja.

```
Sub nueva_hoja()

Set nuevaHoja = Worksheets.Add

nuevaHoja.Name = "Presupuesto"

End Sub
```

Métodos del objeto Workbook

Activate	Activa el objeto
Close	Cierra el objeto. El conjunto Workbooks usa la sintaxis expresión.Close
NewWindow	Crea una ventana nueva o una copia de la ventana especificada.
PrintOut	Imprime el objeto.
Save	Guarda los cambios del libro especificado.

Ejemplo del método Activate

Este ejemplo activa la Hoja1.

```
Sub activar()

Worksheets("Hoja1").Activate
```

NOTAS:

End Sub

Este ejemplo activa el libro Libro4.xls. Si Libro4.xls tiene varias ventanas, el ejemplo activará la primera ventana, Libro4.xls:1.

```
Workbooks("LIBRO4.XLS").Activate
```

Ejemplo del método Close

Este ejemplo cierra Libro1.xls y descarta los cambios efectuados en él.

```
Workbooks("LIBRO1.XLS").Close SaveChanges:=False
```

Este ejemplo cierra todos los libros abiertos. Si hay cambios en algún libro abierto, Microsoft Excel mostrará los mensajes y cuadros de diálogo oportunos para preguntar si desea guardar los cambios.

```
Workbooks.Close
```

Ejemplo del método NewWindow

Este ejemplo crea una nueva ventana para el libro activo.

```
Sub ventana()
    ActiveWorkbook.NewWindow
End Sub
```

Ejemplo del método Save

Este ejemplo guarda el libro activo.

```
ActiveWorkbook.Save
```

Este ejemplo guarda todos los libros abiertos y, a continuación, cierra Microsoft Excel.

```
Sub guardartodos()
For Each I In Application.Workbooks
    w.Save
```

NOTAS:

```
Next I

Application.Quit

End Sub
```

Eventos del objeto Workbook

Los eventos de libros se producen cuando el libro cambia o cuando cambia cualquier hoja del mismo. De forma predeterminada, los eventos en libros están habilitados. Para ver los procedimientos de eventos de un libro, con el botón secundario del mouse (ratón), haga clic en la barra de título de una ventana de libro restaurada o minimizada y, a continuación, en Ver código en el menú contextual. Seleccione el nombre del evento en el cuadro de lista emergente Procedimiento.

Activate	Ocurre cuando se activa un libro, una hoja de cálculo, una hoja de gráfico o un gráfico incrustado.
AddinInstall	Ocurre al instalar el libro como macro automática
AddinUninstall	Ocurre al desinstalar el libro como macro automática
BeforeClose	Ocurre antes de que el libro se cierre. Si el libro ha sido modificado, este evento ocurre antes de pedir al usuario que guarde las modificaciones.
BeforePrint	Ocurre antes de que se imprima el libro o cualquiera de sus partes
BeforeSave	Ocurre antes de que el libro se guarde.
Deactivate	Ocurre cuando se desactiva el gráfico, la hoja de cálculo o el libro.
NewSheet	Ocurre al crear una hoja nueva en el libro. Sintaxis: Private Sub Workbook_NewSheet(ByVal Sh As Object)

NOTAS:

Open	Se produce cuando el libro se abre. Sintaxis: Private Sub Workbook_Open()
SheetActivate	Ocurre cuando se activa una hoja.
SheetBeforeDoubleClick	Ocurre al hacer doble clic en una hoja de cálculo, antes de la acción predeterminada para el doble clic.
SheetBeforeRightClick	Ocurre al hacer clic con el botón secundario del mouse (ratón) en una hoja de cálculo, antes de la acción predeterminada.
SheetCalculate	Ocurre después de recalcular una hoja de cálculo o de trazar los datos modificados de un gráfico.
SheetChange	Ocurre cuando una celda de cualquier hoja de cálculo es modificada por el usuario o por un vínculo externo.
SheetDeactivate	Ocurre cuando se desactiva cualquier hoja.
SheetSelectionChange	Se produce cuando la selección cambia el cualquier libro (no se produce si la selección está en una hoja de gráfico.
WindowActivate	Se produce cuando se activa cualquier ventana de libro.
WindowDeactivate	Se produce cuando se desactiva cualquier ventana de libro.
WindowResize	Se produce cuando se cambia el tamaño de cualquier ventana de libro.

OBJETO DE CONJUNTO WORKSHEETS

Un conjunto de todos los objetos Worksheet del libro especificado o activo. Cada objeto Worksheet representa una hoja de cálculo.

El objeto Worksheet también es un elemento del conjunto Sheets. El conjunto Sheets contiene todas las hojas del libro (hojas de gráficos y hojas de cálculo).

NOTAS:

Propiedades del objeto Worksheets

*Item(*conjunto Worksheets*)*	Devuelve un único objeto Worksheet de un conjunto Worksheets. Sintaxis: expresión.Item(Index) Index : El nombre o índice de la hoja de cálculo.
Visible	True si el objeto está visible. En un gráfico o una hoja de cálculo, esta propiedad puede establecerse como xlVeryHidden. Oculta el objeto de tal manera que el único modo de volver a hacerlo visible es estableciendo esta propiedad como True (el usuario no puede hacerlo visible). Boolean o Long de Lectura/Escritura.

Métodos del objeto Worksheets

Add(conjunto Worksheets*)*	Crea una hoja de cálculo nueva. La nueva hoja de cálculo se convertirá en la hoja activa. Devuelve un objeto Worksheet.
Copy	Copia la hoja en otro lugar del libro. Sintaxis: expresión.Copy(Before, After)
Delete	Elimina el objeto . Sintaxis : expresión.Delete
FillAcrossSheets	Copia un rango en el mismo lugar en toda una serie de hojas de cálculo.
Move	Desplaza la hoja de cálculo a otro lugar del libro.
PrintOut	Imprime el objeto.
PrintPreview	Muestra una vista preliminar del objeto tal como aparecería impreso.
Select	Sintaxis :expresión.Select(Replace)

Ejemplo del método Add

Este ejemplo crea una hoja de cálculo nueva y la inserta antes de la hoja activa.

```
ActiveWorkbook.Worksheets.Add
```

NOTAS:

Este ejemplo agrega una hoja de cálculo nueva después de la última hoja de cálculo del libro activo.

```
Worksheets.Add.Move after:=Worksheets(Worksheets.Count)
```

Ejemplo del método Copy

En este ejemplo se copia la Hoja1, colocando la copia después de la Hoja3.

```
Worksheets("Hoja1").Copy after := Worksheets("Hoja3")
```

En este ejemplo se copia el rango utilizado de la Hoja1, se crea una nueva hoja de cálculo y, a continuación, se pegan los valores del rango copiado en la nueva hoja de cálculo.

```
Worksheets("Hoja1").UsedRange.Copy

Set nuevaHoja = Worksheets.Add

nuevaHoja.Range("A1").PasteSpecial Paste:=xlValues
```

Ejemplo del método Delete

En este ejemplo se eliminan las celdas A1:D10 de la Hoja1 y se desplazan las celdas restantes hacia la izquierda.

```
Worksheets("Hoja1").Range("A1:D10").Delete Shift:=xlShiftToLeft
```

Este ejemplo elimina todas las hojas de cálculo del libro activo sin mostrar el cuadro de diálogo de confirmación.

```
Application.DisplayAlerts = False

For Each w In Worksheets

    w.Delete

Next w

Application.DisplayAlerts = True
```

Ejemplo del método Move

NOTAS:

En este ejemplo se desplaza la Hoja1 detrás de la Hoja3 en el libro activo.

```
Worksheets("Hoja1").Move _
        after:=Worksheets("Hoja3")
```

Ejemplo del método PrintPreview

En este ejemplo se muestra la Hoja1 en vista preliminar.

```
Worksheets("Hoja1").PrintPreview
```

Ejemplo del método Select

Este ejemplo selecciona las celdas A1:B3 de la Hoja1.

```
Worksheets("Hoja1").Activate
Range("A1:B3").Select
```

OBJETO WORKSHEET

Representa una hoja de cálculo. El objeto Worksheet es un elemento del conjunto Worksheets.

Propiedades de Worksheet

Cells	Devuelve un objeto Range que representa las celdas de la hoja de cálculo (no sólo las celdas que se usan actualmente).
Shapes	Devuelve un objeto Shapes que representa todas las formas del libro u hoja de gráficos.
Columns	devuelve un objeto Range que representa todas las columnas de la hoja de cálculo especificada.
Visible	True si el objeto está visible.

Ejemplo de la propiedad Cells

NOTAS:

Este ejemplo establece en Arial de 8 puntos la fuente y el tamaño de fuente de todas las celdas de la Hoja1.

```
Sub Celdas()

With Worksheets("Hoja1").Cells.Font

    .Name = "Arial"

    .Size = 8

End With


End Sub
```

Este ejemplo establece en 14 puntos el tamaño de fuente de la celda C5 de la Hoja1.

```
Sub tamaño()

Worksheets("Hoja1").Cells(5, 3).Font.Size = 14

End sub
```

Ejemplo de la propiedad Shapes

```
Sub figura()

With Worksheets(1).Shapes.AddLine(10, 10, 250, 250).Line

   .DashStyle = msoLineDashDotDot

   .ForeColor.RGB = RGB(50, 0, 128)

End With

End Sub
```

Ejemplo de la propiedad Columns

```
Worksheets("Hoja1").Columns(1).Font.Bold = True
```

NOTAS:

Ejemplo de la propiedad Visible

Este ejemplo oculta la Hoja1.

```
Worksheets("Hoja1").Visible = False
```

Este ejemplo hace visible la Hoja1.

```
Worksheets("Hoja1").Visible = True
```

Métodos de Worksheet

Activate	Convierte esta hoja en la hoja activa. Equivale a hacer clic en la etiqueta de la hoja.
Move	
Delete	
Protect	Protege un gráfico o una hoja de cálculo (sintaxis 1) expresión.Protect(Password, DrawingObjects, Contents, Scenarios, UserInterfaceOnly) o bien un libro para que no sea posible modificarlo. (sintaxis 2) expresión.Protect(Password, Structure, Windows)
ShowDataForm	Muestra el formulario de datos correspondiente a la hoja de cálculo. Sintaxis: expresión.ShowDataForm

Ejemplo del método Activate

```
Worksheets("Hoja1").Activate
```

Ejemplo del método Protect

Este ejemplo protege el libro activo.

```
Sub Proteger( )
```

NOTAS:

```
ActiveWorkbook.Protect Password := "drowssap"

End Sub
```

Ejemplo del método ShowDataForm

Este ejemplo muestra el formulario de datos de la Hoja1.

```
Sub mostrar()

    Worksheets(1).ShowDataForm

End Sub
```

Eventos Worksheet

Activate	Ocurre cuando se activa un libro, una hoja de cálculo, una hoja de gráfico o un gráfico incrustado.
BeforeDoubleClick	Ocurre al hacer doble clic en un gráfico incrustado o en una hoja de cálculo incrustada, antes de la acción predeterminada para el doble clic.
BeforeRightClick	Ocurre al hacer clic con el botón secundario en un gráfico incrustado o en una hoja de cálculo incrustada, antes de la acción predeterminada.
Calculate	Ocurre después de que se recalcule la hoja de cálculo.
Deactivate	Ocurre cuando se desactiva el gráfico, la hoja de cálculo o el libro.
Change	Ocurre cuando alguna celda de la hoja de cálculo es modificada por el usuario o por un vínculo externo.
SelectionChange	Se produce cuando la selección de un libro cambia.

Ejemplo del evento Change

NOTAS:

Este ejemplo cambia a azul el color de las celdas modificadas.

```
Private Sub Worksheet_Change(ByVal Target as Range)
    Target.Font.ColorIndex = 5
End Sub
```

OBJETO DE CONJUNTO SHEETS

Un conjunto de todas las hojas del libro especificado o activo. El conjunto Sheets puede contener objetos Chart o Worksheet.

El conjunto Sheets es útil cuando se desean obtener hojas de cualquier tipo.

Uso del conjunto Sheets

Use la propiedad Sheets para devolver el conjunto Sheets. El ejemplo siguiente imprime todas las hojas del libro activo.

```
Sheets.PrintOut
```

Use el método Add para crear una nueva hoja y agregarla al conjunto. El ejemplo siguiente agrega dos hojas de gráficos al libro activo, situándolas después de la hoja dos del libro.

```
Sheets.Add type:=xlChart, count:=2, after:=Sheets(2)
```

Para devolver un solo objeto Chart o Worksheet, use Sheets(índice), donde índice es el nombre o número de índice de la hoja. El ejemplo siguiente activa la hoja llamada "hoja1".

```
Sheets("Hoja1").Activate
```

Use Sheets(matriz) para especificar varias hojas. El ejemplo siguiente mueve las hojas llamadas "Hoja4" y "Hoja5" al principio del libro.

```
Sheets(Array("Hoja4", "Hoja5")).Move before:=Sheets(1)
```

NOTAS:

Propiedades de objeto Sheet

Application	uando se usa sin calificador de objeto, esta propiedad devuelve un objeto Application que representa a la aplicación Microsoft Excel. Cuando se usa con calificador de objeto, esta propiedad devuelve un objeto Application que representa al creador del objeto especificado (puede usar la propiedad con un objeto de Automatización para devolver la aplicación del objeto). Es de sólo lectura.
Count	Devuelve el número de objetos del conjunto. Long de sólo lectura.
HPageBreaks	Devuelve un conjunto HPageBreaks que representa los saltos de página horizontales de la hoja. Es de sólo lectura.
Item	Devuelve un único objeto Chart o Worksheet de un conjunto Sheets. Sintaxis: expresión.Item(Index) expresión : Una expresión que devuelve un objeto Sheets. Index (Variant): El nombre o el índice de la hoja.
Visible	True si el objeto está visible. En un gráfico o una hoja de cálculo, esta propiedad puede establecerse como xlVeryHidden. Oculta el objeto de tal manera que el único modo de volver a hacerlo visible es estableciendo esta propiedad como True (el usuario no puede hacerlo visible). Boolean o Long de Lectura/Escritura.
VPageBreaks	Devuelve un conjunto VPageBreaks que representa los saltos de página verticales de la hoja. Es de sólo lectura. VPageBreaks :Un conjunto de saltos de página verticales. Cada salto de página vertical está representado por un objeto VPageBreak.

Ejemplo de la propiedad Item (conjunto Sheets)

Este ejemplo activa la Hoja1.

```
Sheets.Item("Hoja1").Activate
```

Ejemplo de la propiedad Visible

NOTAS:

Este ejemplo oculta la Hoja1.

```
Worksheets("Hoja1").Visible = False
```

Este ejemplo hace visible la Hoja1.

```
Worksheets("Hoja1").Visible = True
```

Este ejemplo hace visibles todas las hojas del libro activo.

```
For Each hj In Sheets
    hj.Visible = True
Next hj
```

Métodos de objeto Sheet

Add (conjunto Sheets)	Crea una hoja de cálculo, un gráfico o una macro. Sintaxis: expresión.Add(Before, After, Count, Type) Count Variant opcional. El número de hojas de cálculo que se agregarán. El valor predeterminado es uno. Type *Variant opcional. Especifica el tipo de la hoja.El valor* predeterminado es xlWorksheet.
Copy	Sintaxis 1: expresión.Copy. Copia el objeto en el Portapapeles. Copia una imagen del punto o la serie en el Portapapeles. Sintaxis 2: expresión.Copy(Destination). Copia el rango en el rango especificado o en el Portapapeles. Sintaxis 3: expresión.Copy(Before, After) : Copia la hoja en otro lugar del libro.
Delete	Elimina el objeto. Sintaxis: expresión.Delete
FillAcrossSheets	Copia un rango en el mismo lugar en toda una serie de hojas de cálculo.

NOTAS:

	Sintaxis: expresión.FillAcrossSheets(Range, Type) expresión : Una expresión que devuelve un objeto Sheets o Worksheets. Range :El rango con el que se rellenará un conjunto de hojas. El rango debe ser de una de las hojas de calculo del conjunto. Type (Variant): Especifica como se copiará el rango. Puede ser una de las siguientes constantes XlFillWith: xlFillWithAll, xlFillWithContents o xlFillWithFormulas. El valor predeterminado es xlFillWithAll.
Move	Desplaza la hoja de cálculo a otro lugar del libro. Sintaxis: expresión.Move(Before, After)
PrintOut	Imprime el objeto.
PrintPreview	Muestra una vista preliminar del objeto tal como aparecería impreso. Sintaxis: expresión.PrintPreview
Select	Selecciona el objeto. Sintaxis: expresión.Select(Replace) Replace Variant opcional (sólo se utiliza con hojas). True para reemplazar la selección actual con el objeto especificado. Si es False, la selección actual se extiende para incluir cualquier objeto que haya sido seleccionado previamente y al objeto especificado.

