FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

CURSOS INSTITUCIONALES

AVANZADO EN COMPUTO II

Del 08 al 12 de Julio de 2002

APUNTES GENERALES

CI-092

Instructor: Ing. Rodolfo González Maldonado

SECRETARÍA DE GOBIERNO

JULIO del 2002

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.

NOTAS:

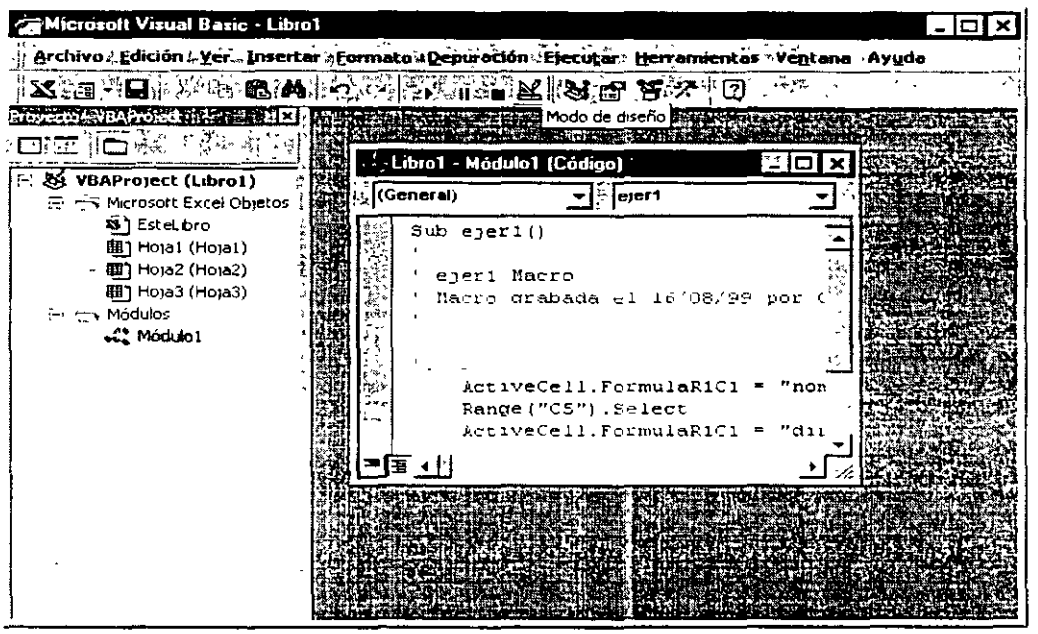
MODIFICAR UNA MACRO

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 Word

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

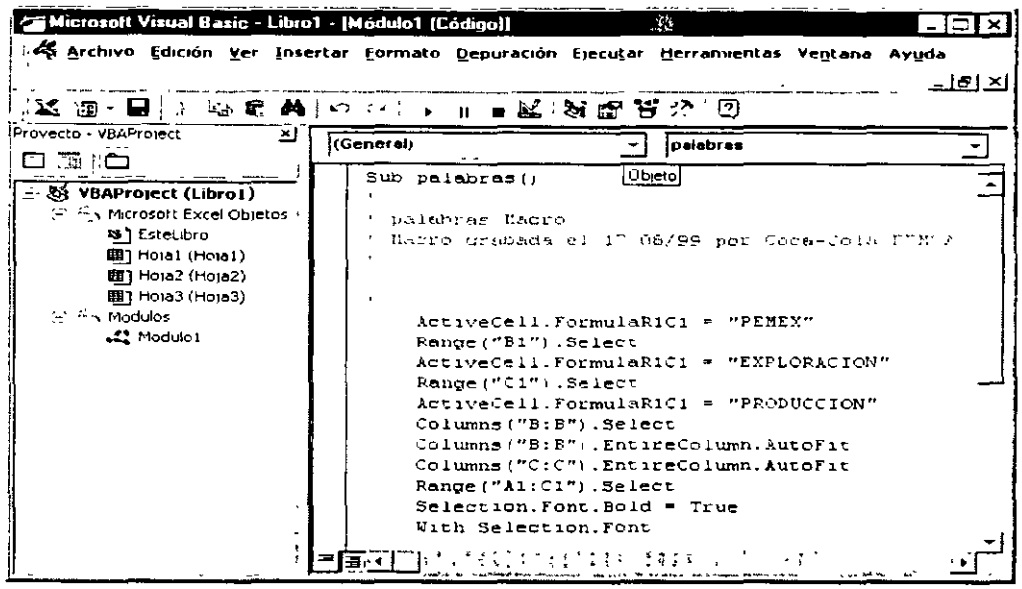
2. En el cuadro Nombre de la macro, seleccione la macro a modificar.

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



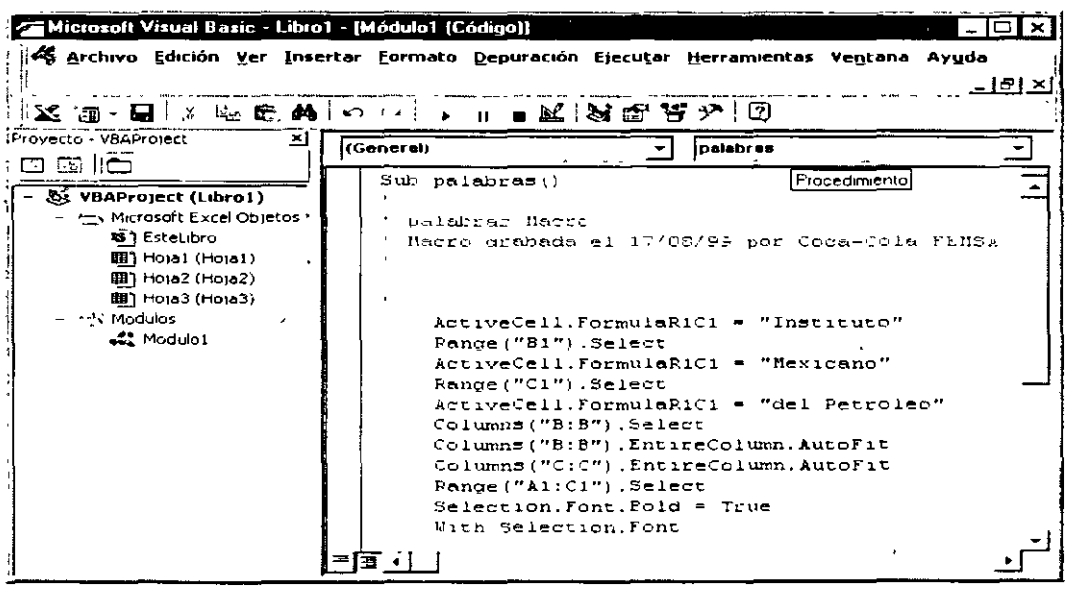
Por ejemplo: Deseamos modificar una macro que escribe las palabras: PEMEX, EXPLORACIÓN, PRODUCCIÓN, cambiándola por Instituto, Mexicano, del Petróleo.

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.

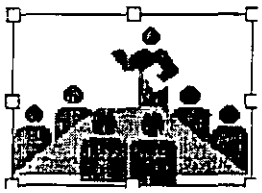
Cerramos el editor y ejecutamos de nuevo la macro, para ver los cambios.



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

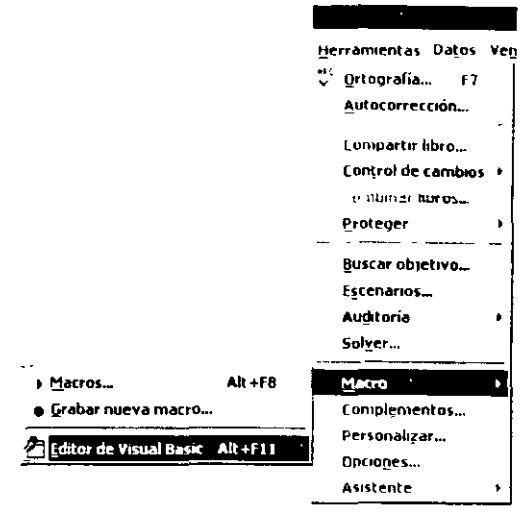
3. Agregar un botón para facilitar la búsqueda, de tal forma que se capture el dato y al dar clic en el botón comience la búsqueda en la base de datos visualizando la información del registro.

NOTAS:

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:

Archivo Edición Ver Insertar Formato Depuración Ejecutar Herramientas Ventana Ayuda

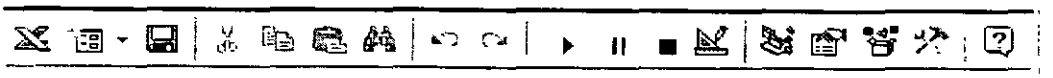
NOTAS:

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

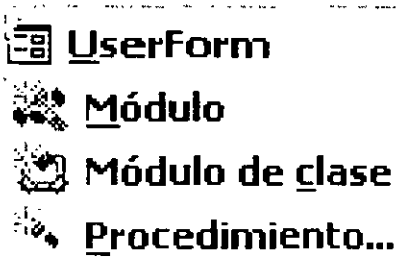


Ver <aplicación principal >

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 Word) disponibles para la aplicación. Sólo está disponible cuando está activo un UserForm.

Asistente de Office

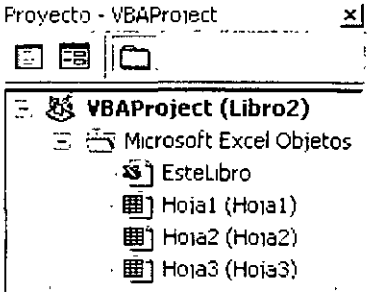
Abre el Asistente de Office donde puede obtener ayuda sobre la ventana, el comando o el cuadro de diálogo que esté activo.

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 Word, 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

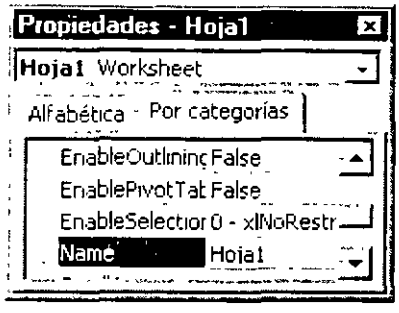
Enumera las propiedades de tiempo de diseño correspondientes a los objetos seleccionados y su configuración actual. Puede cambiar estas propiedades en tiempo de diseño. Cuando seleccione múltiples controles, la ventana de Propiedades contiene una lista de las propiedades comunes a todos los controles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

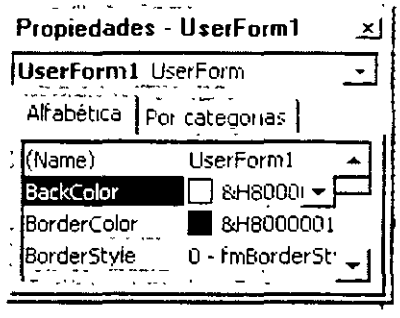
Se puede llamar a la ventana de propiedades usando el icono correspondiente de la barra de herramientas estándar u oprimir la tecla <F4>

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.

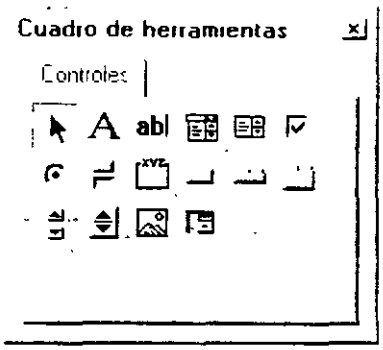
Fichas de Lista de propiedades

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:

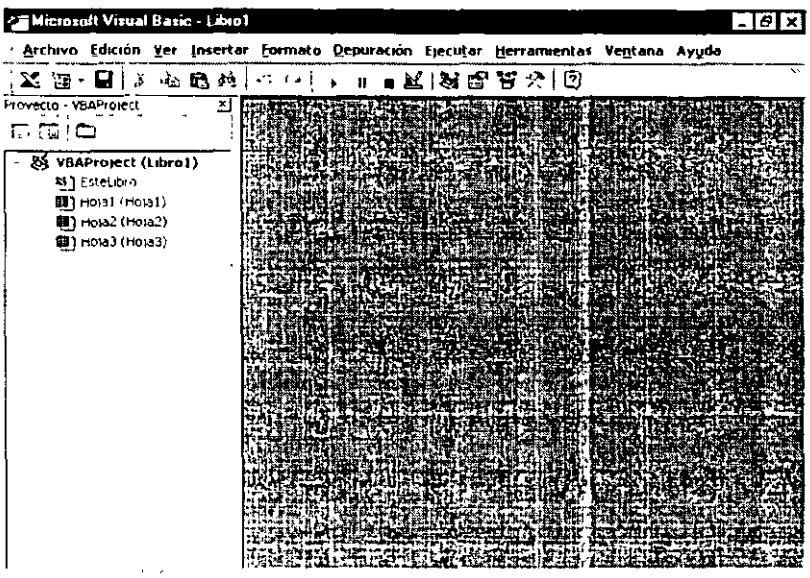
Cuadro de Herramientas



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 especificas 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 vacio.

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

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

NOTAS:

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 Word, 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.

```
Sub CierraPrimero()
```

NOTAS:

```
        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

    End Sub
```

NOTAS:

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:

CAPITULO 4. 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 Word, se puede declarar una variable del tipo Worksheet desde Microsoft Access para representar un objeto Worksheet de Microsoft Word.

NOTAS:

CAPÍTULO 5. 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:

```
        ElseIf rendimiento = 2 Then
            Bonificación= salario * 0.09
        ElseIf 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
[instrucciones]
```

NOTAS:

[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

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

NOTAS:

```
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
        miNum = miNum - 1
```

NOTAS:

```
            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]
[Exit For]
```

NOTAS:

[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 especifica. 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.

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

For Each I In PruebaMatriz
```

NOTAS:

```
        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.

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.

NOTAS:

```
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]
Next [contador]
```

NOTAS:

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 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
```

NOTAS:

```
            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

While condición

[intrucciones]

Wend

NOTAS:

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 6. 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 linea (Chr(13) y Chr(10)) entre cada linea 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

Título = "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 se omite este argumento, el valor predeterminado para buttons es 0..

NOTAS:

title	Opcional. Expresión de cadena que se muestra en la barra de titulo 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.
VbInformation	64	Muestra el icono de mensaje de información.

NOTAS:

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
vbNo	7	No

NOTAS:

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:

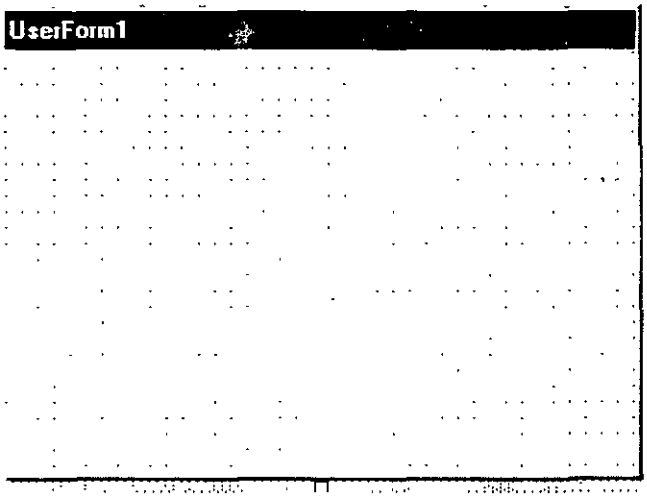
CAPÍTULO 7. FORMULARIOS

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.

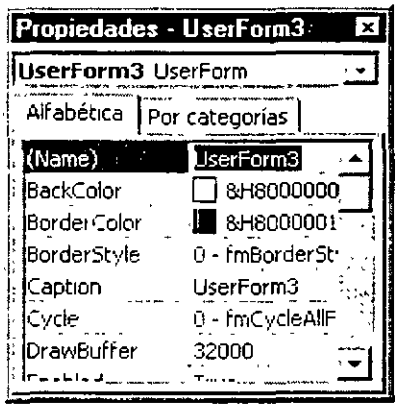
NOTAS:

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.
BorderStyle	Especifica el tipo de borde utilizado por un control o un formulario.

NOTAS:

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

NOTAS:

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.

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.

NOTAS:

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.

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

NOTAS:

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 título 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.

Valores

Un valor entero que indica si el elemento está seleccionado:

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

NOTAS:

–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 vacio 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.
DblClick	

NOTAS:

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: 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.

NOTAS:

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.
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.

NOTAS:

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

NOTAS:

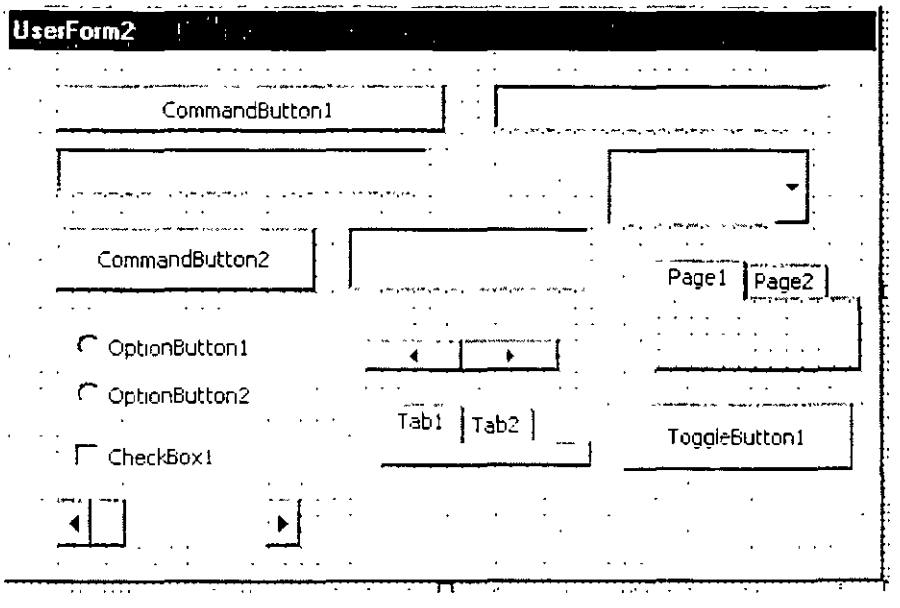
End If

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

   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
```

NOTAS:

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	

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
```

NOTAS:

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.

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

NOTAS:

	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.
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.

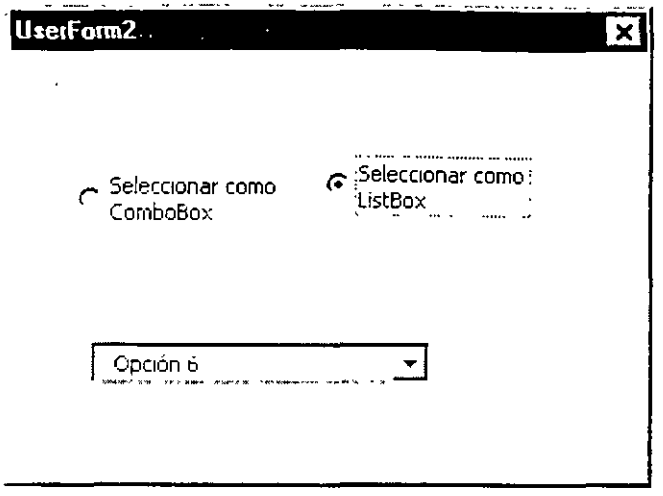
Un control ComboBox llamado ComboBox1.

```
Private Sub OptionButton1_Click()
```

NOTAS:

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

NOTAS:



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.

El formulario contiene:

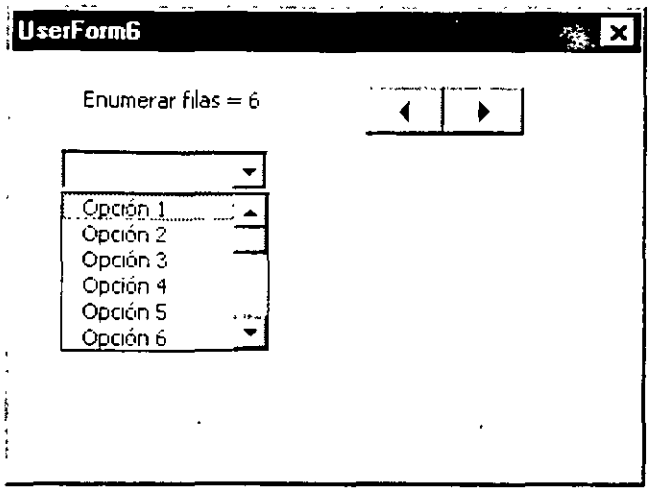
Un control ComboBox llamado ComboBox1.

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
```

NOTAS:




```
    SpinButton1.Max = 12

    SpinButton1.Value = ComboBox1.ListRows

    Label1.Caption = "Enumerar filas = " & SpinButton1.Value

End Sub

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.

NOTAS:

RemoveItem	Quita una fila de la lista en un cuadro de lista o un cuadro combinado.
SetFocus	

DropDown (Ejemplo del método)

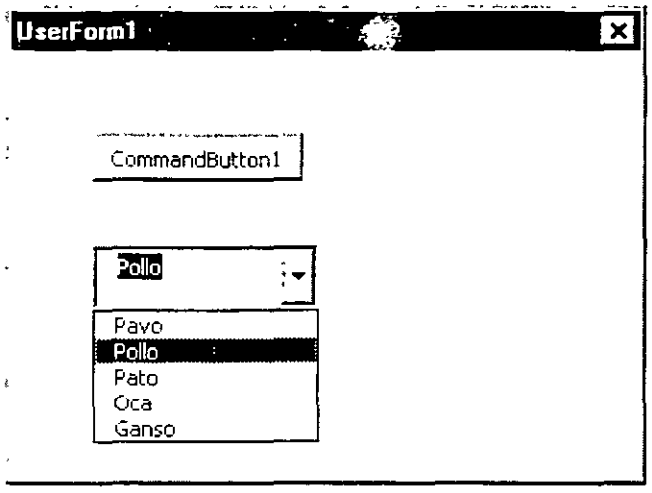
El siguiente ejemplo utiliza el método DropDown para controlar la presentación de la lista en un control ComboBox. El usuario puede mostrar y cerrar la lista de un control ComboBox haciendo clic en el control CommandButton.

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

NOTAS:



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)
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.

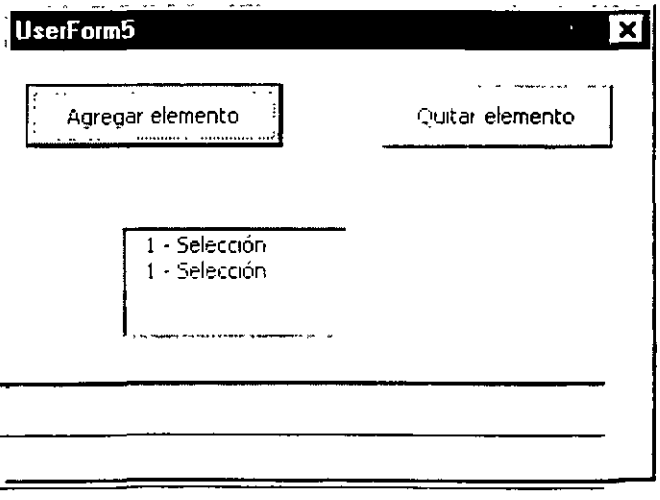
NOTAS:

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.
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:



NOTAS:

- 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
      '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"
```

NOTAS:

```
    CommandButton2.Caption = "Quitar elemento "
End Sub
```

ListStyle, MultiSelect (Ejemplo de las propiedades)

El formulario contiene:

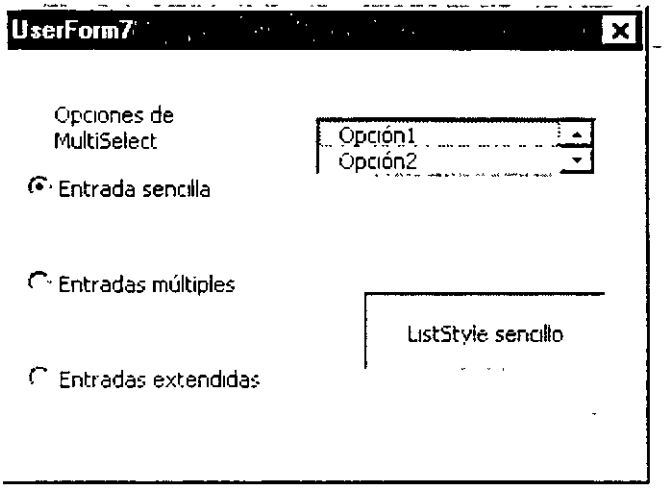
·Un control ListBox llamado ListBox1.

Un control Label llamado Label1.

·Tres controles OptionButton llamados OptionButton1 hasta OptionButton3.

·Un control ToggleButton llamado ToggleButton1.

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()
```

NOTAS:

```
    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
    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
```

NOTAS:

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

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

Private Sub ToggleButton1_Click()
    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
```

NOTAS:

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.

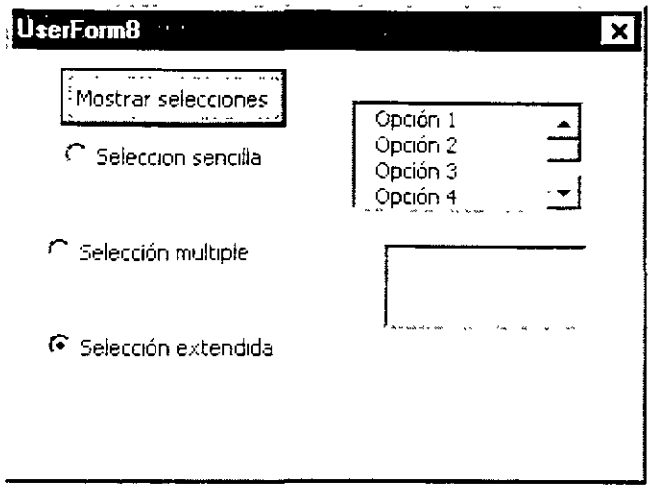
·Tres controles OptionButton llamados OptionButton1 hasta OptionButton3.

```
Dim i As Integer
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()
```

NOTAS:

```
    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)
    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
```

NOTAS:



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.
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:

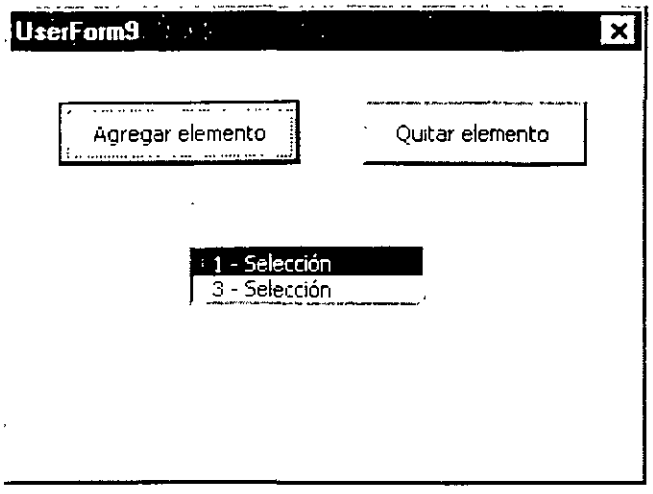
Un control ListBox llamado ListBox1.

NOTAS:

```
'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
        '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
```

NOTAS:

```
    CommandButton1.Caption = "Agregar elemento"
    CommandButton2.Caption = "Quitar elemento "
End Sub
```



Eventos de Listbox

Clic
Change
DblClic

NOTAS: