UNIDAD DIDÁCTICA 8
CUADROS DE DIÁLOGO

1. CUADROS DE DIÁLOGO PREDISEÑADOS

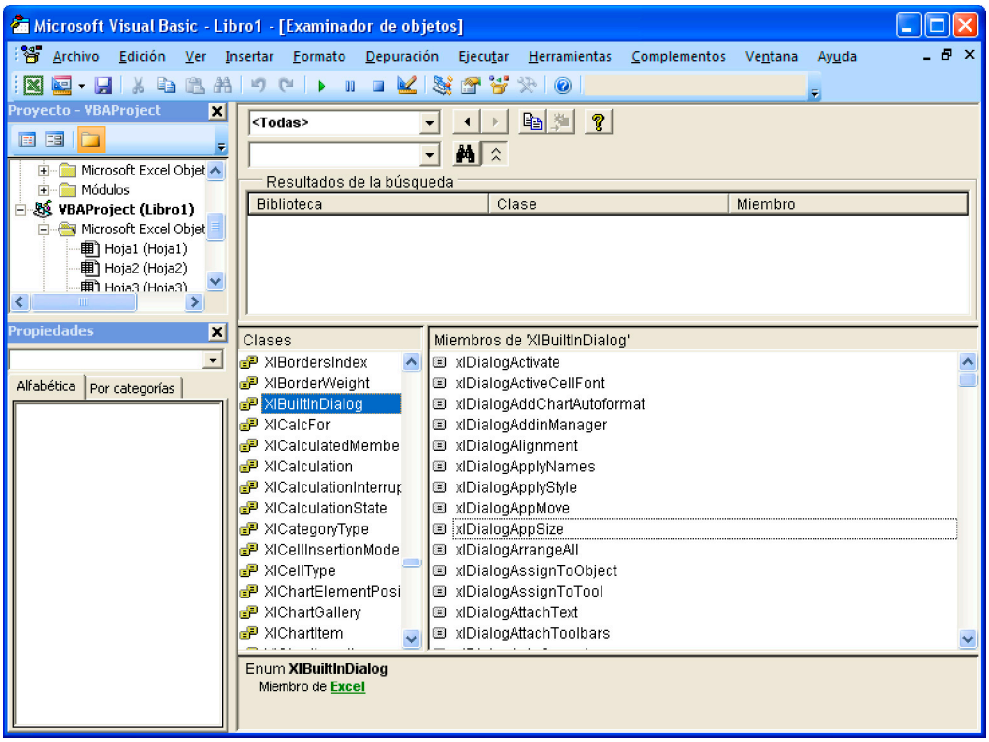
Durante su trabajo con Excel, habrá podido comprobar que accediendo a las distintas opciones del programa, la aplicación nos muestra una infinidad de ventanas y cuadros de diálogo para realizar multitud de tareas. Por ejemplo: abrir un archivo, imprimir una hoja, configurar el formato de las celdas, etc.

Todos esos cuadros de diálogo los podemos utilizar con macros o con parámetros de Visual Basic, facilitándonos las tareas.

Aparte de estos cuadros de diálogo, Excel nos facilita otras ventanas más simples para solicitar una respuesta por parte del usuario, ya sea pulsando un botón o introduciendo datos, etc.

El objeto **Application** que ya conoce, dispone de una propiedad llamada **Dialogs**, que a su vez dispone de una colección de objetos **Dialog**, cada uno de los cuales representa a un cierto cuadro de diálogo.

La clase para utilizar será **XlBuiltinDialog** y desde el **Examinador de objetos** de Visual Basic podremos ver los elementos que forman el grupo.



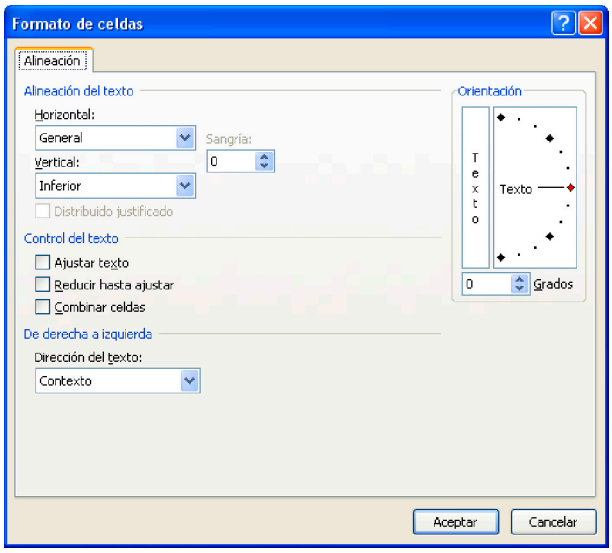
De los miembros con los que cuenta cada objeto, el que nos interesa es el método **Show**. Este método será el que haga aparecer el cuadro de diálogo y, según éste, devolverá un parámetro indicando la acción del usuario sobre la ventana. Lo más normal es que **True** nos indique que se ha hecho clic sobre el botón **Aceptar**, **Abrir** o botones equivalentes, mientras que **False** nos indica la cancelación.

El método **Show** tiene una serie de parámetros para llamar a los cuadros de diálogos o formularios, aunque apenas se utilizan. Es más puede ser que nunca lo utilice.

Un ejemplo de **sintaxis** sería:

Application.Dialogs(xlDialogAlignment).Show

Este código nos mostraría el cuadro de diálogo **Formato de celdas** con la ficha **Alineación**.



De esta forma podrá llamar a cualquier cuadro de diálogo de los que disponga Excel y operar con las distintas opciones de ellos.

A veces nos encontramos con que debemos indicar al usuario que ha cometido un error o simplemente preguntarle si debemos realizar cierta acción o no. Para ello existe una función llamada **MsgBox** que muestra un mensaje en un cuadro de diálogo, espera a que el usuario haga clic en un botón y devuelve un valor correspondiente al botón elegido por el usuario.

Su sintaxis es la siguiente:

MsgBox(Texto[, botones][, cabecera][, ficherodeAyuda, contexto])

- **Texto** es el texto que se va a mostrar en el cuadro de diálogo.
- **Botones** 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 botones es 0.

- **Cabecera** sirve para cambiar el titulo del cuadro de diálogo. Es opcional.
- **FicherodeAyuda** identifica el archivo de ayuda que se utiliza para proporcionar ayuda interactiva en el cuadro de diálogo. Si se especifica fichero de Ayuda, también se debe especificar contexto. Es opcional.
- **Contexto** es igual al número de contexto de Ayuda asignado por el autor al tema de Ayuda correspondiente. Es opcional.

A continuación se muestra una tabla con los valores que puede tener el argumento **Botones**.

Valor	Valor numérico	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
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.

Y los valores que puede devolver son:

Valor	Valor numérico	Descripción
VbOK	1	Se ha pulsado el botón Aceptar
VbCancel	2	Se ha pulsado el botón Cancelar
VbAbort	3	Se ha pulsado el botón Anular
VbRetry	4	Se ha pulsado el botón Reintentar
VbIgnore	5	Se ha pulsado el botón Ignorar
VbYes	6	Se ha pulsado el botón Sí
VbNo	7	Se ha pulsado el botón No

Vamos a ver con ejemplos la utilización de **MsgBox**. Vamos a suponer que escribimos el código dentro del evento de un botón.

```
Sub Botón3_Haga_clic_en()
    MsgBox "¿Desea Salir de Excel?"
End Sub
```

El mensaje que generaría sería el siguiente:

Por defecto aparece el botón **Aceptar** ya que, no hemos especificado nada. Modifiquemos ahora el código para que aparezcan los botones **SI** y **NO**.

```
Sub Botón3_Haga_clic_en()
    MsgBox "¿Desea Salir de Excel?", vbYesNo
End Sub
```

El mensaje que generaría sería el siguiente:

Como podemos apreciar, aparecen los dos botones. Ahora haremos que aparezca también el icono de interrogación, para que a simple vista se vea que el programa nos está preguntado algo.

```
Sub Botón3_Haga_clic_en()
        MsgBox "¿Desea Salir de Excel?", vbYesNo + vbQuestion
End Sub
Sub Botón3_Haga_clic_en()
        MsgBox "¿Desea Salir de Excel?", 36
End Sub
```

El mensaje quedaría así:

Si se fija en el código, las dos funciones generan el mismo mensaje. Si mira la primera tabla, comprobará que el valor de **VbYesNo** es **4**, y el de **VbQuestion** es **32**. La suma de ambos valores es 36, y Visual Basic es capaz de interpretar ese valor con los botones e icono adecuados.

Por último cambiaremos el título de la ventana:

```
Sub Botón3_Haga_clic_en()
MsgBox "¿Desea Salir de Excel?", 36, "Salir del programa"
End Sub
```

El mensaje quedaría:

Ya sabemos construir un mensaje, pero lo interesante es el valor que devuelve al pulsar nosotros algún botón de los que muestra. Por ejemplo y siguiendo con el mismo mensaje, haremos que al pulsar el botón **SI** se cierre el programa de Excel:

```
Sub Botón3_Haga_clic_en()

respuesta = MsgBox(("¿Desea Salir de Excel?"), 36, "Salir del programa")

If respuesta = vbYes Then ActiveWorkbook.Close

End Sub
```

Si no teníamos guardado el libro, se nos preguntará si queremos guardarlo, si no, el programa se cerrará automáticamente.

Existe otra función muy parecida a **MsgBox** llamada **InputBox**. Además de ser parecida nos permite hacerle preguntas al usuario (a través de un cuadro de texto), y que este nos devuelva la respuesta en una variable de tipo **String.**

Su sintaxis es:

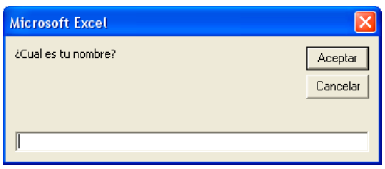
- ***InputBox*** *(mensaje [, titulo][, pordefecto][, xpos][, ypos][, helpfile, contexto])*
- ***mensaje*** es lo que se muestra como mensaje en el cuadro de diálogo. La longitud máxima es de aproximadamente 1024 caracteres, según el ancho de los caracteres utilizados. Si consta de más de una línea, se pueden separar 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.
- ***titulo*** lo que muestra la barra de título del cuadro de diálogo. Si se omite, en la barra de título se coloca el nombre de la aplicación.
- ***pordefecto*** la cadena de texto que devuelve como respuesta predeterminada cuando no se suministra una cadena. Si se omite, se muestra el cuadro de texto vacío.

- ***Xpos,Ypos*** especifica, en twips, la distancia en sentido horizontal o vertical (Xpos o Ypos), entre el borde izquierdo (o superior) del cuadro de diálogo y el borde izquierdo (o superior) de la pantalla. Si se omiten, el cuadro de diálogo se centra en la pantalla.
- ***helpfile*** 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 contexto.
- ***contexto*** Expresión numérica que es el número de contexto de Ayuda asignado por el autor al tema de Ayuda correspondiente.

Veamos un ejemplo, en el que se nos pedirá nuestro nombre, y este quedará escrito en la celda que en ese momento esté seleccionada.

```
Sub Botón1_Haga_clic_en()
        respuesta = InputBox("¿Cuál es tu nombre?")
        If respuesta <> "" Then
                ActiveCell.FormulaR1C1 = respuesta
        End If
End Sub
```

Antes de escribir en la celda seleccionada, comprobamos que el usuario ha escrito algo y que el campo no quedo vacío.



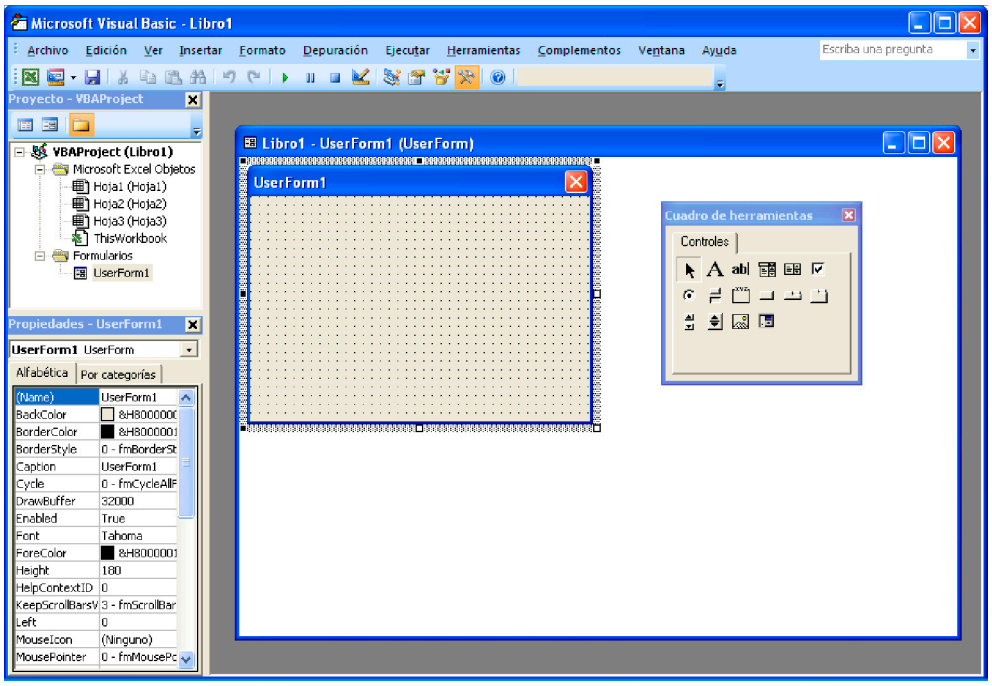
2. CREACIÓN DE NUESTROS PROPIOS CUADROS DE DIÁLOGO

Como ha podido comprobar, Excel cuenta con recursos suficientes como para cubrir nuestras necesidades en la mayoría de las ocasiones. Con los cuadros de diálogo prediseñados podemos abrir y guardar archivos, modificar atributos, acceder a propiedades del entorno, gráficos y tablas, etc. Mediante las funciones **MsgBox** e **InputBox** se facilita la comunicación de mensajes, petición de confirmaciones y solicitud de datos simples.

En ocasiones puede ocurrir que no encontremos el elemento adecuado para la función que deseamos implementar. En estos casos podemos recurrir a la creación de un cuadro de diálogo propio, o lo que es lo mismo, un **Formulario**.

El diseño de un formulario puede ser desde algo muy simple, hasta un formulario bastante complejo, en el que podríamos abarcar otro curso específico.

Para comenzar la creación de un formulario, debemos situarnos en la ventana de Visual Basic y seleccionar la opción **UserForm** del menú **Insertar**. De este modo aparecerá una ventana con un formulario vacío y un cuadro con herramientas para la elaboración del formulario.



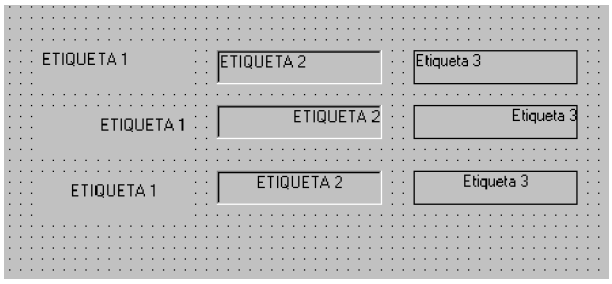
Al insertar un formulario, éste recibe un nombre y título por defecto, para cambiárselo debemos editar la propiedad **Name** y el título en la propiedad **Caption** y establecer los nombres que queramos.

El Formulario vacío nos sirve de poco, para que nos resulte útil debemos incluir en él componentes de los que aparecen en el cuadro de herramientas. Estos elementos son los siguientes:

ETIQUETAS

Un **label** es un control que nos permite presentar un texto. La etiqueta debe usarse en aquellos casos en los que exista una información estática o dinámica que no deba ser cambiada por el operador.

Puede adoptar estas formas: con borde tridimensional, borde plano o sin borde, y el texto justificado a la izquierda, a la derecha o centrado.



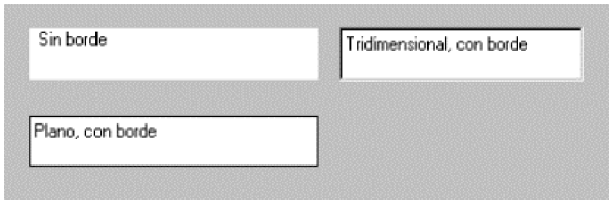
CUADROS DE TEXTO

Las cajas de texto son los controles en los que Visual Basic presenta o introduce textos. Es por tanto un control bidireccional. Normalmente se usan para introdución de textos, o para la presentación de aquellos que el operador pueda cambiar. Para cambiar o escribir un texto en una caja de texto, basta con conseguir que esa caja de texto tenga el foco y teclear el texto en el teclado. Esto se puede lograr, bien haciendo click con el ratón en esa caja de texto, bien con la tecla TAB, bien por programa.

La caja de texto no se debe usar nunca para presentar textos que el operador de la aplicación no deba cambiar. Úsese para ello la etiqueta, control no bidireccional, que además tiene la ventaja de ocupar menos memoria de programa.

Las cajas de texto pueden tener una o varias líneas, según esté la propiedad Multiline. La capacidad máxima de una caja de textos es de 64 Kbytes.

La forma de una caja de texto es la siguiente, dependiendo de las propiedades BorderStyle y Appearance



CUADRO DE LISTA Y CUADROS COMBINADOS

Un **cuadro de lista** muestra una lista de elementos en la que el usuario puede seleccionar uno o más. Si el número de elementos supera el número que puede mostrarse, se agregará automáticamente una barra de desplazamiento al control.

Un **cuadro combinado** combina las características de un **campo de texto** y un **cuadro de lista**. Los usuarios pueden introducir información en la parte del cuadro de texto y seleccionar un elemento en la parte de cuadro de lista del control. En resumen, un **cuadro combinado** es la combinación de un **cuadro de lista**, que se comporta como si de un **cuadro de lista** se tratase, y de un **campo de texto**, con comportamiento análogo a un **campo de texto** sencillo, con la particularidad aquí de que el texto se le puede introducir por teclado, o elegir uno de los que figuran en la parte **cuadro de lista** del Combo.

Estos controles toman la siguiente forma:

Puede verse en la figura un ejemplo de presentación de un **cuadro de lista** (izquierda), un **cuadro combinado** con la lista desplegada (Centro) y un **cuadro combinado** con la lista sin desplegar (Combo2 a la derecha).

La lista tiene varios elementos. Cada línea de esta lísta es un elemento de la lista. Como el número de elementos de la lista tiene más elementos de los que le podían caber, generó automáticamente la barra de desplazamiento vertical.

El **cuadro combinado** está normalmente sin desplegar. Se despliega cuando se hace clic con el ratón en la flecha que tiene en su parte derecha. Al desplegarse, muestra la lista con todos sus elementos. Haciendo clic con el ratón en cualquiera de sus elementos, el elemento elegido pasa a la parte cuadro de texto del Combo y la lista vuelve a replegar.

El **cuadro de lista** (y por tanto el **cuadro combinado**) tienen unas propiedades y métodos particulares que solamente se pueden aplicar durante el tiempo de ejecución:

- **ListCount** - Indica el número de elementos que tiene la lista
- **ListIndex** - Indica el número de orden del elemento seleccionado dentro de la lista.
- **AddItem** - Añade un elemento a la lista.
- **RemoveItem** - Elimina un elemento de la lista.
- **Text** - Obtiene el elemento seleccionado.
- **List (n)** - Obtiene el elemento cuyo orden dentro de la lista es n.

– **ListCount** valdrá 0 si la lista no tiene ningún elemento, y n si tiene n elementos.

Para seleccionar un elemento de la lista, basta con hacer clic con el ratón sobre él. Ese elemento se resaltará con fondo en azul. Una vez seleccionado un elemento, la propiedad **ListIndex** tomará el valor del número de orden que ocupa ese elemento en la lista, comenzando por el 0 para el elemento que ocupa el primer lugar. Si no se selecciona ningún elemento, el valor de la propiedad **ListIndex** será -1. El primer elemento de la lista es **ListIndex** 0, y el valor de la propiedad **ListCount** siempre es uno más que el valor mayor de **ListIndex**.

En el **cuadro combinado** la propiedad **Text** contiene el texto que contenga la parte **TextBox** del Combo, bien haya sido introducida desde teclado o mediante la recuperación de un elemento la parte **ListBox** del mismo.

CASILLA DE VERIFICACIÓN Y BOTÓN DE OPCIÓN

El control **CheckBox**, o casilla de verificación, permite elegir una opción (activada/desactivada, True/False) que el usuario puede establecer o anular haciendo clic. Una X en una casilla de verificación indica que está seleccionada, activada, o con valor **True**. Cada casilla de verificación es independiente de las demás que puedan existir en el formulario, pudiendo tomar cada una de ellas el valor True o False, a voluntad del operador.

Un control **OptionButton**, o botón de opción, muestra una opción que se puede activar o desactivar, pero con dependencia del estado de otros controles **OptionButton** que existan en el formulario.

Generalmente, los controles **OptionButton** se utilizan en un grupo de opciones para mostrar opciones de las cuales el usuario sólo puede seleccionar una. Los controles **OptionButton** se agrupan dibujándolos dentro de un contenedor como un control **Frame**, un control **PictureBox** o un formulario. Todos los controles **OptionButton** que están dentro del mismo contenedor actúan como un solo grupo, e independientes de los controles **OptionButton** de otros grupos distintos.

Aunque puede parecer que los controles **OptionButton** y **CheckBox** funcionan de forma similar, hay una diferencia importante. Cuando un usuario selecciona un **OptionButton**, los otros controles del mismo grupo **OptionButton** de-

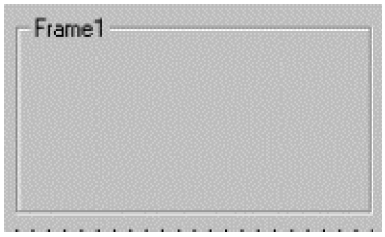
jan de estas disponibles automáticamente. Por contraste, se puede seleccionar cualquier número de controles **CheckBox**.

MARCO

Un control **Frame** o **Marco** proporciona un agrupamiento identificable para controles. También puede utilizar un **Frame** para subdividir un formulario funcionalmente por ejemplo, para separar grupos de controles OptionButton.

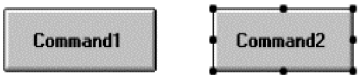
Para agrupar controles, en primer lugar trace el control Frame y, a continuación, meta los controles dentro de Frame. De este modo podrá mover al mismo tiempo el Frame y los controles que contiene. Si traza un control fuera del Frame y, a continuación, intenta moverlo dentro de éste, el control se colocará sobre el Frame, pero no pertenecerá a el. Es decir, si es un OptionButton este se comportará como si estuviese fuera del Frame, aunque físicamente esté dentro de el.

Cuando un control Frame tiene dentro otros controles, y hacemos invisible al Frame, mediante su propiedad **Visible = False**, los controles interiores al Frame quedan también invisibles.



BOTÓN DE COMANDO

El botón de comando puede usarse para la entrada de datos con el ratón, o para validar cualquier operación. El tamaño puede cambiarse a voluntad, pero la forma siempre es rectangular. En la figura vemos dos botones de comando, uno de ellos (el Command2) marcado con unos puntos en su contorno. Estos puntos nos permiten variar su tamaño en tiempo de diseño. También puede cambiarse su tamaño y posición en tiempo de ejecución.



IMAGEN

Este control permite presentar todo tipo de ficheros gráficos (.BMP, WMF, .ICO, .CUR)

Use la propiedad **Stretch** para determinar si el gráfico se escala para que se ajuste al control o viceversa. Poniendo esta propiedad a True el gráfico se adaptará a las dimensiones del control **Image.** Si se pone a False, el control **Image** tomará las medidas del gráfico que contenga. Si el gráfico es un bitmap (Fichero .BMP), con la propiedad **Stretch** a True podemos variar el tamaño del bitmap, variando las propiedades **Width** y **Height** del control **Image**, propiedades que se pueden cambiar en tiempo de ejecución. Si esta propiedad está a False, el tamaño del bitmap no cambiará, presentándose solamente una porción del bitmap, caso que el control **Image** sea menor que el tamaño del bitmap, o sobrará espacio en el control, en caso contrario. Aunque puede colocar un control **Image** en un con-

tenedor, un control **Image** no puede actuar como contenedor. Esto se entiende mejor con un ejemplo.

Un **Image** es transparente, es decir, deja ver el fondo del formulario en las partes no ocupadas por su gráfico. Por lo tanto, no tendrían sentido en este control propiedades como **Backcolor**, **FillColor**, o **FillStyle**.

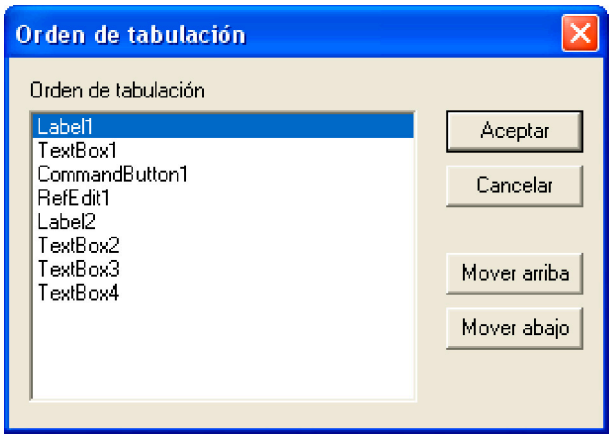
REFEDIT

Con este control podremos solicitar una referencia a una celdilla o rango. Al pulsar sobre él, se habilitará la hoja de cálculo para que seleccionemos las celdillas.

Un cuadro de diálogo personalizado se usa como cualquier otro. Con el método **Show** lo haríamos aparecer, mientras que usando el método **Hide** del propio cuadro de diálogo lo ocultaríamos. Al mostrar el cuadro de diálogo, el acceso a los componentes se realizará como en cualquier otra ventana. En principio habrá un control activo y utilizando la tecla **Tab** iríamos accediendo a los demás.

Todos los controles cuentan con las propiedades **TabStop** y **TabIndex**. Con la primera indicaremos si debe accederse o no al control usando la tecla **Tab**. El valor por defecto, **True** indica que así debe ser, pero podemos modificarlo. Mediante **TabIndex** se establece el orden de acceso a los componentes. Su contenido será un número que indicará la posición que ocupa el control en ese orden de acceso, independientemente de su posición física en el formulario.

En lugar de establecer individualmente la propiedad **TabIndex** de cada componente, lo cual es un proceso tedioso, puede usar la opción **Orden de tabulación** del menú emergente asociado al formulario. Al elegirla aparecerá una ventana, como la de la figura de la derecha.



En la lista aparecen todos los controles del formulario. Puede seleccionar cualquiera de ellos y, a continuación, utilizar los botones **Mover arriba** y **Mover abajo** para cambiar el orden de acceso.