Ejemplo del método Add

Este ejemplo inserta una hoja de cálculo nueva antes de la última hoja de cálculo del libro activo.

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

Ejemplo del método Copy

En este ejemplo se copia la Hoja1, colocando la copia después de la Hoja3.

```
Worksheets("Hoja1").Copy after := Worksheets("Hoja3")
```

NOTAS:

En este ejemplo se copia el rango utilizado de la Hoja1, se crea una nueva hoja de cálculo y, a continuación, se pegan los valores del rango copiado en la nueva hoja de cálculo.

```
Worksheets("Hoja1").UsedRange.Copy

Set nuevaHoja = Worksheets.Add

nuevaHoja.Range("A1").PasteSpecial Paste:=xlValues
```

En este ejemplo se copian las fórmulas de las celdas A1:D4 de la Hoja1 en las celdas E5:H8 de la Hoja2.

```
Worksheets("Hoja1").Range("A1:D4").Copy _

destination:=Worksheets("Hoja2").Range("E5")
```

Ejemplo del método Delete

En este ejemplo se eliminan las celdas A1:D10 de la Hoja1 y se desplazan las celdas restantes hacia la izquierda.

```
Worksheets("Hoja1").Range("A1:D10").Delete Shift:=xlShiftToLeft
```

Ejemplo del método FillAcrossSheets

Este ejemplo rellena el rango A1:C5 de la Hoja1, la Hoja5 y la Hoja7 con el contenido del mismo rango de la Hoja1.

```
x = Array("Hoja1", "Hoja5", "Hoja7")

Sheets(x).FillAcrossSheets _

Worksheets("Hoja1").Range("A1:C5")
```

Ejemplo del método Move

En este ejemplo se desplaza la Hoja1 detrás de la Hoja3 en el libro activo.

```
Worksheets("Hoja1").Move _

after:=Worksheets("Hoja3")
```

NOTAS:

Ejemplo del método PrintOut

En este ejemplo se imprime la hoja activa.

```
ActiveSheet.PrintOut
```

Ejemplo del método PrintPreview

En este ejemplo se muestra la Hoja1 en vista preliminar.

```
Worksheets("Hoja1").PrintPreview
```

Ejemplo del método Select

Este ejemplo selecciona las celdas A1:B3 de la Hoja1.

```
Worksheets("Hoja1").Activate
Range("A1:B3").Select
```

OBJETO RANGE

Representa una celda, una fila, una columna, una selección de celdas que contienen uno o más bloques contiguos de celdas o un rango 3D.

Propiedades del objeto Range

Range	Use Range(arg), donde arg asigna un nombre al rango, para devolver un objeto Range que represente una sola celda o un rango de celdas.
Cells	Use Cells(fila; columna), donde fila es el índice de fila y columna es el índice de columna, para devolver una sola celda.
Range y Cells	Para devolver un objeto Range use Range(celda1; celda2), donde celda1 y celda2 son objetos Range que especifican las celdas inicial y final.
Offset	Use Offset(fila; columna), donde fila y columna son los desplazamientos de fila y columna, para devolver un rango con un desplazamiento

NOTAS:

	específico con respecto a otro.
Areas	La propiedad Areas es muy útil para trabajar con selecciones que contienen varias áreas. Divide una selección de varias áreas en objetos Range individuales y después devuelve los objetos en forma de conjunto.
Value	El valor de la celda especificada. Si la celda está vacía, Value devuelve el valor Empty. Si el objeto Range contiene más de una celda, devuelve una matriz de valores.

Ejemplo de la Propiedad Range

El ejemplo siguiente coloca el valor de la celda A1 en la celda A5.

```
Worksheets("Hoja1").Range("A5").Value = _
    Worksheets("Hoja1").Range("A1").Value
```

El ejemplo siguiente llena el rango A1:H8 con números aleatorios estableciendo la fórmula de cada celda del rango. La propiedad Range, si se emplea sin un calificador de objeto (un objeto colocado a la izquierda del punto), devuelve un rango de la hoja activa. Si la hoja activa no es una hoja de cálculo, este método fallará. Use el método Activate para activar una hoja de cálculo antes de usar la propiedad Range sin un calificador de objeto explícito.

```
Worksheets("Hoja1").Activate
Range("A1:H8").Formula = "=aleatorio()"    'El rango está en la hoja activa
```

El ejemplo siguiente borra el contenido del rango llamado "Criterios".

```
Worksheets(1).Range("criterios").ClearContents
```

Si usa un argumento de texto para la dirección del rango, deberá especificar la dirección en notación de estilo A1 (no podrá usar la notación F1C1).

Ejemplo de Propiedad Cells

El ejemplo siguiente establece en 24 el valor de la celda A1.

NOTAS:

```
Worksheets(1).Cells(1, 1).Value = 24
```

El ejemplo siguiente establece la fórmula de la celda A2.

```
ActiveSheet.Cells(2, 1).Formula = "=suma(b1:b5)"
```

Aunque también puede usar Range("A1") para devolver la celda A1, en algunas ocasiones la propiedad Cells puede ser más conveniente, ya que permite usar una variable para la fila o la columna. El ejemplo siguiente crea encabezados de fila y columna en la Hoja1. Tenga en cuenta que, después de activar la hoja de cálculo, puede usar la propiedad Cells sin una declaración explícita de hoja (devuelve una celda de la hoja activa).

```
Sub SetUpTable()

Worksheets("Hoja1").Activate

For elAño = 1 To 5

    Cells(1, elAño + 1).Value = 1990 + elAño

Next elAño

'

For elSemestre = 1 To 4

    Cells(elSemestre + 1, 1).Value = "Q" & elSemestre

Next elSemestre

End Sub
```

Para devolver parte de un rango use expresión.Cells(fila; columna), donde expresión es una expresión que devuelve un objeto Range y fila y columna son relativas a la esquina superior izquierda del rango.

El ejemplo siguiente establece la fórmula de la celda C5.

```
Worksheets(1).Range("c5:c10").Cells(1, 1).Formula = "=aleatorio()"
```

NOTAS:

El siguiente ejemplo rellena las primeras 20 celdas de la tercera columna con valores entre 5 y 100, en incrementos de 5. La variable contador se utiliza como índice de fila para la propiedad Cells.

```
Sub BucleAtravés()

    Dim contador As Integer

    For contador = 1 To 20

        Worksheets("Hoja1").Cells(contador, 3).Value = contador * 5

    Next contador

End Sub
```

Ejemplo de propiedade Range y Cells

El ejemplo siguiente establece el estilo de línea de los bordes de las celdas 1:J10.

```
With Worksheets(1)

  .Range(.Cells(1, 1), .Cells(10, 10)).Borders.LineStyle = xlThick

End With
```

Observe el punto delante de cada propiedad Cells. El punto es necesario si el resultado del enunciado With precedente se aplica a la propiedad Cells, en cuyo caso, se indica que las celdas están en la hoja de cálculo uno (sin el punto, la propiedad Cells devolvería las celdas de la hoja activa).

Ejemplo de Propiedad Offset

El ejemplo siguiente selecciona la celda situada tres filas debajo y una columna a la derecha de la celda de la esquina superior izquierda de la selección actual. No se puede seleccionar una celda que no esté en la hoja activa, por lo que primero deberá activar la hoja.

```
Worksheets("hoja1").Activate  'no se puede seleccionar si la hoja no es la activa

Selection.Offset(3, 1).Range("a1").Select
```

NOTAS:

Ejemplo de la Propiedad Value

Este ejemplo establece el valor de la celda A1 de la Hoja1 como 3.14159.

```
Sub Valor()

Worksheets("Hoja1").Range("A1").Value = 3.14159

End Sub
```

Métodos del objeto Range

Union	se Union(rango1, rango2, ...) para devolver rangos de varias áreas, es decir, rangos compuestos por dos o más bloques contiguos de celdas.
Activate	Activa una sola celda, que debe estar en la selección actual. Para seleccionar un rango de celdas, use el método Select. Sintaxis: expresión.Activate
AutoFill	Rellena automáticamente las celdas del rango especificado. Sintaxis: expresión.AutoFill(Destination, Type)
Clear	Quita todo el objeto.
Delete	Elimina el objeto. Sintaxis: expresión.Delete(Shift) Shift Variant opcional. Se usa únicamente con objetos Range. Especifica cómo se desplazan las celdas para reemplazar celdas eliminadas. Puede ser una de las siguientes constantes XlDeleteShiftDirection: xlShiftToLeft o xlShiftUp. Si este argumento se omite, Microsoft Excel tomará una decisión basándose en la forma del rango.
FillLeft	Rellena hacia la izquierda desde la celda o celdas que están más a la derecha en el rango especificado. El contenido y formato de la celda o celdas que estén en la fila del extremo derecho de un rango se copiarán en el resto de las columnas del rango. Sintaxis: expresión.FillLeft

NOTAS:

FillDown	Rellena hacia abajo desde la celda o celdas superiores del rango especificado hasta el final del rango. El contenido y formato de la celda o celdas que están en la fila superior de un rango se copiarán en el resto de las filas del rango. Sintaxis: expresión.FillDown
FillRight	Rellena hacia la derecha desde la celda o celdas que están más a la izquierda en el rango especificado. El contenido y formato de la celda o celdas que estén en la columna del extremo izquierdo de un rango se copiará en el resto de las columnas del rango. Sintaxis: expresión.FillRight
FillUp	Rellena hacia arriba desde la celda o celdas inferiores del rango especificado hasta el principio del rango. El contenido y formato de la celda o celdas que están en la fila inferior de un rango se copiarán en el resto de las filas del rango.
FunctionWizard	Ejecuta el Asistente para funciones en la celda superior izquierda del rango. Sintaxis: expresión.FunctionWizard
Insert	Inserta una celda o un rango de celdas en la hoja de cálculo o en la hoja de macros y desplaza las otras celdas para crear espacio. Sintaxis: expresión.Insert(Shift)
Replace	Busca y reemplaza caracteres en las celdas de un rango. No cambia la selección ni la celda activa.
Select	Selecciona el objeto.
Show	Desplaza el contenido de la ventana activa para hacer visible el rango. El rango debe ser una celda individual del documento activo. Sintaxis: expresión.Show
Sort	Ordena una tabla dinámica, un rango o la región actual (si el rango especificado sólo contiene una celda).

Ejemplo de Método Union

El ejemplo siguiente crea un objeto definido como la unión de los rangos A1:B2 y C3:D4 y, a continuación, selecciona el rango definido.

```
Dim r1 As Range, r2 As Range, miRangoMultiÁrea As Range
```

NOTAS:

```
Worksheets("Hoja1").Activate

Set r1 = Range("A1:B2")

Set r2 = Range("C3:D4")

Set miRangoMultiÁrea = Union(r1, r2)

miRangoMultiÁrea.Select
```

Ejemplo del método Activate

Este ejemplo activa la Hoja1.

```
Worksheets("Hoja1").Activate
```

Este ejemplo selecciona las celdas A1:C3 de la Hoja1 y convierte a la celda B2 en la celda activa.

```
Worksheets("Hoja1").Activate

Range("A1:C3").Select

Range("B2").Activate
```

Este ejemplo activa el libro Libro4.xls. Si Libro4.xls tiene varias ventanas, el ejemplo activará la primera ventana, Libro4.xls:1.

```
Workbooks("LIBRO4.XLS").Activate
```

Ejemplo del método AutoFill

Este ejemplo rellena automáticamente las celdas A1:A20 de la Hoja1, basándose en el rango de origen A1:A2 de la misma hoja. Antes de ejecutar este ejemplo, escriba 1 en la celda A1 y 2 en la celda A2.

```
Set rangoOrigen = Worksheets("Hoja1").Range("A1:A2")

Set llenarRango = Worksheets("Hoja1").Range("A1:A20")

rangoOrigen.AutoFill Destination:=llenarRango
```

NOTAS:

Ejemplo del método Clear

Este ejemplo borra las fórmulas y el formato de las celdas A1:G37 de la Hoja1.

```
Worksheets("Hoja1").Range("A1:G37").Clear
```

En este ejemplo se eliminan las celdas A1:D10 de la Hoja1 y se desplazan las celdas restantes hacia la izquierda.

```
Worksheets("Hoja1").Range("A1:D10").Delete Shift:=xlShiftToLeft
```

Ejemplo del método Delete

Este ejemplo elimina todas las hojas de cálculo del libro activo sin mostrar el cuadro de diálogo de confirmación.

```
Application.DisplayAlerts = False

For Each w In Worksheets

    w.Delete

Next w

Application.DisplayAlerts = True
```

Ejemplo del método FillDown

Este ejemplo llena el rango A1:A10 de la Hoja1 en función del contenido de la celda A1.

```
Worksheets("Hoja1").Range("A1:A10").FillDown
```

Ejemplo del método FillLeft

Este ejemplo llena el rango A1:M1 de la Hoja1 en función del contenido de la celda M1.

```
Worksheets("Hoja1").Range("A1:M1").FillLeft
```

Ejemplo del método FillRight

Este ejemplo rellena el rango A1:M1 de la Hoja1 en función del contenido de la celda A1.

```
Worksheets("Hoja1").Range("A1:M1").FillRight
```

NOTAS:

Ejemplo del método FunctionWizard

Este ejemplo ejecuta el Asistente para funciones en la celda activa de la Hoja1.

```
Worksheets("Hoja1").Activate

ActiveCell.FunctionWizard
```

Ejemplo del método Insert

Este ejemplo inserta una nueva fila antes de la fila cuatro de la Hoja1.

```
Worksheets("Hoja1").Rows(4).Insert
```

Este ejemplo inserta nuevas celdas en el rango A1:C5 de la Hoja1 y las desplaza hacia abajo.

```
Worksheets("Hoja1").Range("A1:C5").Insert shift:=xlShiftDown
```

Este ejemplo inserta una nueva fila en la celda activa.

```
ActiveCell.EntireRow.Insert
```

Ejemplo del método Replace

Este ejemplo reemplaza todas las apariciones de la función SENO por la función COS. El rango de reemplazamiento es la columna A de la Hoja1.

```
Worksheets("Hoja1").Columns("A").Replace _
    What:="SENO", Replacement:="COS", _
    SearchOrder:=xlByColumns, MatchCase:=True
```

Ejemplo del método Sort

Este ejemplo ordena el rango A1:C20 de la Hoja1, usando la celda A1 como primera clave de ordenación y la celda B1 como segunda clave. La ordenación se realiza en orden ascendente por fila, y no hay encabezados.

NOTAS:

```
Worksheets("Hoja1").Range("A1:C20").Sort _
    Key1:=Worksheets("Hoja1").Range("A1"), _
    Key2:=Worksheets("Hoja1").Range("B1")
```

SELECCIONAR Y ACTIVAR CELDAS

Al trabajar con Microsoft Excel, normalmente se selecciona una o varias celdas y, a continuación, se realiza una acción, como darles formato o escribir valores. En Visual Basic normalmente no es necesario seleccionar las celdas antes de modificarlas.

Por ejemplo, si desea escribir una fórmula en la celda D6 utilizando Visual Basic, no es necesario seleccionar el rango D6. Sólo necesita devolver el objeto Range y, a continuación, establecer la propiedad Formula en la fórmula que desee, como se muestra en el siguiente ejemplo.

```
Sub EscribirFórmula()
    Worksheets("Hoja1").Range("D6").Formula = "=SUMA(D2:D5)"
End Sub
```

Usar el método Select y la propiedad Selection

El método Select activa las hojas y los objetos de las hojas; la propiedad Selection devuelve un objeto que representa la selección actual de la hoja activa del libro activo. Antes de poder utilizar la propiedad Selection, debe activar un libro, activar o seleccionar un hoja y, a continuación, seleccionar un rango, u otro objeto, utilizando el método Select.

La grabadora de macros con frecuencia creará una macro que utilice el método Select y la propiedad Selection.

El siguiente procedimiento Sub utiliza la grabadora de macros, y muestra cómo trabajan juntas Select y Selection.

```
Sub Macro1()
    Sheets("Hoja1").Select
```

NOTAS:

```
    Range("A1").Select
    ActiveCell.FormulaR1C1 = "Nombre"
    Range("B1").Select
    ActiveCell.FormulaR1C1 = "Dirección"
    Range("A1:B1").Select
    Selection.Font.Bold = True
End Sub
```

El siguiente ejemplo realiza la misma tarea, sin activar o seleccionar la hoja de cálculo o las celdas.

```
Sub Etiquetas()
    With Worksheets("Hoja1")
        .Range("A1") = "Nombre"
        .Range("B1") = "Dirección"
        .Range("A1:B1").Font.Bold = True
    End With
End Sub
```

Seleccionar celdas en la hoja de cálculo activa

Si utiliza el método Select para seleccionar celdas, recuerde que Select sólo funciona en la hoja de cálculo activa. Si ejecuta el procedimiento Sub desde el módulo, el método Select fallará a menos que el procedimiento active la hoja de cálculo antes de utilizar el método Select en un rango de celdas. Por ejemplo, el siguiente procedimiento copia una fila de la Hoja1 a la Hoja2 del libro activo.

NOTAS:

```
Sub CopiarFilas()
    Worksheets("Hoja1").Rows(1).Copy
    Worksheets("Hoja2").Select
    Worksheets("Hoja2").Rows(1).Select
    Worksheets("Hoja2").Paste
End Sub
```

Activar una celda en una selección

Puede utilizar el método Activate para activar una celda en una selección. Sólo puede haber una celda activa, aunque se haya seleccionado un rango de celdas. El siguiente procedimiento selecciona un rango y, a continuación, activa una celda del rango sin cambiar la selección.

```
Sub Activar()
    Worksheets("Hoja1").Activate
    Range("A1:D4").Select
    Range("B2").Activate
End Sub
```

CÓMO HACER REFERENCIA A CELDAS Y RANGOS

Hacer referencia a celdas y rangos usando notación A1 usando el método Range

Referencia	Significado
Range("A1")	Celda A1

NOTAS:

Range("A1:B5")	Celdas de la A1 a la B5
Range("C5:D9,G9:H16")	Selección de varias áreas
Range("A:A")	Columna A
Range("1:1")	Fila uno
Range("A:C")	Columnas de la A a la C
Range("1:5")	Filas de la uno a la cinco
Range("1:1,3:3,8:8")	Filas uno, tres y ocho
Range("A:A,C:C,F:F")	Columnas A, C y F

El siguiente procedimiento Sub cambia el formato de las celdas A1:D5 a negrita.

```
Sub FormatoRango()

    Workbooks("Libro1").Sheets("Hoja1").Range("A1:D5") _

        .Font.Bold = True

End Sub
```

Hacer referencia a celdas usando números de índice

Puede utilizar la propiedad Cells para hacer referencia a una sola celda utilizando los números de fila y de columna. Esta propiedad devuelve un objeto Range que representa una sola celda.

En el siguiente ejemplo, Cells(6,1) devuelve la celda A6 de la Hoja1. Entonces, la propiedad Value se establece en 10.

```
Sub EscribirValor()

    Worksheets("Hoja1").Cells(6, 1).Value = 10
```

NOTAS:

```
End Sub
```

La propiedad Cells funciona bien para ejecutar bucles en un rango de celdas, ya que puede sustituir las variables por los números de índice, como se muestra en el siguiente ejemplo.

```
Sub BucleAtravés()

    Dim contador As Integer

    For contador = 1 To 20

        Worksheets("Hoja1").Cells(contador, 3).Value = contador

    Next contador

End Sub
```

Hacer referencia a filas y columnas

Utilice la propiedad Rows o Columns para trabajar con filas o columnas enteras. Estas propiedades devuelven un objeto Range que representa un rango de celdas.

Referencia	**Significado**
Rows(1)	Fila uno
Rows	Todas las filas de la hoja de cálculo
Columns(1)	Columna uno
Columns("A")	Columna uno
Columns	Todas las columnas de la hoja de cálculo

En el siguiente ejemplo, Rows(1) devuelve la fila uno de la Hoja1. A continuación, la propiedad Bold del objeto Font del rango se establece en True.

```
Sub FilaNegrita()

    Worksheets("Hoja1").Rows(1).Font.Bold = True

End Sub
```

NOTAS:

Hacer referencia a celdas usando una notación abreviada

Puede utilizar el estilo de referencia A1 o un rango con nombre entre paréntesis como método abreviado para la propiedad Range. No es necesario escribir la palabra "Range" o utilizar comillas,

Ejemplo:

```
Sub BorrarRango()

    Worksheets("Hoja1").[A1:B5].ClearContents

End Sub

Sub EstablecerValor()

    [MiRango].Value = 30

End Sub
```

Hacer referencia a rangos con nombre

Es más sencillo identificar los rangos por nombre que por la notación A1. Para asignar un nombre a un rango seleccionado, haga clic en el cuadro de nombre situado a la izquierda de la barra de fórmulas, escriba un nombre y, a continuación, presione la tecla ENTRAR.

El siguiente ejemplo hace referencia al rango denominado "MiRango" en el libro "MiLibro.xls".

```
Sub FormatoRango()

    Range("MiLibro.xls!MiRango").Font.Italic = True

End Sub
```

El siguiente ejemplo hace referencia al rango de hojas de cálculo específico denominado "Hoja1!Ventas" en el libro "MiLibro.xls".

```
Sub FormatoVentas()

    Range("[Informe.xls]Hoja1!Ventas").BorderAround weight:=xlthin
```

NOTAS:

```
End Sub
```

Para seleccionar un rango con nombre utilice el método GoTo, que activa el libro y la hoja de cálculo y, a continuación, selecciona el rango.

```
Sub BorrarRango()

    Application.Goto Reference:="MiLibro.xls!MiRango"

    Selection.ClearContents

End Sub
```

El siguiente ejemplo muestra cómo se escribiría el mismo procedimiento para el libro activo.

```
Sub BorrarRango()

    Application.Goto Reference:="MiRango"

    Selection.ClearContents

End Sub
```

Ejecutar un bucle en las celdas de un rango con nombre

El siguiente ejemplo ejecuta un bucle en cada celda del rango con nombre, utilizando el bucle For Each...Next. Si el valor de alguna celda del rango excede el valor de limit, el color de la celda cambia a amarillo.

```
Sub AplicarColor()

    Const limit As Integer = 25

    For Each c In Range("MiRango")

        If c.Value > limit Then

            c.Interior.ColorIndex = 27

        End If
```

NOTAS:

```
    Next c

End Sub
```

Hacer referencia a celdas relacionadas con otras celdas

Una manera de trabajar con una celda relacionada con otra es utilizar la propiedad Offset.

El siguiente ejemplo asigna un formato de doble subrayado al contenido de la celda situada una fila más abajo y a tres columnas de la hoja de cálculo activa.

```
Sub Subrayar()

    ActiveCell.Offset(1, 3).Font.Underline = xlDouble

End Sub
```

Hacer referencia a celdas usando un objeto Range

Si establece una variable de objeto para un objeto Range, puede manipular fácilmente el rango utilizando el nombre de la variable.

El siguiente procedimiento crea la variable de objeto miRango y, a continuación, asigna la variable al rango A1:D5 de la Hoja1 del libro activo. Las instrucciones posteriores modifican las propiedades del rango, sustituyendo el nombre de la variable por el objeto del rango.

```
Sub Aleatorio()

    Dim miRango As Range

    Set miRango = Worksheets("Hoja1").Range("A1:D5")

    miRango.Formula = "=ALEATORIO()"

    miRango.Font.Bold = True

End Sub
```

NOTAS:

Hacer referencia a todas las celdas de una hoja de cálculo

Al aplicar la propiedad Cells a una hoja de cálculo sin especificar un número de índice, el método devuelve un objeto Range que representa todas las celdas de la hoja de cálculo.

El siguiente procedimiento Sub borra el contenido de todas las celdas de la Hoja1 del libro activo.

```
Sub BorrarHoja()

    Worksheets("Hoja1").Cells.ClearContents

End Sub
```

Hacer referencia a varios rangos

Utilice los métodos Range y Union para hacer referencia a cualquier grupo de rangos;

Puede hacer referencia a varios rangos con la propiedad Range, pero debe poner comas entre dos o más referencias.

El siguiente ejemplo borra el contenido de los tres rangos de la Hoja1.

```
Sub BorrarRangos()

    Worksheets("Hoja1").Range("C5:D9,G9:H16,B14:D18").ClearContents

End Sub
```

Los rangos con nombre permiten que la propiedad Range funcione más fácilmente con varios rangos. El siguiente ejemplo funciona cuando los tres rangos con nombre están en la misma hoja.

```
Sub BorrarNombrada()

    Range("MiRango, TuRango, SuRango").ClearContents

End Sub
```

Puede combinar varios rangos en un objeto Range utilizando el método Union.

NOTAS:

El siguiente ejemplo crea un objeto Range denominado misVariosRangos, los define como A1:B2 y C3:D4 y, a continuación, asigna el formato de negrita a los rangos combinados.

```
Sub RangosVarios()
    Dim r1, r2, misVariosRangos As Range
    Set r1 = Sheets("Hoja1").Range("A1:B2")
    Set r2 = Sheets("Hoja1").Range("C3:D4")
    Set miVariosRangos = Union(r1, r2)
    miVariosRangos.Font.Bold = True
End Sub
```

OBJETO DE CONJUNTO CHARTS

Un conjunto de todas las hojas de gráficos del libro especificado o activo. Cada hoja de gráficos está representada por un objeto Chart. No se incluyen los gráficos incrustados en hojas de cálculo o de diálogoUso del conjunto Charts

Use la propiedad Charts para devolver el conjunto Charts.

El ejemplo siguiente imprime todas las hojas de gráficos del libro activo.

```
Charts.PrintOut
```

Utilice el método Add para crear una nueva hoja de gráficos y agregarla al libro.

El ejemplo siguiente agrega una nueva hoja de gráficos al libro activo y sitúa la hoja nueva inmediatamente después de la hoja de cálculo llamada "Hoja1".

```
Charts.Add after:=Worksheets("Hoja1")
```

Puede combinar el método Add con el método ChartWizard para agregar un nuevo gráfico que contenga datos de una hoja de cálculo.

NOTAS:

El ejemplo siguiente agrega un gráfico de líneas nuevo basado en las celdas A1:A20 de la hoja de cálculo llamada "Hoja1".

```
With Charts.Add
    .ChartWizard source:=Worksheets("Hoja1").Range("a1:a20"), _
    gallery:=xlLine, title:="Datos de febrero"
End With
```

Para devolver un solo objeto Chart, use Charts(índice), donde índice es el nombre o número de índice de la hoja de gráficos. El ejemplo siguiente cambia a rojo el color de la serie uno de la hoja de gráficos uno.

```
Charts(1).SeriesCollection(1).Interior.Color = RGB(255, 0, 0)
```

El conjunto Sheets contiene todas las hojas del libro (hojas de gráficos y hojas de cálculo). Para devolver un sola hoja, use Sheets(índice), donde índice es el nombre o número de la hoja.

Objeto Chart

Representa un gráfico de una hoja de cálculo. El gráfico puede estar incrustado (contenido en un objeto ChartObject) o estar en otra hoja de gráficos.

Propiedades Objeto Charts y chart

Chart	Use la propiedad Chart para devolver un objeto Chart que representa el gráfico contenido en un objeto ChartObject.
ChartArea	Devuelve un objeto ChartArea que representa al área de gráfico completa del gráfico.

NOTAS:

ActiveChart	Si un gráfico es el objeto activo, puede usar la propiedad ActiveChart para hacer referencia a él. Una hoja de gráficos está activa si el usuario la ha seleccionado o bien si se ha activado con el método Activate.
ActiveSheet	Si una hoja de gráfico es la hoja activa, puede usar la propiedad ActiveSheet para hacer referencia a ella.
ChartTitle	Devuelve un objeto ChartTitle que representa el título del gráfico especificado.
HasTitle	True si el eje o el gráfico tiene un título visible. Boolean de Lectura/Escritura. El título de un eje está representado por un objeto AxisTitle.
HasLegend	True si el gráfico tiene una leyenda. Boolean de Lectura/Escritura.
HeightPercent	Devuelve o establece el alto de un gráfico 3D como un porcentaje de su ancho (entre 5 y 500%). Long de Lectura/Escritura.
PlotArea	Devuelve un objeto PlotArea que representa el área de trazado de un gráfico.
Rotation	Devuelve o establece la rotación de la vista del gráfico 3D (la rotación del área de trazado alrededor del eje Z, en grados). El valor de esta propiedad debe estar comprendido entre 0 y 360, excepto para los gráficos de barras 3D, donde el valor debe ser de 0 a 44. El valor predeterminado es 20. Aplicable únicamente a gráficos 3D. Variant de Lectura/Escritura.
Elevation	Devuelve o establece la elevación de la vista del gráfico 3D, en grados. Long de Lectura/Escritura. La elevación del gráfico es el alto con el que se ve el gráfico, en grados. El valor predeterminado es 15. El valor de esta propiedad debe estar comprendido entre -90 y 90, excepto para los gráficos de barras 3D, en los que el valor debe estar entre 0 y 44.
Floor	Devuelve un objeto Floor que representa al plano inferior de un gráfico 3D.

NOTAS:

Perspective	Devuelve o establece la perspectiva de la vista del gráfico 3D. Debe estar comprendido entre 0 y 100.

Ejemplo del método Add

Este ejemplo crea una hoja de gráficos vacía y la inserta antes de la hoja activa.

```
ActiveWorkbook.Charts.Add Before:=Worksheets(Worksheets.Count)
```

Ejemplo de la propiedad ChartArea

Este ejemplo establece el color interior del área del Gráfico1 como rojo y el color del borde como azul.

```
With Charts("Gráfico1").ChartArea
    .Interior.ColorIndex = 3
    .Border.ColorIndex = 5
End With
```

Ejemplo de la propiedad ActiveChart

El ejemplo siguiente activa la hoja de gráficos uno y, a continuación, establece el tipo y el título del gráfico.

```
Sub Titulo()
Charts(1).Activate
With ActiveChart
  .Type = xlLine
  .HasTitle = True
  .ChartTitle.Text = "Ventas de Pemex"
End With
```

NOTAS:

Ejemplo de la Propiedad ActiveSheet

El ejemplo siguiente usa el método Activate para activar la hoja de gráficos llamada "Gráfico1" y, a continuación, establece en azul el color interior de la serie uno del gráfico.

```
Sub acti()

Charts("Gráfico13").Activate

ActiveSheet.SeriesCollection(1).Interior.ColorIndex = 5

End Sub
```

Ejemplo de la propiedad ChartTitle

Este ejemplo establece el texto del título del Gráfico1.

```
Sub Titulo()

With Charts("Gráfico1")

        .HasTitle = True

        .ChartTitle.Text = "Ventas del primer trimestre"

End With

End Sub
```

Ejemplo de la propiedad HasTitle

Este ejemplo agrega un rótulo al eje de categorías del Gráfico1.

```
Sub rotulo()

With Charts("Gráfico1").Axes(xlCategory)

        .HasTitle = True

        .AxisTitle.Text = "Ventas julio"

End With
```

NOTAS:

```
End Sub
```

Ejemplo de la propiedad HasLegend

Este ejemplo activa la leyenda del Gráfico1 y, a continuación, establece en azul el color de fuente de la leyenda.

```
Sub leyenda()

With Charts("Gráfico1")

    .HasLegend = True

    .Legend.Font.ColorIndex = 5

End With

End Sub
```

Ejemplo de la propiedad HeightPercent

Este ejemplo establece el alto del Gráfico1 en el 80% de su ancho. El ejemplo debe ejecutarse en un gráfico 3D.

```
Charts("Gráfico1").HeightPercent = 80
```

Ejemplo de la propiedad PlotArea

Este ejemplo establece en aguamarina el color del interior del área de trazado del Gráfico1.

```
Sub area()

Charts("Gráfico13").PlotArea.Border.ColorIndex = 8


End Sub
```

Ejemplo de la propiedad Rotation

Este ejemplo establece en 30 grados la rotación del Gráfico1. El ejemplo debe ejecutarse en un gráfico 3D.

NOTAS:

```
Sub rota()

    Charts("Gráfico13").Rotation = 30

End Sub
```

Ejemplo de la propiedad Elevation

Este ejemplo establece en 34 grados la elevación del Gráfico1. El ejemplo debe ejecutarse en un gráfico 3D (la propiedad Elevation no funciona en gráficos 2D).

```
Sub ele()

Charts("Gráfico13").Elevation = 34

End Sub
```

Ejemplo de la propiedad Floor

Este ejemplo establece en azul el color del plano inferior del Gráfico1. El ejemplo debe ejecutarse en un gráfico 3D (la propiedad Floor no funciona en gráficos 2D).

```
Sub inferior()

Charts("Gráfico13").Floor.Interior.ColorIndex = 5

End Sub
```

Ejemplo de la propiedad Perspective

Este ejemplo establece la perspectiva del Gráfico1 como 70. Debe ejecutarse en un gráfico 3D.

```
Sub pers()

    Charts("Gráfico13").RightAngleAxes = False

    Charts("Gráfico13").Perspective = 70

End Sub
```

NOTAS:

Métodos Objeto Charts y chart

Add	Crea una hoja de gráficos nueva. Sintaxis: expresión.Add(Before, After, Count)
Activate	Convierte el gráfico en el gráfico activo.
Axes	Devuelve un objeto que representa a un solo eje (un objeto Axis, sintaxis : expresión.Axes(Type, AxisGroup)
ChartWizard	Modifica las propiedades del gráfico especificado. Sintaxis: expresión.ChartWizard(Source, Gallery, Format, PlotBy, CategoryLabels, SeriesLabels, HasLegend, Title, CategoryTitle, ValueTitle, ExtraTitle) expresión Requerida. Una expresión que devuelve un objeto Chart. Source Variant opcional. El rango que contiene los datos de origen del nuevo gráfico. Si este argumento se omite, Microsoft Excel modificará la hoja de gráficos activa o el gráfico seleccionado en la hoja de cálculo activa. Gallery Variant opcional. El tipo del gráfico. Puede ser una de las siguientes constantes XlChartType: xlArea, xlBar, xlColumn, xlLine, xlPie, xlRadar, xlXYScatter, xlCombination, xl3DArea, xl3DBar, xl3DColumn, xl3DLine, xl3DPie, xl3DSurface, xlDoughnut o xlDefaultAutoFormat. Format Variant opcional. El número de opción de los autoformatos incorporados. Puede ser un número del 1 al 10, dependiendo del tipo de la galería. Si este argumento se omite, Microsoft Excel elegirá un valor predeterminado basándose en el tipo de la galería y en el origen de los datos. PlotBy Variant opcional. Especifica si los datos de cada serie se ubican en filas o en columnas. Puede ser una de las siguientes constantes XlRowCol: xlRows o xlColumns. CategoryLabels Variant opcional. Un número entero que especifica el número de filas o de columnas del rango de origen que contienen rótulos de categorías. Los valores permitidos van desde cero hasta

NOTAS:

	uno menos que el número máximo de las categorías o series correspondientes. SeriesLabels Variant opcional. Un número entero que especifica el número de filas o de columnas del rango de origen que contienen rótulos de series. Los valores permitidos van desde cero hasta uno menos que el número máximo de las categorías o series correspondientes. HasLegend Variant opcional. True si se incluye una leyenda. Title Variant opcional. El texto del título del gráfico CategoryTitle Variant opcional. El texto del título del eje de categorías. ValueTitle Variant opcional. El texto del título del eje de valores. ExtraTitle Variant opcional. El título del eje de series en gráficos 3D o el segundo título del eje de valores en gráficos 2D.

Ejemplo del método Add

Este ejemplo crea una hoja de gráficos vacía y la inserta antes de la hoja activa.

```
ActiveWorkbook.Charts.Add Before:=Worksheets(Worksheets.Count)
```

Ejemplo del método Axes

Este ejemplo agrega un rótulo al eje de categorías del Gráfico1.

```
Sub agrega()

With Charts("Gráfico1").Axes(xlCategory)

        .HasTitle = True

        .AxisTitle.Text = "Ventas julio"
```

NOTAS:

```
End With

End Sub
```

Ejemplo del método ChartWizard

Este ejemplo cambia el formato del Gráfico1 a un gráfico de líneas, agrega una leyenda y después títulos a los ejes de categorías y valores.

```
Charts("Gráfico1").ChartWizard _

    Gallery:=xlLine, _

    HasLegend:=True, CategoryTitle:="Año", ValueTitle:="Ventas"
```

CREAR UNA MACRO USANDO EL EDITOR DE VISUAL BASIC

Crear un nuevo libro

```
Sub nuevo( )

  Workbooks.Add

End Sub
```

Abrir un libro

```
Sub abrir()

    Workbooks.Open FileName:="C:\Mis documentos\abrir.xls"

End Sub
```

NOTAS:

Este ejemplo abre el libro Análisis.xls

```
Sub Abrir()

    Workbooks.Open "ANÁLISIS.XLS"

End Sub
```

Cerrar un libro

```
Sub cerrar()

    ActiveWorkbook.Close
End Sub
```

Este ejemplo cierra Libro1.xls y descarta los cambios efectuados en él.

```
Workbooks("LIBRO1.XLS").Close SaveChanges:=False
```

Este ejemplo cierra todos los libros abiertos. Si hay cambios en algún libro abierto, Microsoft Excel mostrará los mensajes y cuadros de diálogo oportunos para preguntar si desea guardar los cambios.

```
    Workbooks.Close
```

Guardar un libro

```
Sub guardar()
ActiveWorkbook.SaveAs FileName:="C:\Mis documentos\guardar.xls", FileFormat _
  :=xlNormal, Password:="", WriteResPassword:="", ReadOnlyRecommended:= _
        False, CreateBackup:=False
    End Sub
```

Guardar Libro Activo

NOTAS:

```
Sub guardar( )

ActiveWorkbook.Save

End Sub
```

Crear Hoja

```
Sub crear_hoja( )

        Sheets.Add

End Sub
```

Este ejemplo inserta una hoja de cálculo nueva antes de la última hoja de cálculo del libro activo.

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

Seleccionar Hoja

```
Sub selec_hoja( )

Sheets("Hoja1").Select

End Sub
```

Eliminar Hoja

```
Sub eliminar( )

ActiveWindow.SelectedSheets.Delete

End Sub
```

NOTAS:

PRACTICA

1. Crear una macro que inserte un gráfico circular de 3d que muestre los nombres de 5 alumnos y las calificaciones correspondiente al curso de Excel Avanzado,

2. Crear una macro que inserte los gráficos circulares de 2d que muestre los nombres de 5 alumnos y las calificaciones correspondiente al curso de Power Point

3. Crear una macro que inserte los gráficos circulares de 2d que muestre los nombres de 5 alumnos y las calificaciones correspondiente al curso de Power Point

Por ejemplo:

Nombre	Word	Excel	Power	Calificacion
Pedro		10	9	8 9.0
Juan		8	10	8 8.7
Luis		7	6	5 6.0
Marcos		8	7	9 8.0

NOTAS:

CAPITULO 5. VARIABLES

DECLARAR VARIABLES

Definición de variable

Un lugar de almacenamiento con nombre que puede contener cierto tipo de datos que puede ser modificado durante la ejecución del programa. Cada variable tiene un nombre único que la identifica dentro de su nivel de ámbito. Puede especificar un tipo de datos o no.

Nombres de variable deben comenzar con un carácter alfabético, deben ser únicos dentro del mismo ámbito, no deben contener más de 255 caracteres y no pueden contener un punto o carácter de declaración de tipo.

Declaración de 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.

nivel de procedimiento

Instrucciones localizadas dentro de los procedimientos Function, Property o Sub. Generalmente, las declaraciones aparecen primero, seguidas de asignaciones y otro código ejecutable.

Módulo: Un conjunto de declaraciones y procedimientos.

nivel de módulo

Código en la sección de declaraciones de un módulo. Cualquier código fuera de un procedimiento se denomina código de nivel de módulo. Las declaraciones se deben colocar primero, seguidas de los procedimientos.

NOTAS:

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.

Tipos de Datos

Característica de una variable que determina qué tipo de datos puede tener. Los tipos de datos incluyen Byte, Boolean, Integer, Long, Currency, Single, Double, Date, String, Object, Variant (predeterminado) y tipos definidos por el usuario, así como tipos específicos de objetos.

String (Texto)

Tipo de datos que consiste en una secuencia de caracteres que representa a los caracteres por sí mismos en vez de sus valores numéricos. Un tipo String puede incluir letras, números, espacios en blanco y signos de puntuación. El tipo de datos String puede almacenar cadenas de longitud fija en un intervalo de 0 a aproximadamente 63000 caracteres y cadenas dinámicas en un intervalo de longitud de 0 a aproximadamente 2 mil millones de caracteres. El carácter de declaración de tipo es el signo de dólar ($) que representa el tipo String en Visual Basic.

Boolean

Las variables tipo Boolean se almacenan como números de 16 bits (2 bytes), pero sólo pueden ser True o False. Las variables tipo Boolean se presentan como True o False

(cuando se utiliza Print) o #TRUE# o #FALSE# (cuando se utiliza Write #). Utilice las palabras clave True y False para asignar uno de los dos estados a las variables tipo Boolean.

Cuando se convierten a tipo Boolean otros tipos numéricos, 0 se convierte en False, y el resto de los valores se convierten en True. Cuando los valores tipo Boolean se convierten a otros tipos de datos numéricos, False se convierta en 0 y True se convierte en -1.

Byte

Las variables tipo Byte se almacenan como números de 8 bits (1 byte) sencillos sin signo con un intervalo de valores entre 0 y 225.

El tipo de dato Byte es útil para almacenar datos binarios.

NOTAS:

Variant

El tipo de datos Variant se especifica automáticamente si no se especifica otro tipo de datos al declarar una constante, variable, o argumento. Las variables declaradas como del tipo de datos Variant pueden contener valores numéricos, cadenas de texto, fecha, hora o Booleans y pueden convertir los valores que contienen de forma automática. Los valores numéricos Variant ocupan 16 bytes de memoria (lo que sólo es significativo en procedimientos grandes o módulos complejos) y son más lentos a la hora de su acceso que las variables de tipo explícito de los restantes tipos. Es muy raro utilizar el tipo de datos Variant para una constante. Los valores de cadena Variant necesitan 22 bytes de memoria.

Currency

Las variables tipo Currency se almacenan como números de 64 bits (8 bytes) en un formato de número entero a escala de 10.000 para dar un número de punto fijo con 15 dígitos a la izquierda del signo decimal y 4 dígitos a la derecha. Esta representación proporciona un intervalo de -922.337.203.685.477,5808 a 922.337.203.685.477,5807. El carácter de declaración de tipo para Currency es el signo @.

El tipo de datos Currency es útil para cálculos monetarios y para cálculos de punto fijo, en los cuales la precisión es especialmente importante.

Date

Las variables tipo Date se almacenan como números IEEE de signo flotante de 64 bits (8 bytes) que van del 1 de enero del 100 al 31 de diciembre de 9999 y horarios de 0:00:00 a 23:59:59. Cualquier valor reconocible de fecha literal se puede asignar a las variables tipo Date. Los literales de fechas se deben poner entre caracteres de signo de número (#). Por ejemplo, #1 Enero, 1993# o #1 Ene 93#.

Las variables tipo Date presentan fechas de acuerdo al formato de fecha corto reconocido por su sistema. La hora se presenta de acuerdo al formato de hora reconocido por su sistema (12 ó 24 horas).

Cuando se convierten a tipo Date otros tipos de datos numéricos, los valores a la izquierda del signo decimal representan la información de fecha, mientras que los valores a la derecha del signo decimal representan la hora. Medianoche es 0 y mediodía es 0,5. Los números enteros negativos representan fechas anteriores al 30 de diciembre de 1899.

NOTAS:

Double

Las variables tipo Double (coma flotante y precisión doble) se almacenan como números IEEE de coma flotante de 64 bits (8 bytes) con valores de -1,79769313486232E308 a -4,94065645841247E-324 para valores negativos y de 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos. El carácter de declaración de tipo para Double es el signo de número (#).

Integer

Las variables tipo Integer se almacenan como números de 16 bits (2 bytes) con valores que van de -32.768 a 32.767. El carácter de declaración de tipo para el tipo Integer es el signo de porcentaje (%).

Las variables tipo Integer también se pueden utilizar para representar valores enumerados. Un valor enumerado puede contener un conjunto finito de números enteros únicos, cada uno de los cuales tiene un significado especial en el contexto en el que se utiliza. Los valores enumerados proporcionan una forma cómoda de seleccionar entre un número conocido de opciones. Por ejemplo, cuando se pregunta al usuario que elija un color de una lista, se podría tener 0 = negro, 1 = blanco y así sucesivamente. Es una buena práctica de programación definir constantes utilizando la instrucción Const para cada valor enumerado.

Long

Las variables tipo Long (entero largo) se almacenan como números con signo de 32 bits (4 bytes) con un valor comprendido entre - 2.147.483.648 y 2.147.483.647. El carácter de declaración de tipo para Long es el signo &.

Object

Las variables tipo Object se almacenan como direcciones de 32 bits (4 bytes) que hacen referencia a objetos. Al utilizar la instrucción Set, una variable declarada con tipo Object puede tener asignado cualquier referencia a un objeto.

Single

Las variables tipo Single (coma flotante y precisión simple) se almacenan como números IEEE de coma flotante de 32 bits (4 bytes) con valores que van de -3,402823E38 a -1,401298E-45 para valores negativos y de 1,401298E-45 a 3,402823E38 para valores positivos. El carácter de declaración de tipo para Single es el signo de exclamación (!).

NOTAS:

Utilizar la instrucción Public

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

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

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 erroneos.

NOTAS:

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. Consulte la documentación de su aplicación para encontrar la forma de modificar las opciones de entorno de Visual Basic. Tenga en cuenta que esta opción no tiene ningún efecto sobre el código que se haya escrito con anterioridad.

Nota: Las matrices fijas y dinámicas siempre se tiene que declarar explícitamente.

Declarar una variable de objeto para automatización

Cuando se utiliza una aplicación para controlar los objetos de otra aplicación, debe establecerse una referencia a la biblioteca de tipos de la otra aplicación. Una vez que se ha establecido la referencia, se pueden declarar variables de objeto conforme a su tipo más específico. Por ejemplo, si desde Microsoft Access se establece una referencia a la biblioteca de tipos de Microsoft Excel, se puede declarar una variable del tipo Worksheet desde Microsoft Access para representar un objeto Worksheet de Microsoft Excel.

NOTAS:

CAPÍTULO 6. CONDICIONES

UTILIZAR INSTRUCCIONES WITH

La instrucción With permite especificar una vez un objeto o tipo definido por el usuario en una serie entera de instrucciones. Las instrucciones With aceleran la ejecución de los procedimientos y ayudan a evitar el tener que escribir repetidas veces las mismas palabras.

El siguiente ejemplo introduce en un rango de celdas el número 30, aplica a esas celdas un formato en negrita y hace que su color de fondo sea el amarillo.

Sintaxis

With objeto

[instrucciones]

End With

La sintaxis de la instrucción With consta de las siguientes partes:

Objeto:	Nombre de un objeto o de un tipo definido por el usuario.
Instrucciones:	(Opcional) Una o más instrucciones que se van a ejecutar sobre objeto.

```
Sub RangoFormato()

    With Worksheets("Hoja1").Range("A1:C10")

        .Value = 30

        .Font.Bold = True

        .Interior.Color = RGB(255, 255, 0)

    End With
```

NOTAS:

```
End Sub
```

Las instrucciones With se pueden anidar para aumentar su eficiencia. El siguiente ejemplo inserta una formula en la celda A1 y selecciona a continuación el tipo de letra.

```
Sub MiEntrada()
    With Workbooks("Libro1").Worksheets("Hoja1").Cells(1, 1)
        .Formula = "=SQRT(50)"
        With .Font
            .Name = "Arial"
            .Bold = True
            .Size = 8
        End With
    End With
End Sub
```

UTILIZAR INSTRUCCIONES IF...THEN...ELSE

Se puede usar la instrucción If...Then...Else para ejecutar una instrucción o bloque de instrucciones determinadas, dependiendo del valor de una condición. Las instrucciones If...Then...Else se pueden anidar en tantos niveles como sea necesario. Sin embargo, para hacer más legible el código es aconsejable utilizar una instrucción Select Case en vez de recurrir a múltiples niveles de instrucciones If...Then...Else anidadas.

Sintaxis

If condición Then [instrucciones]-[Else instrucciones_else]

NOTAS:

Puede utilizar la siguiente sintaxis en formato de bloque:

If condición Then

[instrucciones]

[ElseIf condición-n Then

[instrucciones_elseif] ...

[Else

[instrucciones_else]]

End If

La sintaxis de la instrucción If...Then...Else consta de tres partes:

Condición:	Uno o más de los siguientes dos tipos de expresiones:numérica o del formulario
Instrucciones:	(Opcional). en formato de bloque; se requiere en formato de línea sencilla que no tenga una cláusula Else. Una o más instrucciones separadas por dos puntos ejecutados si la condición es True.
Condición n	(Opcional). Igual que condición.
Instrucciones_elseif	(Opcional).Una o más instrucciones ejecutadas si lacondición-n asociada es True.
instrucciones_else	(Opcional) Una o más instrucciones ejecutadas si ninguna de las expresiones anteriores condición o condición-n es True.

NOTAS:

Ejecutar una sola instrucción cuando una condición es True

Para ejecutar una sola instrucción cuando una condición es True, se puede usar la sintaxis de línea única de la instrucción If...Then...Else. El siguiente ejemplo muestra la sintaxis de línea única, en la que se omite el uso de la palabra clave Else:

```
Sub FijarFecha()
    miFecha = #13/2/95#
    If miFecha < Now Then miFecha = Now
End Sub
```

Para ejecutar más de una línea de código, es preciso utilizar la sintaxis de múltiples líneas. Esta sintaxis incluye la instrucción End If, tal y como muestra el siguiente ejemplo:

```
Sub AvisoUsuario(valor as Long)
    If valor = 0 Then
        Aviso.ForeColor = "Red"
        Aviso.Font.Bold = True
        Aviso.Font.Italic = True
    End If
End Sub
```

Ejecutar unas instrucciones determinadas si una condición es True y ejecutar otras si es False

Use una instrucción If...Then...Else para definir dos bloques de instrucciones ejecutables: un bloque que se ejecutará cuando la condición es True y el otro que se ejecutará si la condición es False.

NOTAS:

```
Sub AvisoUsuario(valor as Long)
    If valor = 0 Then
        Aviso.ForeColor = vbRed
        Aviso.Font.Bold = True
        Aviso.Font.Italic = True
    Else
        Aviso.Forecolor = vbBlack
        Aviso.Font.Bold = False
        Aviso.Font.Italic = False
    End If
End Sub
```

Comprobar una segunda condición si la primera condición es False

Se pueden añadir instrucciones ElseIf a una instrucción If...Then...Else para comprobar una segunda condición si la primera es False. Por ejemplo, el siguiente procedimiento función calcula una bonificación salarial dependiendo de la clasificación del trabajador. La instrucción que sigue a la instrucción Else sólo se ejecuta cuando las condiciones de todas las restantes instrucciones If y ElseIf son False.

```
Function Bonificación(rendimiento, salario)
    If rendimiento = 1 Then
        Bonificación = salario * 0.1
```

NOTAS:

```
        Elself rendimiento = 2 Then
                Bonificación= salario * 0.09
        Elself rendimiento = 3 Then
                Bonificación = salario * 0.07
        Else
                Bonificación = 0
        End If
End Function
```

UTILIZAR INSTRUCCIONES DO...LOOP

Se pueden usar instrucciones Do...Loop para ejecutar un bloque de instrucciones un número indefinido de veces. Las instrucciones se repiten mientras una condición sea True o hasta que llegue a ser True.

sintaxis

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

O bien, puede utilizar esta sintaxis:

```
Do
```

NOTAS:

[instrucciones]

[Exit Do]

[instrucciones]

Loop [{While | Until} condición]

La sintaxis de la instrucción Do Loop consta de las siguientes partes:

Condición:	(Opcional) Expresión numérica o expresión de cadena que es True o False. Si la condición es Null, condición se considera False.
Instrucciones:	Una o más instrucciones que se repiten mientras o hasta que condición sea True.

Repetir instrucciones mientras una condición es True

Hay dos formas de utilizar la palabra clave While para comprobar el estado de una condición en una instrucción Do...Loop. Se puede comprobar la condición antes de entrar en el bucle, o después de que el bucle se haya ejecutado al menos una vez.

En el siguiente procedimiento ComPrimeroWhile, la condición se comprueba antes de entrar en el bucle. Si miNum vale 9 en vez de 20, las instrucciones contenidas en el bucle no se ejecutarán nunca. En el procedimiento ComFinalWhile, las instrucciones contenidas en el bucle *sólo se ejecutarán una vez antes de que la condición llegue a ser False.*

```
Sub ComPrimeroWhile()

    contador = 0

    miNum = 20

    Do While miNum > 10

        miNum = miNum - 1

        contador = contador + 1

    Loop
```

NOTAS:

```
        MsgBox "El bucle se ha repetido " & contador & " veces."
End Sub

Sub ComFinalWhile()
        contador = 0
        miNum = 9
        Do
                miNum = miNum - 1
                contador = contador + 1
        Loop While miNum > 10
        MsgBox "El bucle se ha repetido " & contador & " veces."
End Sub
```

Repetir instrucciones hasta que una condición llegue a ser True

Hay dos formas de utilizar la palabra clave Until para comprobar el estado de una condición en una instrucción Do...Loop. Se puede comprobar la condición antes de entrar en el bucle (como muestra el procedimiento ComPrimeroUntil) o se pueden comprobar después de que el bucle se haya ejecutado al menos una vez (como muestra el procedimiento ComFinalUntil). El bucle sigue ejecutándose mientras la condición siga siendo False.

```
Sub ComPrimeroUntil()
        contador = 0
        miNum = 20
        Do Until miNum = 10
```

NOTAS:

```
            miNum = miNum - 1
            contador = contador + 1
        Loop
        MsgBox "El bucle se ha repetido " & contador & " veces."
End Sub

Sub ComFinalUntil()
        contador = 0
        miNum = 1
        Do
            miNum = miNum + 1
            contador = contador + 1
        Loop Until miNum = 10
        MsgBox "El bucle se ha repetido " & counter & " veces."
End Sub
```

UTILIZAR INSTRUCCIONES FOR EACH...NEXT

Las instrucciones For Each...Next repiten un bloque de instrucciones para cada uno de los objetos de una colección o para cada elemento de una matriz. Visual Basic asigna valor automáticamente a una variable cada vez que se ejecuta el bucle.

Sintaxis

```
For Each elemento In grupo
[instrucciones]
```

NOTAS:

[Exit For]

[instrucciones]

Next [elemento]

La sintaxis de la instrucción For Each...Next consta de las siguientes partes:

Elemento:	Variable que se utiliza para iterar por los elementos del conjunto o matriz. Para conjuntos, elemento solamente puede ser una variable Variant, una variable de objeto genérica o cualquier variable de objeto específica. Para matrices, elemento solamente puede ser una variable tipo Variant.
Grupo	Nombre de un conjunto de objetos o de una matriz (excepto una matriz de tipos definidos por el usuario).
Instrucciones:	Opcional) Una o más instrucciones que se ejecutan para cada elemento de un grupo.

Por ejemplo, el siguiente procedimiento cierra todos los formularios excepto el que contiene al procedimiento que se está ejecutando.

```
Sub CierraFormul()
    For Each frm In Application.Forms
        If frm.Caption <> Screen.ActiveForm.Caption Then frm.Close
    Next
End Sub
```

El siguiente código recorre todos los elementos de una matriz e introduce en cada uno de ellos el valor de la variable índice I.

NOTAS:

```
Dim PruebaMatriz(10) As Integer, I As Variant

For Each I In PruebaMatriz

    PruebaMatriz(I) = I

Next I
```

Recorrer un conjunto de celdas

Se puede usar el bucle For Each...Next para recorrer las celdas pertenecientes a un rango determinado. El siguiente procedimiento recorre las celdas del rango A1:D10 de la Página1 y convierte cualquier valor absoluto menor de 0,01 en 0 (cero).

```
Sub RedondeoACero()

    For Each miObjeto in miColeccion

        If Abs(miObjeto.Value) < 0.01 Then miObjeto.Value = 0

    Next

End Sub
```

Salir de un bucle For Each...Next antes de que finalice

Se puede salir de un bucle For Each...Next mediante la instrucción Exit For. Por ejemplo, cuando se produce un error se puede usar la instrucción Exit For en el bloque de instrucciones True de una instrucción If...Then...Else o Select Case que detecte específicamente el error. Si el error no se produce, la instrucción If...Then...Else es False y el bucle se seguirá ejecutando normalmente.

NOTAS:

El siguiente ejemplo detecta la primera celda del rango A1:B5 que no contiene un número. Si se encuentra una celda en esas condiciones, se presenta un mensaje en pantalla y Exit For abandona el bucle.

```
Sub BuscaNumeros()
    For Each miObjeto In MiColeccion
        If IsNumeric(miObjeto.Value) = False Then
            MsgBox "El objeto contiene un valor no numérico."
            Exit For
        End If
    Next c
End Sub
```

UTILIZAR INSTRUCCIONES FOR...NEXT

Las instrucciones For...Next se pueden utilizar para repetir un bloque de instrucciones

un número determinado de veces. Los bucles For usan una variable contador cuyo valor se aumenta o disminuye cada vez que se ejecuta el bucle.

Sintaxis

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

NOTAS:

Next [contador]

La sintaxis de la instrucción For...Next consta de las siguientes partes:

Parte	Descripción
Contador:	Variable numérica que se utiliza como contador de bucle. La variable no puede ser de tipo Boolean, ni ningún elemento de matriz.
Principio:	Valor inicial del contador.
Fin:	Valor final del contador.
Incremento:	(Opcional) Cantidad en la que cambia el contador cada vez que se ejecuta el bucle. Si no se especifica, el valor predeterminado de incremento es uno.
Instrucciones:	(Opcional). Una o más instrucciones entre For y Next que se ejecutan un número especificado de veces.

El siguiente procedimiento hace que el equipo emita un sonido 50 veces. La instrucción For determina la variable contador x y sus valores inicial y final. La instrucción Next incrementa el valor de la variable contador en 1.

```
Sub Bips()
    For x = 1 To 50
        Beep
    Next x
End Sub
```

Mediante la palabra clave Step, se puede aumentar o disminuir la variable contador en el valor que se desee. En el siguiente ejemplo, la variable contador j se incrementa en 2 cada vez

NOTAS:

que se repite la ejecución del bucle. Cuando el bucle deja de ejecutarse, total representa la suma de 2, 4, 6, 8 y 10.

```
Sub DosTotal()
    For j = 2 To 10 Step 2
        total = total + j
    Next j
    MsgBox "El total es " & total
End Sub
```

Para disminuir la variable contador utilice un valor negativo en Step. Para disminuir la variable contador es preciso especificar un valor final que sea menor que el valor inicial. En el siguiente ejemplo, la variable contador miNum se disminuye en 2 cada vez que se repite el bucle. Cuando termina la ejecución del bucle, total representa la suma de 16, 14, 12, 10, 8, 6, 4 y 2.

```
Sub NuevoTotal()
    For miNum = 16 To 2 Step -2
        total = total + miNum
    Next miNum
    MsgBox "El total es " & total
End Sub
```

UTILIZAR INSTRUCCIONES WHILE...WEND

Ejecuta una serie de intrucciones mientras una condición dada sea True.

Sintaxis

NOTAS:

While condición

[intrucciones]

Wend

La sintaxis de la instrucción While...Wend consta de las siguientes partes:

Parte	Descripción
condición	Requerido. Expresión numérica o expresión de cadena cuyo valor es True o False. Si condición es Null, condición se considera False.
Instrucciones	Opcional. Una o más instrucciones que se ejecutan mientras la condición es True.

Si condición es True, todas las instrucciones se ejecutan hasta que se encuentra la instrucción Wend. Después, el control vuelve a la instrucción While y se comprueba de nuevo condición. Si condición es aún True, se repite el proceso. Si no es True, la ejecución se reanuda con la instrucción que sigue a la instrucción Wend.

En este ejemplo se utiliza la instrucción While...Wend para incrementar una variable de contador. Las instrucciones del bucle se ejecutan mientras la condición sea True.

```
Dim Contador

Contador = 0  ' Inicializa la variable.

While Contador < 20  ' Comprueba el valor del Contador.

        Contador = Contador + 1      ' Incrementa Contador.

Wend  ' Finaliza el bucle End While cuando Contador > 19.

Debug.Print Contador          ' Imprime 20 en la ventana Depuración.
```

NOTAS:

CAPÍTULO 7. FUNCIONES

INPUTBOX

Muestra un mensaje en un cuadro de diálogo, espera que el usuario escriba un texto o haga clic en un botón y devuelve un tipo String con el contenido del cuadro de texto.

Sintaxis

InputBox(prompt[, title][, default][, xpos][, ypos][, helpfile, context])

La sintaxis de la función InputBox consta de estos argumentos con nombre:

prompt

Expresión de cadena 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 separarlos utilizando un carácter de retorno de carro (Chr(13)), un carácter de avance de línea (Chr(10)) o una combinación de los caracteres de retorno de carro-avance de línea (Chr(13) y Chr(10)) entre cada línea y la siguiente.

Title (Opcional).

Expresión de cadena que se muestra en la barra de título del cuadro de diálogo. Si omite title, en la barra de título se coloca el nombre de la aplicación.

Default(Opcional).

Expresión de cadena que se muestra en el cuadro de texto como respuesta predeterminada cuando no se suministra una cadena. Si omite default, se muestra el cuadro de texto vacío.

xpos (Opcional).

NOTAS:

Expresión numérica que especifica, en twips, la distancia en sentido horizontal entre el borde izquierdo del cuadro de diálogo y el borde izquierdo de la pantalla. Si se omite xpos, el cuadro de diálogo se centra horizontalmente.

ypos *(*Opcional).

Expresión numérica que especifica, en twips, la distancia en sentido vertical entre el borde superior del cuadro de diálogo y el borde superior de la pantalla. Si se omite ypos, el cuadro de diálogo se coloca a aproximadamente un tercio de la altura de la pantalla, desde el borde superior de la misma.

Helpfile *(*Opcional).

Expresión de cadena que identifica el archivo de Ayuda que se utilizará para proporcionar ayuda interactiva para el cuadro de diálogo. Si se especifica helpfile, también deberá especificarse context.

Context *(*Opcional).

Expresión numérica que es el número de contexto de Ayuda asignado por el autor al tema de Ayuda correspondiente. Si se especifica context, también deberá especificarse helpfile.

En este ejemplo se muestran distintas maneras de utilizar la función InputBox para indicar al usuario que debe introducir un valor. Si se omiten las posiciones x e y, el diálogo se centra automáticamente según los ejes respectivos. La variable MyValue contiene el valor introducido por el usuario, si éste elige Aceptar o presiona ENTRAR. Si el usuario elige Cancelar, se devuelve una cadena de caracteres de longitud cero.

```
Dim Mensaje, Título, ValorPred, MiValor

Mensaje = " Introduzca un número del 1 a 3"        ' Establece el mensaje.

Titulo = "Demostración de InputBox"        ' Establece el título.

ValorPred = "1"        ' Establece el valor predeterminado.

' Muestra el mensaje, el título, y el valor predeterminado.

MiValor = InputBox(Mensaje, Título, ValorPred)
```

NOTAS:

' Muestra el mensaje, el título y el valor predeterminado.

MiValor = InputBox(Mensaje, Título, , , , "DEMO.HLP", 10)

' Se muestra el diálogo en la posición 100, 100.

MiValor = InputBox(Mensaje, Título, ValorPred, 100, 100)

MsgBox

Muestra un mensaje en un cuadro de diálogo, espera a que el usuario haga clic en un botón y devuelve un tipo Integer correspondiente al botón elegido por el usuario.

Sintaxis

MsgBox(prompt[, buttons][, title][, helpfile, context])

La sintaxis de la función MsgBox consta de estos argumentos con nombre:

prompt	Expresión de cadena que representa el prompt 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 separarlos utilizando un carácter de retorno de carro (Chr(13)) o 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 y la siguiente.
buttons	Opcional. Expresión numérica que corresponde a la suma de los valores que especifican el número y el tipo de los botones que se pretenden mostrar, el estilo de icono que se va a utilizar, la identidad del botón predeterminado y la modalidad del cuadro de mensajes. Si

NOTAS:

	se omite este argumento, el valor predeterminado para buttons es 0..
title	Opcional. Expresión de cadena que se muestra en la barra de título del cuadro de diálogo. Si se omite title, en la barra de título se coloca el nombre de la aplicación.
helpfile	Opcional. Expresión de cadena que identifica el archivo de Ayuda que se utiliza para proporcionar ayuda interactiva en el cuadro de diálogo. Si se especifica helpfile, también se debe especificar context.
context	Opcional. Expresión numérica que es igual al número de contexto de Ayuda asignado por el autor al tema de Ayuda correspondiente. Si se especifica context, también se debe especificar helpfile.

El argumento buttons tiene estos valores:

Constante	Valor	Descripción
vbOKOnly	0	Muestra solamente el botón Aceptar.
VbOKCancel	1	Muestra los botones Aceptar y Cancelar.
VbAbortRetryIgnore	2	Muestra los botones Anular, Reintentar e Ignorar.
VbYesNoCancel	3	Muestra los botones Sí, No y Cancelar.
VbYesNo	4	Muestra los botones Sí y No.
VbRetryCancel	5	Muestra los botones Reintentar y Cancelar.
VbCritical	16	Muestra el icono de mensaje crítico.
VbQuestion	32	Muestra el icono de pregunta de advertencia.
VbExclamation	48	Muestra el icono de mensaje de advertencia.

NOTAS:

VbInformation	64	Muestra el icono de mensaje de información.
VbDefaultButton1	0	El primer botón es el predeterminado.
VbDefaultButton2	256	El segundo botón es el predeterminado.
VbDefaultButton3	512	El tercer botón es el predeterminado.
VbDefaultButton4	768	El cuarto botón es el predeterminado.
VbApplicationModal	0	Aplicación modal; el usuario debe responder al cuadro de mensajes antes de poder seguir trabajando en la aplicación actual.
VbSystemModal	4096	Sistema modal; se suspenden todas las aplicaciones hasta que el usuario responda al cuadro de mensajes.

Estas constantes las especifica Visual Basic for Applications. Por tanto, el nombre de las mismas puede utilizarse en cualquier lugar del código en vez de sus valores reales.

Valores devueltos

Constante	**Valor**	**Descripción**
vbOK	1	Aceptar
vbCancel	2	Cancelar
vbAbort	3	Anular
vbRetry	4	Reintentar
vbIgnore	5	Ignorar
vbYes	6	Sí

NOTAS:

vbNo	7	No

En este ejemplo se utiliza la función MsgBox para mostrar un mensaje de error crítico en un cuadro de diálogo con botones Sí y No. El botón No se considera la respuesta predeterminada. El valor devuelto por la función MsgBox depende del botón elegido por el usuario. En este ejemplo, se supone que DEMO.HLP es un archivo de Ayuda que contiene un tema con un número de contexto igual a 1000.

```
Dim Mensaje, Estilo, Título, Ayuda, Ctxt, Respuesta, MiCadena

Mensaje = "¿Desea continuar?"      ' Define el mensaje.

Estilo = vbYesNo + vbCritical + vbDefaultButton2   ' Define los botones.

Título = "Demostración de MsgBox" ' Define el título.

Ayuda = "DEMO.HLP"          ' Define el archivo de ayuda.

Ctxt = 1000    ' Define el tema

                            ' el contexto

                            ' Muestra el mensaje.

Respuesta = MsgBox(Mensaje, Estilo, Título, Ayuda, Ctxt)


If Respuesta = vbYes Then   ' El usuario eligió el botón Sí.

        MiCadena = "Sí"      ' Ejecuta una acción.

Else    ' El usuario eligió el botón No.

        MiCadena = "No"      ' Ejecuta una acción.

End If
```

NOTAS:

USO DE FUNCIONES DE HOJA DE CÁLCULO DE MICROSOFT EXCEL EN VISUAL BASIC

Lista de funciones para hojas de cálculo en Visual Basic

Acosh	Acos	Asenoh	Aseno	Atan2	Atanh
BdContar	BdDesvEst	BdDesvEt	BdExtraer	BdMx	BdMín
BdPromedio	BdSuma	BdVarP	BdVar	VA	BuscarH
Buscar	ContarSí	Cosh	Covarianza	Cuartil	Cuenta
ContarBlanco	Ddb	Decimal	DesvEstP	DesvEst	Desvia2
Db	DíaSem	DistrBetaInv	DistrBeta	Var	DistrBinom
Días360	Log10	Log	Máx	Mediana	Mín
Fact	NomPropio	NPer	PagoInt	PagoPrin	Pago
Moneda	Promedio	Redondear	Sln	Suma	Tasa
Producto	VF	BdProducto	BuscarV	Moda	Pi

Llamar a una función de hoja de cálculo desde Visual Basic

En Visual Basic, las funciones de hoja de calculo de Microsoft Excel pueden ejecutarse mediante el objeto WorksheetFunction.

El siguiente procedimiento Sub usa la función Mín para obtener el valor más pequeño de un rango de celdas. En primer lugar, se declara la variable miRango como un objeto Range y, a continuación, se establece como el rango A1:C10 de la Hoja1. Otra variable, respuesta, se

NOTAS:

asigna al resultado de aplicar la función Mín a miRango. Por último, el valor de respuesta se muestra en un cuadro de mensaje.

```
Sub UseFunction()

    Dim miRango As Range

    Set miRango = Worksheets("Hoja1").Range("A1:C10")

    respuesta = Application.WorksheetFunction.Min(miRango)

    MsgBox respuesta

End Sub
```

Insertar una función de hoja de cálculo en una celda

Para insertar una función de hoja de cálculo en una celda, especifique la función como el valor de la propiedad Formula del objeto Range correspondiente. En el siguiente ejemplo, la función ALEATORIO (que genera un número aleatorio) se asigna a la propiedad Formula del rango A1:B3 de la Hoja1 del libro activo.

```
Sub InsertFormula()

    Worksheets("Hoja1").Range("A1:B3").Formula = "=ALEATORIO()"

End Sub
```

Ejemplo del uso de funciones de hoja de cálculo en Visual Basic

Este ejemplo usa la función de hoja de cálculo Pago para calcular un pago de préstamo hipotecario. Tenga en cuenta que en el ejemplo se usa el método InputBox en lugar de la función InputBox, para que el método pueda verificar el tipo. El enunciado Static hace que Visual Basic conserve los valores de las tres variables, que se mostrarán como valores predeterminados la próxima vez que se ejecute el programa.

NOTAS:

```
Static impPrést

Static intPrést

Static plazPrést

impPrést = Application.InputBox _

        (Prompt:="Importe del préstamo (por ejemplo, 100.000)", _

                Default:=impPrést, Type:=1)

intPrést = Application.InputBox _

        (Prompt:="Tipo de interés anual (por ejemplo, 8,75)", _

                Default:=impPrést, Type:=1)

plázPrést = Application.InputBox _

        (Prompt:="Plazos en años (por ejemplo 30)", _

                Default:=impPrést, Type:=1)

pago = Application.WorksheetFunction _

        .Pmt(intPrést / 1200, plazPrést * 12, canPrést)


MsgBox "El pago mensual es " & Format(pago, "Currency")
```

NOTAS:

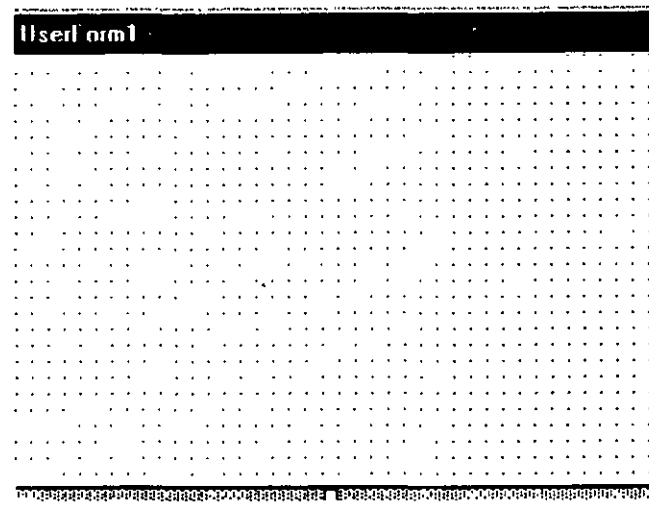
CAPÍTULO 8. FORMULARIOS

USERFORM

Un objeto UserForm es una ventana o cuadro de diálogo que conforma una parte del interfaz de usuario de una aplicación.

UserForm (Ventana)

Permite crear ventanas o cuadros de diálogo en el proyecto. Puede dibujar y ver los controles en un formulario.



Mientras esté diseñando un formulario:

- Cada ventana de formulario tiene un botón Maximizar, Minimizar y Cerrar.
- Puede ver la cuadrícula del formulario y determinar el tamaño de las líneas de cuadrícula en la ficha General del cuadro de diálogo Opciones.
- Puede utilizar los botones del cuadro de herramientas para dibujar controles en el formulario. Puede establecer controles para alinearlos con la cuadrícula del formulario en la ficha General del cuadro de diálogo Opciones.

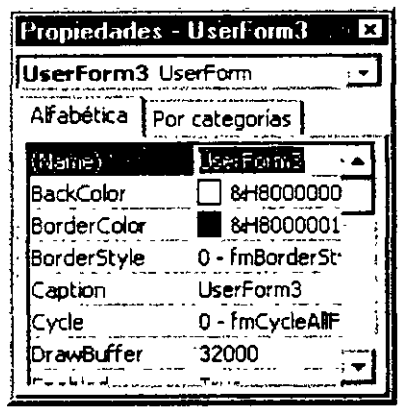
NOTAS:

Crear un UserForm

Para crear un UserForm, haga clic en UserForm en el menú Insertar del Editor de Visual Basic.

O utilize el botón de la barra de herramienta estandar 

Utilice la ventana Propiedades para cambiar el nombre, comportamiento y aspecto del formulario



Propiedades de UserForm

ActiveControl	Identifica y permite la manipulación del control que tiene el enfoque. Sintaxis: objeto.ActiveControl
BackColor	Especifica el color de segundo plano del objeto. color de segundo plano: El color de la región del cliente de una ventana vacía o de una pantalla , en la cual tiene lugar el diseño y muestra de color. Sintaxis: objeto.BackColor [= Long]
BorderColor	Especifica el color del borde de un objeto.

NOTAS:

BorderStyle	Especifica el tipo de borde utilizado por un control o un formulario.
Caption	Texto descriptivo que identifica o describe a un objeto. Sintaxis: objeto.Caption [= String]
Enabled	Especifica si un control puede recibir el enfoque y responder a eventos generados por el usuario.
Font	Define las características del texto utilizado por un control o un formulario. Sus propiedades: Bold, Italic, Size, StrikeThrough, Underline, Weight
ForeColor	Especifica el color de primer plano de un objeto. ***color de primer plano***: El color que está seleccionado actualmente para diseñar o mostrar texto en la pantalla. En las pantallas monocromáticas, el color de primer plano es el de un mapa de bits u otro gráfico.
Height, Width	El alto o el ancho, en puntos de un objeto.
Left, Top	La distancia entre un control y el borde izquierdo o superior del formulario que lo contiene.
MousePointer	Especifica el tipo de puntero del mouse mostrado cuando el usuario sitúa el mouse sobre un objeto en particular.
Name	Especifica el nombre de un control u objeto, o el nombre de una fuente asociada al objeto Font.
ScrollBars	Especifica si un control, formulario o página tiene barras de desplazamiento verticales, horizontales o ambas.
Visible	Especifica si un objeto es visible o está oculto.
Zoom	Especifica cuánto cambia el tamaño del objeto mostrado.

NOTAS:

Métodos de UserForm

Hide	Oculta un objeto, pero no lo descarga.
Load (instrucción)	Carga un objeto, pero no lo muestra.
PrintForm	Envía una imagen bit a bit de un objeto UserForm a la impresora.
Show	Muestra un objeto UserForm.
Unload	Quita un objeto de memoria.

Eventos de UserForm

Activate, Deactivate	El evento Activate se produce cuando un objeto pasa a ser la ventana activa. El evento Deactivate se produce cuando un objeto ya no es la ventana activa. Sintaxis : Private Sub objeto_Activate() Private Sub objeto_Deactivate()
AddControl	Se produce cuando se inserta un control en un formulario, un control Frame, o un objeto Page de un control MultiPage.
Click	Se produce en uno de estos dos casos: El usuario hace clic en un control con el mouse. El usuario selecciona definitivamente un valor para un control con más de un valor posible.
DblClick	Se produce cuando el usuario señala a un objeto y hace clic dos veces con un botón del mouse.
KeyDown, KeyUp	Se producen en secuencia cuando un usuario presiona y suelta una tecla. El evento KeyDown se produce cuando el usuario presiona una tecla. El evento KeyUp se produce cuando el usuario suelta una tecla.

NOTAS:

MouseMove	Se produce cuando el usuario mueve el mouse.
Zoom	Se produce cuando cambia el valor de la propiedad Zoom.

La caja de herramientas

Es un conjunto de herramientas que se emplean para embellecer un formulario en blanco con los controles necesarios.

CONTROLES DEL CUADRO DE HERRAMIENTAS ESTÁNDAR

Seleccionar objetos

Seleccionar objetos es el único elemento del cuadro de elementos que no dibuja un control. Cuando se selecciona, sólo puede cambiar el tamaño o mover un control que ya se haya dibujado en un formulario.

Etiqueta (Label)

Permite tener texto que no desee que cambie el usuario, como el título debajo de un gráfico.

Cuadro de texto (TextBox)

Contiene texto que el usuario puede introducir o cambiar.

Cuadro combinado(ComboBox)

Permite dibujar un cuadro de lista combinado y un cuadro de texto. El usuario puede elegir un elemento de la lista o introducir un valor en el cuadro de texto.

Cuadro de lista (ListBox)

Se utiliza para mostrar una lista de elementos entre los que puede elegir el usuario. Puede desplazarse por la lista si ésta contiene más elementos de los que se pueden ver en un determinado momento.

NOTAS:

Casilla de verificación (CheckBox)

Crea una casilla que el usuario puede elegir fácilmente para indicar si algo es verdadero o falso o para mostrar varias elecciones cuando el usuario puede elegir más de una.

Botón de opción (OptionButton)

Permite mostrar varias elecciones entre las que el usuario sólo puede elegir una.

Botón de alternar(ToggleButton)

Crea un botón que alterna entre activado y desactivado.

Marco (Frame)

Permite crear una agrupación gráfica o funcional de controles. Para agrupar los controles, dibuje primero el marco y después los controles dentro del marco.

Botón de comando(CommandButton)

Crea un botón que el usuario puede elegir para realizar la acción de un comando.

Barra de tabulaciones

Permite definir múltiples páginas para la misma área de una ventana o cuadro de diálogo de la aplicación.

Página múltiple

Presenta múltiples pantallas de información como un solo conjunto.

Barra de desplazamieno

Proporciona una herramienta gráfica para desplazarse rápidamente por una larga lista de elementos o una gran cantidad de información, para indicar la posición actual en una

NOTAS:

escala o como un dispositivo de entrada o indicador de velocidad o cantidad.

Botón de número

Un control de giro que se puede utilizar con otro control para aumentar o reducir los números. También lo puede utilizar para desplazarse hacia delante o detrás de un intervalo de valores o una lista de elementos.

Imagen

Muestra una imagen gráfica de un mapa de bits, icono o metaarchivo en el formulario. Las imágenes mostradas en un control Imagen sólo pueden ser decorativas y utilizan menos recursos que un Cuadro de imagen

Agregar un control a un formulario

Utilice cualquiera de los siguientes métodos para agregar un control del Cuadro de herramientas al formulario. También puede utilizar estos métodos para insertar un control en un control Frame, TabStrip o MultiPage del formulario.

- Haga clic en un control en el Cuadro de herramientas y después haga clic en el formulario. El control aparece en su tamaño predeterminado. Puede arrastrar el control para cambiar su tamaño.
- Arrastre un control del Cuadro de herramientas al formulario. El control aparece en su tamaño predeterminado.
- Haga doble clic en el control del Cuadro de herramientas y después haga clic en el formulario una vez por cada control que desee crear. Por ejemplo, para crear cuatro botones de comando, haga doble clic en el control CommandButton del Cuadro de herramientas y después haga clic cuatro veces en el formulario.

NOTAS:

Eliminar un elemento del Cuadro de herramientas

1 En el cuadro de herramientas, haga clic con el botón secundario del mouse en el icono del elemento que desea quitar.

2 En el menú de acceso directo, seleccione Eliminar. El comando incluirá el nombre del control seleccionado.

CUADROS DE TEXTO (TEXTBOX)

TextBox muestra información de un usuario o de un conjunto de datos organizados.

Un control TextBox es el control utilizado más habitualmente para mostrar información escrita por un usuario. También puede mostrar un conjunto de datos como una tabla, una consulta, una hoja de cálculo o el resultado de un cálculo. Si un control TextBox es dependiente de un origen de datos, al cambiar el contenido del control TextBox también cambia el valor del origen de datos dependiente.

El formato aplicado a cualquier fragmento de texto en un control TextBox afectará a todo el texto del control. Por ejemplo, si cambia la fuente o el tamaño del punto de cualquier carácter del control, el cambio afectará a todos los caracteres del control.

La propiedad predeterminada de un control TextBox es Value.

El evento predeterminado de un control TextBox es Change.

Propiedades

TextBox es un control flexible controlado por las siguientes propiedades: Text, MultiLine, WordWrap y AutoSize.

La propiedad **Text** contiene el texto que se va a mostrar en el cuadro de texto.

La propiedad **MultiLine** controla si el control TextBox puede mostrar texto en una única línea o en múltiples líneas. Los caracteres de nueva línea identifican dónde termina una línea y empieza otra. Si la propiedad MultiLine es False, el texto se trunca en vez de ajustarse.

NOTAS:

La propiedad WordWrap permite que el control TextBox ajuste las líneas de texto más largas que el ancho del mismo para que quepan.

Si no utiliza la propiedad WordWrap, el control TextBox inicia una nueva línea de texto cuando encuentra un carácter de nueva línea en el texto. Si la propiedad WordWrap está desactivada, puede tener líneas de texto que no quepan completamente en el control TextBox. El control TextBox muestra las partes de texto que caben dentro de su ancho y trunca las partes de texto que no caben. La propiedad WordWrap no es aplicable a menos que MultiLine sea True.

La propiedad AutoSize controla si el control TextBox se adapta para mostrar todo el texto. Cuando se utiliza la propiedad AutoSize con un control TextBox, el ancho del control TextBox se comprime o se expande de acuerdo con la cantidad de texto existente en el control TextBox y el tamaño de fuente utilizado para mostrar el texto.

La propiedad AutoSize se comporta bien en las siguientes situaciones:

- Presentación de un titulo de una o más líneas.
- Presentación del contenido de un control TextBox de una única línea.
- Presentación del contenido de un control TextBox de múltiples líneas que es de sólo lectura para el usuario.

La propiedad Value

Especifica el estado o el contenido de un control determinado.

Sintaxis

objeto.Value [= Variant]

La sintaxis de la propiedad Value consta de las siguientes partes:

Parte Descripción

objeto	Requerido. Un objeto válido.
Variant	Opcional. El estado o el contenido del control.

NOTAS:

Valores

Un valor entero que indica si el elemento está seleccionado:

Null Indica que el elemento no está ni seleccionado ni borrado.

–1 Verdadero. Indica que el elemento está seleccionado.

0 Falso. Indica que el elemento está desactivado.

Evite el uso de la propiedad AutoSize con un control TextBox vacío que también puede utilizar las propiedades MultiLine y WordWrap. Cuando el usuario introduce texto en un control TextBox con estas propiedades, el control TextBox cambia de tamaño automáticamente a un cuadro alto y estrecho de un carácter de ancho y del mismo largo que la línea de texto.

Métodos de Cuadros de Texto(TextBox)

SetFocus	Mueve el enfoque a esta instancia de un objeto.
Copy	Copia el contenido de un objeto al Portapapeles.
Cut	
Move	
Paste	
ZOrder	

Eventos de Cuadros de Texto(TextBox)

Change	Se produce cuando cambia el valor de la propiedad Value.

NOTAS:

DblClick	

ETIQUETAS(LABEL)

Un control Label en un formulario muestra un texto descriptivo como títulos, leyendas, imágenes o breves instrucciones. Por ejemplo, las etiquetas para una libreta de direcciones podrían incluir un control Label para el nombre, la calle o la ciudad. Un control Label no muestra valores de orígenes de datos ni expresiones; es siempre independiente y no cambia cuando se mueve de un registro a otro.

La propiedad predeterminada de un control Label es Caption.

El evento predeterminado de un control Label es Click.

Propiedades de Etiquetas(Label)

Caption	Texto descriptivo que identifica o describe a un objeto.
Font	Define las características del texto utilizado por un control o un formulario.
TextAlign	Especifica cómo se alinea el texto en un control.

Métodos de Etiquetas(Label)

Move	Mueve un formulario o un control o mueve todos los controles de la colección Controls.
ZOrder	Coloca el objeto al principio o al final del orden-z.

Eventos de Etiquetas(Label)

Click	Se produce en uno de estos dos casos:

NOTAS:

	El usuario hace clic en un control con el mouse. El usuario selecciona definitivamente un valor para un control con más de un valor posible.
DblClick	Se produce cuando el usuario señala a un objeto y hace clic dos veces con un botón del mouse.

BOTÓN DE COMANDO (COMMANDBOTTON)

Inicia, finaliza o interrumpe una acción o una serie de acciones.

La propiedad predeterminada de un control CommandButton es Value.

El evento predeterminado de un control CommandButton es Click.

Propiedades de Botón de Comando (CommandButton)

Autosize	Especifica si un objeto cambia de tamaño automáticamente para mostrar todo su contenido.Sintaxis objeto.Autosize[= Boolean] True: Cambia automáticamente el tamaño del control para mostrar todo su contenido.
Cancel	Devuelve o establece un valor indicando si un botón de comando es el botón Cancelar en un formulario. Sintaxis: objeto.Cancel [= Boolean] Boolean =True El control CommandButton es el botón Cancelar.
Caption	
Default	Designa el botón de comando predeterminado de un formulario. Sintaxis: bjeto.Default [= Boolean] True El control CommandButton es el botón predeterminado.

NOTAS:

Accelerator	Establece o recupera la tecla de aceleración para un control.
Value	Especifica el estado o el contenido de un control determinado.
TakeFocusOnClick	Especifica si un control recibe el enfoque cuando se hace clic en él.
TabStop	Indica si un objeto puede recibir el enfoque cuando el usuario presiona la tecla TAB para ir al mismo. Sintaxis: objeto.TabStop[= Boolean] True Determina si el objeto puede recibir el foco al presionar TAB

Ejemplo del Uso de Propiedades Acceleraor y Caption

Este ejemplo cambia las propiedades Accelerator y Caption de un control CommandButton cada vez que el usuario hace clic en el botón utilizando el mouse o la tecla de aceleración. El evento Click contiene el código para cambiar las propiedades Accelerator y Caption.

```
Private Sub UserForm_Initialize()
        CommandButton1.Accelerator = "C"
' Establece la tecla de aceleración a ALT + C
End Sub

Private Sub CommandButton1_Click ()
If CommandButton1.Caption = "Aceptar" Then
        ' Comprueba el título y lo cambia.
CommandButton1.Caption = "Se hizo clic"
CommandButton1.Accelerator = "S"        'Establece la tecla de aceleración a ALT + S
```

NOTAS:

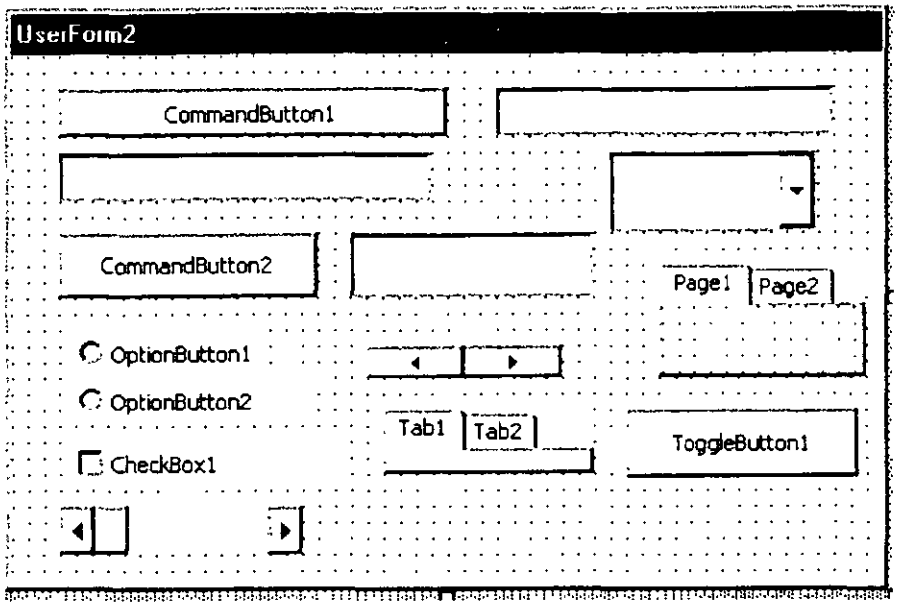
```
Else
        CommandButton1.Caption = "Aceptar"
        CommandButton1.Accelerator = "A"        'Establece la tecla de aceleración a ALT + A
End If
End Sub
```

Ejemplo de la Propiedad Value

El siguiente ejemplo demuestra los valores que pueden tener los diferentes tipos de controles mostrando la propiedad Value de un control seleccionado.

Un control CommandButton llamado CommandButton1. Un control TextBox llamado TextBox1. Un control CheckBox llamado CheckBox1. Un control ComboBox llamado ComboBox1. Un control CommandButton llamado CommandButton2. Un control ListBox llamado ListBox1. Un control MultiPage llamado MultiPage1. Dos controles OptionButton llamados OptionButton1 y OptionButton2. Un control ScrollBar llamado ScrollBar1. Un control SpinButton llamado SpinButton1. Un control TabStrip llamado TabStrip1. Un control TextBox llamado TextBox2. Un control ToggleButton llamado ToggleButton1.

NOTAS:



codigo

```
Dim i As Integer

Private Sub CommandButton1_Click()

   TextBox1.Text = "El valor de " & ActiveControl.Name & " es " & ActiveControl.Value

End Sub


Private Sub UserForm_Initialize()

   CommandButton1.Caption = "Obtener el valor del control actual "

   CommandButton1.AutoSize = True

   CommandButton1.TakeFocusOnClick = False
```

NOTAS:

```
    CommandButton1.TabStop = False


    TextBox1.AutoSize = True
      For i = 0 To 10
      ComboBox1.AddItem "Opción " & (i + 1)


      ListBox1.AddItem "Selección " & (100 - i)
    Next i
      CheckBox1.TripleState = True
    ToggleButton1.TripleState = True
      TextBox2.Text = "Escriba aquí un texto."
End Sub
```

Métodos del Control Botón de Comando (CommandButton)

Move	Mueve un formulario o un control o mueve todos los controles de la colección Controls. Sintaxis: objeto.Move([Izquierda [, Superior[, Ancho[, Altura[, Esquema]]]]])
SetFocus	Mueve el enfoque a esta instancia de un objeto.
ZOrder	

NOTAS:

Eventos del Control Botón de Comando (CommandButton)

Click	El usuario hace clic en un control con el mouse. Sintaxis:Private Sub objeto_Click( )
DblClick	Se produce cuando el usuario señala a un objeto y hace clic dos veces con un botón del mouse.

Ejemplo del Método Move en un CommandButton

El siguiente ejemplo demuestra el movimiento de todos los controles en un formulario utilizando el método Move con la colección Controls. El usuario hace clic en el control CommandButton para mover los controles.

```
Private Sub CommandButton1_Click()
   'Mueve cada control 25 puntos a la derecha y 25 puntos arriba en el formulario.
Controls.Move 25, -25
End Sub
```

CUADRO COMBINADO (COMBOBOX)

Combina las características de un control ListBox y un control TextBox. El usuario puede escribir un valor nuevo, como en un control TextBox o bien puede seleccionar un valor existente como en un control ListBox.

La lista en un control ComboBox está formada por filas de datos. Cada fila puede tener una o más columnas, las cuales pueden mostrarse con o sin títulos. Algunas aplicaciones no son compatibles con títulos de columnas, mientras que otras proporcionan solamente una compatibilidad limitada.

NOTAS:

La propiedad predeterminada de un control ComboBox es Value.

El evento predeterminado de un control ComboBox es Change.

Nota Si desea que se muestre siempre más de una línea de la lista, puede utilizar un control ListBox en vez de un control ComboBox. Si desea utilizar un control ComboBox y limitar los valores a los que están incluidos en la lista, puede establecer la propiedad Style del control ComboBox de manera que el control se asemeje a un cuadro de lista desplegable.

Propiedades de Cuadro Combinado (ComboBox)

Style	Especifica cómo el usuario puede elegir o establecer el valor del control. Sintaxis:objeto.Style [= fmStyle] FmStyle :Especifica cómo un usuario establece el valor de un control ComboBox. Valores: 0 (fmStyleDropDownCombo)El control ComboBox funciona como un cuadro combinado desplegable. El usuario puede escribir un valor en el área de edición o seleccionar un valor de la lista desplegable 2 (FmStyleDropDownList) El control ComboBox se comporta como un cuadro de lista. El usuario debe elegir un valor de la lista
Value	Especifica el estado o el contenido de un control determinado. Sintaxis: objeto.Value [= Variant]
Name	Especifica el nombre de un control u objeto
List	Devuelve o establece las entradas de la lista de un control ComboBox.
ListIndex	Identifica el elemento seleccionado actualmente en un control ListBox o ComboBox.

NOTAS:

MaxLength	Especifica el número máximo de caracteres que un usuario puede introducir en un control TextBox o ComboBox.
ListRows	Especifica el número máximo de filas que se muestran en la lista. Sintaxis: objeto.ListRows [= Long]
MatchRequired	Especifica si un valor introducido en la parte de texto de un control ComboBox debe coincidir con una entrada de la parte de lista existente del control. El usuario puede introducir valores no coincidentes, pero no puede abandonar el control hasta que se introduzca un valor coincidente
MatchFound	Indica si el texto que el usuario ha escrito en un cuadro combinado coincide con cualquier entrada de la lista.

Ejemplo de la Propiedad Style en un ComboBox

El siguiente ejemplo utiliza la propiedad Style para cambiar el efecto de escribir en el área de texto de un control ComboBox.

El formulario contiene:

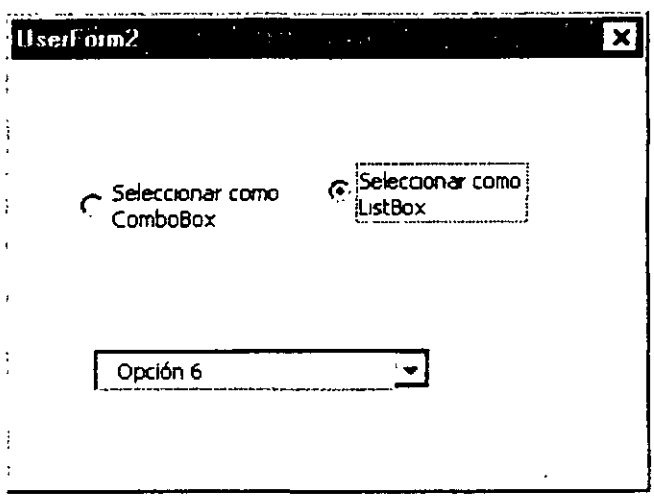
Dos controles OptionButton llamados OptionButton1 y OptionButton2.

Un control ComboBox llamado ComboBox1.

```
Private Sub OptionButton1_Click()
   ComboBox1.Style = fmStyleDropDownCombo
End Sub
Private Sub OptionButton2_Click()
   ComboBox1.Style = fmStyleDropDownList
End Sub
```

NOTAS:

```
Private Sub UserForm_Initialize()
  Dim i As Integer
For i = 1 To 10
    ComboBox1.AddItem "Opción " & i
  Next i
    OptionButton1.Caption = "Seleccionar como ComboBox"
  OptionButton1.Value = True
  ComboBox1.Style = fmStyleDropDownCombo
    OptionButton2.Caption = "Seleccionar como ListBox"
End Sub
```



ListRows (Ejemplo de la propiedad)

El siguiente ejemplo utiliza un control SpinButton para controlar el número de filas de la lista desplegable de un control ComboBox.

NOTAS:

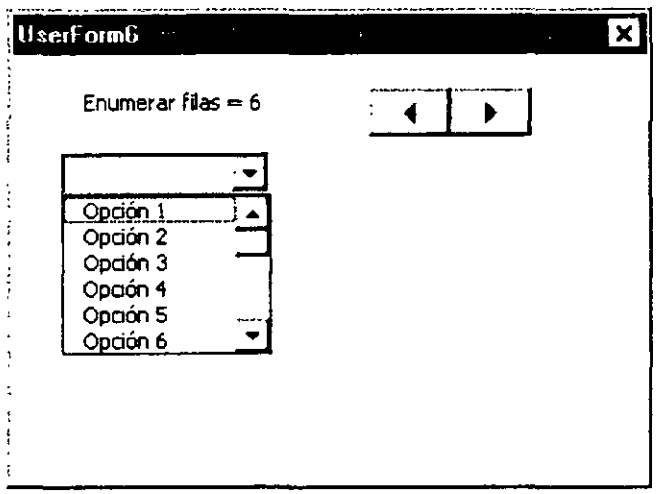
El formulario contiene:

Un control ComboBox llamado ComboBox1.

Un control SpinButton llamado SpinButton1.

Un control Label llamado Label1.

```
Private Sub UserForm_Initialize()
   Dim i As Integer
   For i = 1 To 20
      ComboBox1.AddItem "Opción " & (ComboBox1.ListCount + 1)
   Next i
      SpinButton1.Min = 0
```




```
   SpinButton1.Max = 12
   SpinButton1.Value = ComboBox1.ListRows
   Label1.Caption = "Enumerar filas = " & SpinButton1.Value
End Sub
```

NOTAS:

```
Private Sub SpinButton1_Change()
    ComboBox1.ListRows = SpinButton1.Value
    Label1.Caption = " Enumerar filas = " & SpinButton1.Value
End Sub
```

Métodos de Cuadro Combinado (ComboBox)

AddItem	Para un cuadro combinado, agregue un elemento a la lista o un agregue una fila a la lista.
Clear	Elimina todas las entradas de la lista.
DropDown	Muestra la parte de la lista de un control ComboBox.
RemoveItem	Quita una fila de la lista en un cuadro de lista o un cuadro combinado.
SetFocus	

DropDown (Ejemplo del método)

El siguiente ejemplo utiliza el método DropDown para controlar la presentación de la lista en un control ComboBox. El usuario puede mostrar y cerrar la lista de un control ComboBox haciendo clic en el control CommandButton.

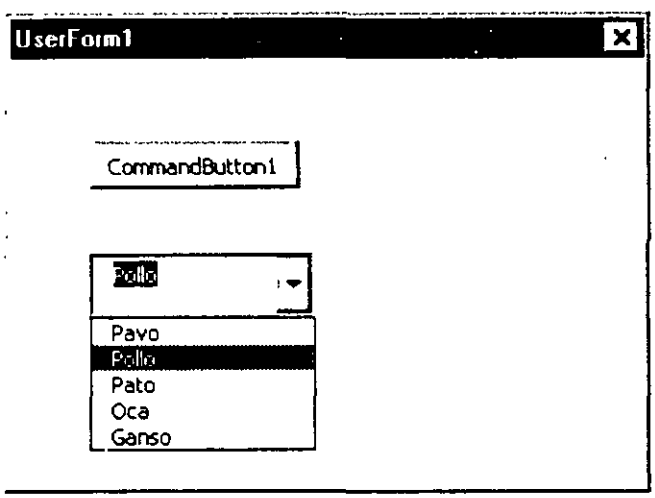
Para utilizar este ejemplo, copie este código de ejemplo en la parte Declaraciones de un formulario. Asegúrese de que el formulario contiene:

- Un control ComboBox llamado ComboBox1.
- Un control CommandButton llamado CommandButton1.

```
Private Sub CommandButton1_Click()
    ComboBox1.DropDown
```

NOTAS:

```
End Sub
Private Sub UserForm_Initialize()
   ComboBox1.AddItem "Pavo"
   ComboBox1.AddItem "Pollo"
   ComboBox1.AddItem "Pato"
   ComboBox1.AddItem "Oca"
   ComboBox1.AddItem "Ganso"
End Sub
```



Eventos de Cuadro Combinado (ComboBox)

Click	
Change	Se produce cuando cambia el valor de la propiedad Value.
DblClick	Sintaxis: Private Sub objeto_DblClick(ByVal Cancelar As MSForms.ReturnBoolean)

NOTAS:

Change	
DropButtonClick	Se produce cada vez que se muestra o se oculta una lista desplegable.

CUADRO DE LISTA(LISTBOX)

Muestra una lista de valores y le permite seleccionar uno o varios.

La propiedad predeterminada de un control ListBox es Value.

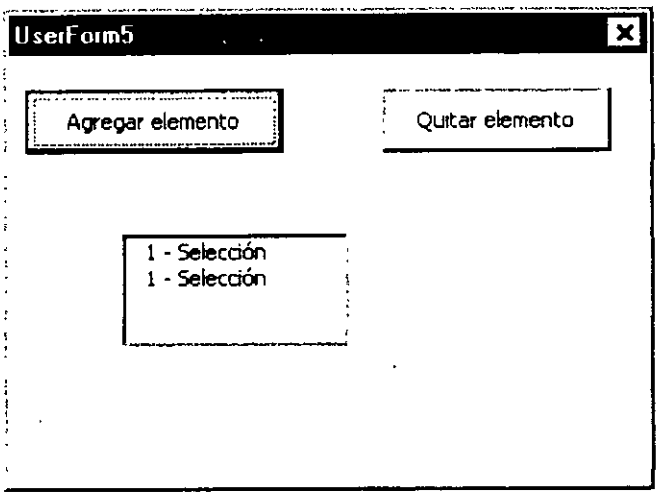
El evento predeterminado de un control ListBox es Click.

Propiedades de Cuadro de Lista (ListBox)

List	Devuelve o establece las entradas de la lista de un control ListBox o ComboBox.
ListCount	Devuelve el número de entradas de lista de un control.
ListIndex	Identifica el elemento seleccionado actualmente en un control ListBox o ComboBox.
ListStyle	Especifica la apariencia visual de la lista en un control ListBox o ComboBox. Sintaxis: objeto.ListStyle [= fmListStyle] fmListStyle] = fmListStylePlain (Cuadro de lista normal, con el fondo de los elementos resaltado.) fmListStyleOption (botones de opción o casillas de verificación para una lista de selección múltiple (predeterminado).
MultiSelect	Indica si el objeto permite selecciones múltiples.

NOTAS:

Selected	Devuelve o establece el estado de selección de los elementos en un control ListBox.



El siguiente ejemplo agrega y elimina el contenido de un control ListBox utilizando los métodos AddItem, RemoveItem y SetFocus y las propiedades ListIndex y ListCount.

El formulario contiene:

- Un control ListBox llamado ListBox1.
- Dos controles CommandButton llamados CommandButton1 y CommandButton2.

```
Dim ContadorEntrada As Single

Private Sub CommandButton1_Click()

  ContadorEntrada = ContadorEntrada + 1

  ListBox1.AddItem (ContadorEntrada & " - Selección")

End Sub


Private Sub CommandButton2_Click()

  ListBox1.SetFocus

  'Asegúrese de que el control ListBox contiene elementos de la lista
```

NOTAS:

```
    If ListBox1.ListCount >= 1 Then
      'Si no hay nada seleccionado, elige el último elemento de la lista.
      If ListBox1.ListIndex = -1 Then
        ListBox1.ListIndex = ListBox1.ListCount - 1
      End If
      ListBox1.RemoveItem (ListBox1.ListIndex   End If
  End Sub
  Private Sub UserForm_Initialize()
    ContadorEntrada = 0
    CommandButton1.Caption = "Agregar elemento"
    CommandButton2.Caption = "Quitar elemento "
  End Sub
```

ListStyle, MultiSelect (Ejemplo de las propiedades)

El formulario contiene:

·Un control ListBox llamado ListBox1.

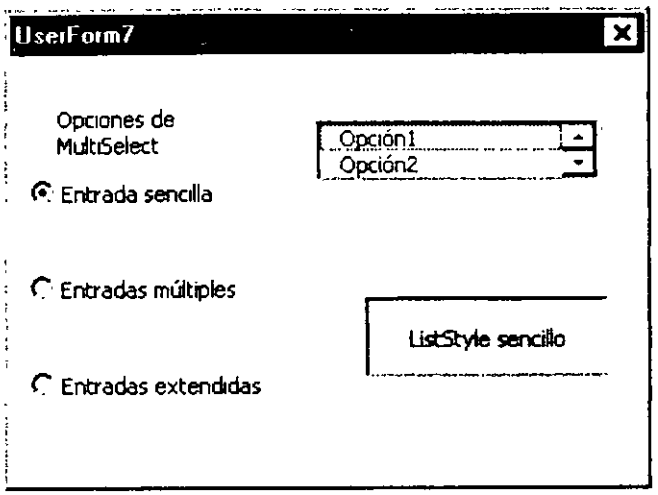
·Un control Label llamado Label1.

·Tres controles OptionButton llamados OptionButton1 hasta OptionButton3.

·Un control ToggleButton llamado ToggleButton1.

NOTAS:

El siguiente ejemplo utiliza las propiedades ListStyle y MultiSelect para controlar la apariencia de un control ListBox. El usuario elige un valor para la propiedad ListStyle utilizando un control ToggleButton y elige un control OptionButton para uno de los valores de la propiedad MultiSelect.




```
Private Sub UserForm_Initialize()

  Dim i As Integer


  For i = 1 To 8

    ListBox1.AddItem "Opción" & (ListBox1.ListCount + 1)

  Next i

    Label1.Caption = "Opciones de MultiSelect"

  Label1.AutoSize = True


  ListBox1.MultiSelect = fmMultiSelectSingle

  OptionButton1.Caption = "Entrada sencilla"

  OptionButton1.Value = True
```

NOTAS:

```
    OptionButton2.Caption = "Entradas múltiples"
    OptionButton3.Caption = "Entradas extendidas"
     ToggleButton1.Caption = "ListStyle - Sencillo"
   ToggleButton1lue = True
    ToggleButton1.Width = 90
    ToggleButton1.Height = 30
End Sub

Private Sub OptionButton1_Click()
   ListBox1.MultiSelect = fmMultiSelectSingle
End Sub

Private Sub OptionButton2_Click()
   ListBox1.MultiSelect = fmMultiSelectMulti
End Sub

Private Sub OptionButton3_Click()
   ListBox1.MultiSelect = fmMultiSelectExtended
End Sub

Private Sub ToggleButton1_Click()
```

NOTAS:

```
    If ToggleButton1.Value = True Then
      ToggleButton1.Caption = "ListStyle sencillo"

      ListBox1.ListStyle = fmListStylePlain
    Else

      ToggleButton1.Caption = "OptionButton o CheckBox"
      ListBox1.ListStyle = fmListStyleOption
    End If
End Sub
```

MultiSelect, Selected (Ejemplo de las propiedades)

El siguiente ejemplo utiliza las propiedades MultiSelect y Selected para demostrar cómo el usuario puede seleccionar uno o más elementos de un control ListBox. El usuario especifica un método de selección eligiendo un botón de opción y después selecciona un elemento o elementos del control ListBox. El usuario puede mostrar los elementos seleccionados en un segundo control ListBox haciendo clic en el control CommandButton.

El formulario contiene:

·Dos controles ListBox llamados ListBox1 y ListBox2.

·Un control CommandButton llamado CommandButton1.

·Tres controles OptionButton llamados OptionButton1 hasta OptionButton3.

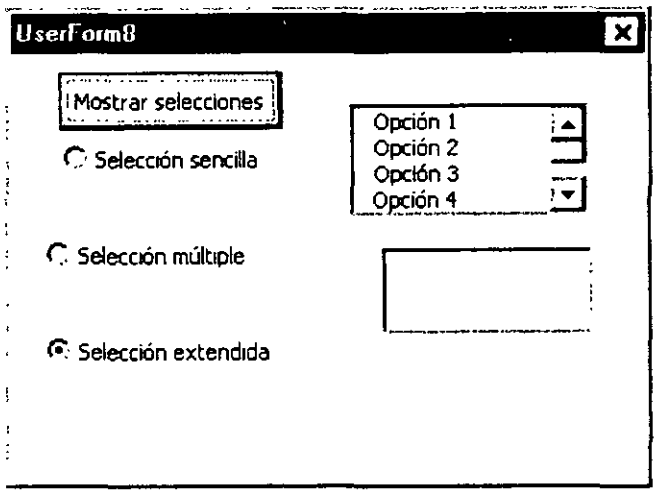
```
Dim i As Integer
```

NOTAS:

```
Private Sub CommandButton1_Click()
   ListBox2.Clear
      For i = 0 To 9
      If ListBox1.Selected(i) = True Then
         ListBox2.AddItem ListBox1.List(i)
      End If
   Next i
End Sub
Private Sub OptionButton1_Click()
   ListBox1.MultiSelect = fmMultiSelectSingle
End Sub
Private Sub OptionButton2_Click()
   ListBox1.MultiSelect = fmMultiSelectMulti
End Sub

Private Sub OptionButton3_Click()
   ListBox1.MultiSelect = fmMultiSelectExtended
End Sub
Private Sub UserForm_Initialize()
   For i = 0 To 9
      ListBox1.AddItem "Opción " & (ListBox1.ListCount + 1)
```

NOTAS:

```
    Next i

       OptionButton1.Caption = "Selección sencilla"

    ListBox1.MultiSelect = fmMultiSelectSingle

    OptionButton1.Value = True

       OptionButton2.Caption = "Selección múltiple"

    OptionButton3.Caption = "Selección extendida"

       CommandButton1.Caption = "Mostrar selecciones"

    CommandButton1.AutoSize = True

End Sub
```



Métodos de Cuadro de Lista (ListBox)

AddItem	Para un cuadro de lista de columna sencilla o un cuadro combinado, agrega un elemento a la lista.
RemoveItem	Quita una fila de la lista en un cuadro de lista o un cuadro combinado.

NOTAS:

SetFocus	Mueve el enfoque a esta instancia de un objeto.
Clear	elimina todas las entradas de la lista.

ListBox (Ejemplo del control)

El siguiente ejemplo agrega y elimina el contenido de un control ListBox utilizando los métodos AddItem, RemoveItem y SetFocus y las propiedades ListIndex y ListCount.

El formulario contiene:

·Un control ListBox llamado ListBox1.

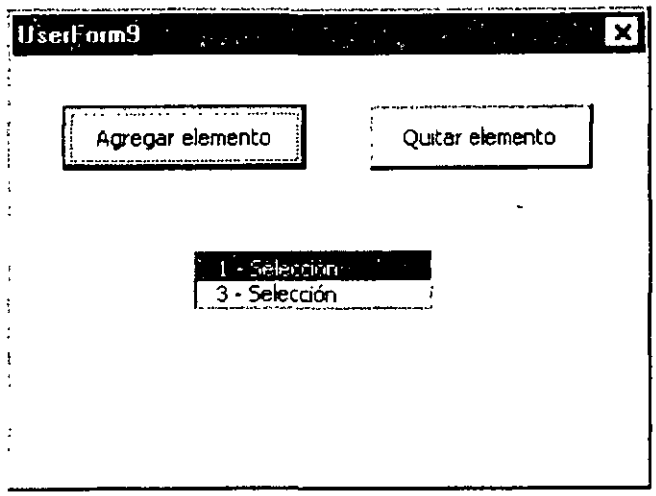
·Dos controles CommandButton llamados CommandButton1 y CommandButton2.

```
Dim ContadorEntrada As Single
Private Sub CommandButton1_Click()
   ContadorEntrada = ContadorEntrada + 1
   ListBox1.AddItem (ContadorEntrada & " - Selección")
End Sub

Private Sub CommandButton2_Click()
   ListBox1.SetFocus
   'Asegúrese de que el control ListBox contiene elementos de la lista
   If ListBox1.ListCount >= 1 Then
```

NOTAS:

```
        'Si no hay nada seleccionado, elige el último elemento de la lista.
        If ListBox1.ListIndex = -1 Then
            ListBox1.ListIndex = ListBox1.ListCount - 1
        End If
        ListBox1.RemoveItem (ListBox1.ListIndex)
    End If
End Sub


Private Sub UserForm_Initialize()
    ContadorEntrada = 0
    CommandButton1.Caption = "Agregar elemento"
    CommandButton2.Caption = "Quitar elemento "
End Sub
```

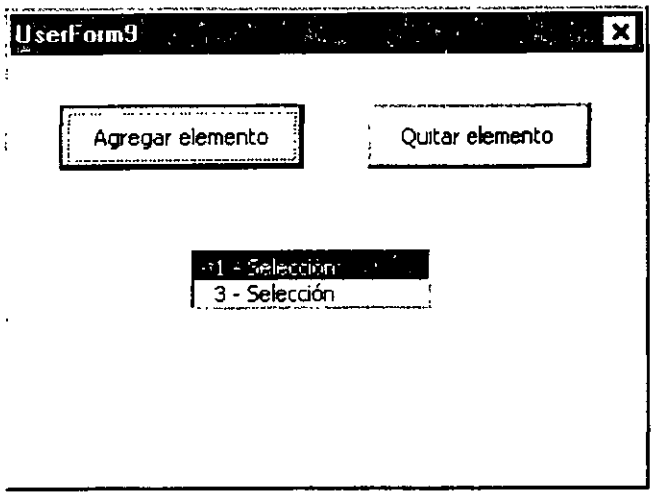
NOTAS:



Eventos de Listbox

Clic
Change
DblClic

NOTAS:



Eventos de Listbox

Clic
Change
DblClic

NOTAS: