



Interacción Persona Ordenador

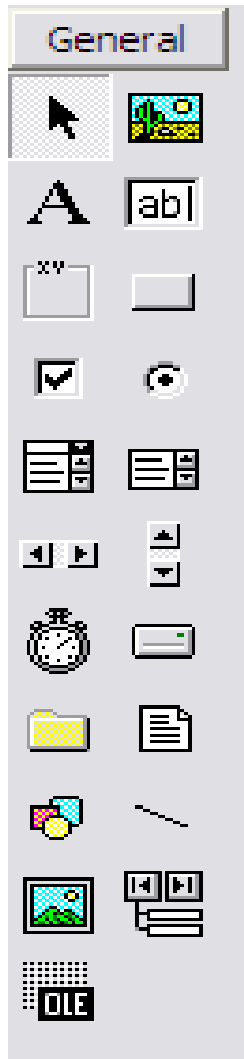
Visual Basic

Tema 4

Controles, Matrices y Procedimientos

Visual Basic

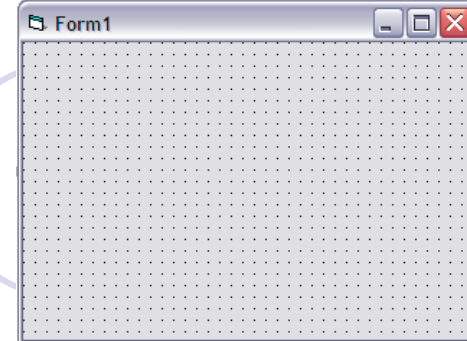
Controles.



- La finalidad principal de un formulario es servir de contenedor para los controles que nuestro programa necesite. Estos controles permitirán mostrar información al usuario, solicitar datos de distintas formas, ofrecer listas de elementos, etc.
- Cada uno de los controles tiene una serie de propiedades, implementa unos métodos y puede responder a un conjunto de eventos. Muchos de estos elementos son comunes a todos los controles, por ejemplo las propiedades *Name* o *Caption*, o el evento *MouseDown* existen en casi todos ellos.

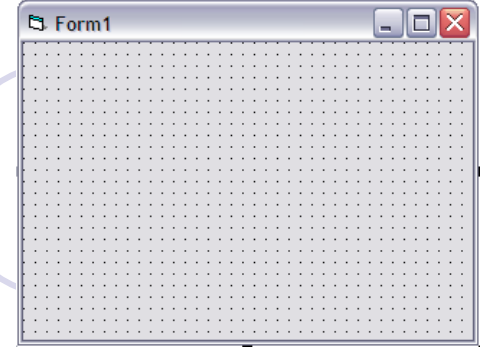
Visual Basic

Controles – Formulario.



- Del formulario ya hemos hablado en temas anteriores. Sólo vamos a hacer algunas indicaciones:

Evento	Ocorre cuando
Load	Se carga un formulario
Unload	El formulario está a punto de cerrarse
KeyDown	El usuario pulsa una tecla
KeyUp	El usuario suelta una tecla
KeyPress	El usuario pulsa una tecla; este evento ocurre después del evento KeyDown y antes de KeyUp
MouseDown	El usuario pulsa un botón del ratón
MouseUp	El usuario suelta el botón pulsado del ratón
Click	El usuario pulsa y suelta el botón de un ratón



- **Procedimientos y Métodos:**

- **Cls:** Borra todos los gráficos del formulario.

[Nombre_Formulario].Cls

- **LoadPicture:** Nos permite cargar un dibujo que puede ser:

- Un Icono (.ICO).
 - Un BitMap (.BMP).
 - Un MetaArchivo (.WMF).

LoadPicture ([Nombre_Archivo])

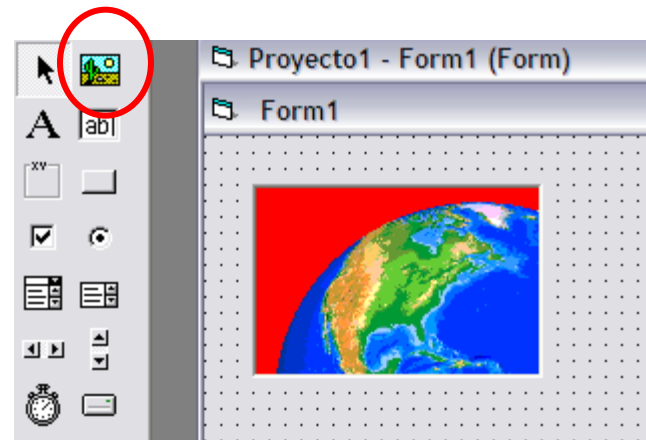
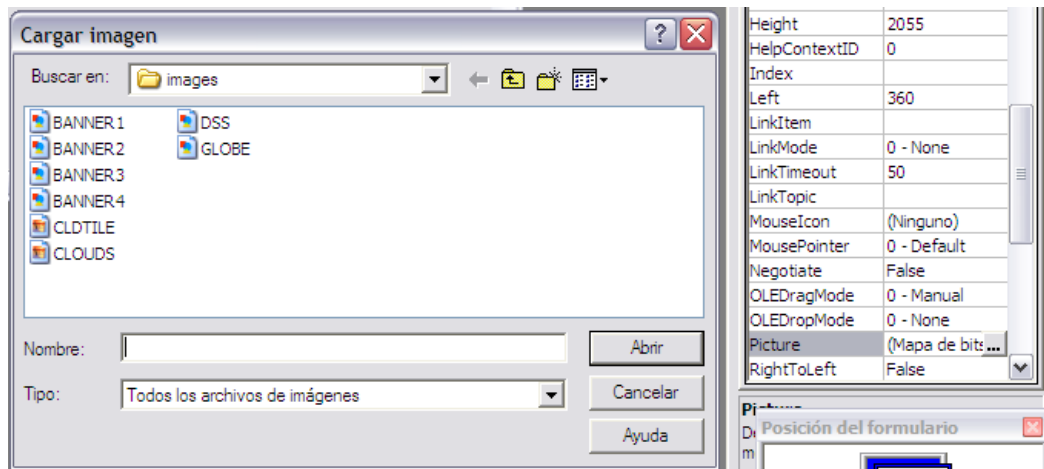
- **Print:** Imprime un texto en el formulario.

[Nombre_Formulario].Print[[Expresión][{;|,}]...]

Visual Basic

Controles – Caja de Imagen (PictureBox).

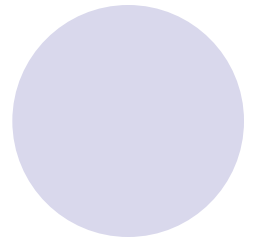
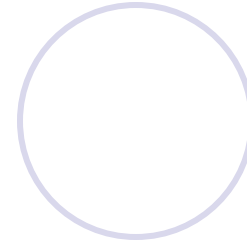
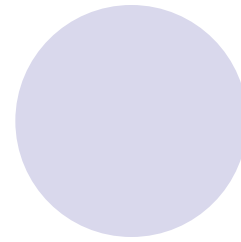
- Un control Caja de Imagen puede visualizar un gráfico a partir de un mapa de bits, un icono, un metarchivo, un metarchivo mejorado, un fichero *.jpeg* o un fichero *.gif*. Si el control no es lo suficientemente grande como para visualizar la imagen entera, ésta se recortará automáticamente al tamaño del control. Se utiliza para presentar gráficos, para actuar como contenedor de otros controles y para presentar el resultado de los métodos gráficos o texto con el método **Print**.



Visual Basic

Controles – Caja de Imagen (PictureBox).

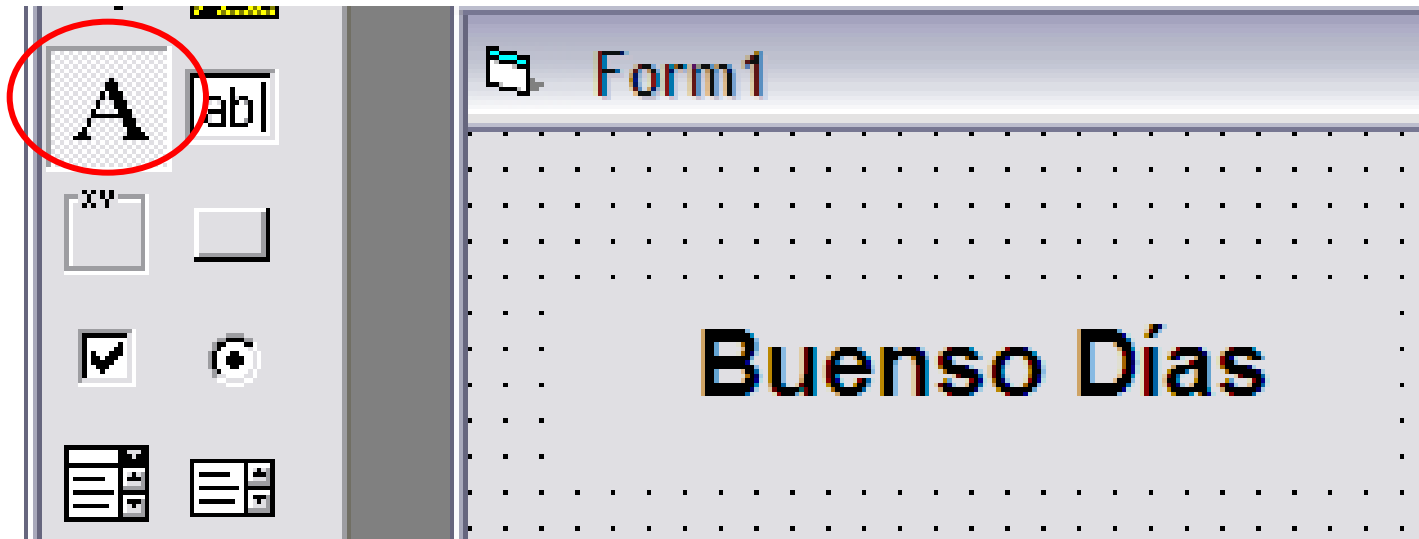
- **Procedimientos y Métodos:**
 - Cls, LoadPicture y Print.



Visual Basic

Controles – Etiqueta (Label).

- El control Etiqueta (**Label**), nos servirá para situar cualquier texto estático en el interior de un formulario, indicando posición, color, tipo de letra, etc. Que el texto sea estático quiere decir que el usuario del programa no podrá interactuar con él, modificándolo o borrándolo, ya que se trata de un texto fijo, sólo modificable desde el propio código o durante el diseño del formulario.



Visual Basic

Controles – Etiqueta (Label).

Nombre Propiedad	Descripción
(Nombre)	Será el nombre que utilizaremos para referirnos al control etiqueta en el código. No tiene nada que ver con la propiedad <i>Caption</i> .
Alignment	Sitúa el texto de la etiqueta dentro del marco de la etiqueta. El texto se puede ajustar a la izquierda (0 – Left Justify), a la derecha (1 – Right Justify) o al centro del marco de la etiqueta (2 – Center).
Appearance	Establece la apariencia que debe tener la etiqueta: 1 - 3D o 0 - Flat (“plano”).
Autosize	Ajusta el marco de la etiqueta (True) al texto de la etiqueta.
BackColor	Establece el color de fondo con el que la etiqueta presentará la información.
BackStyle	Especifica el estilo del marco de la etiqueta: 0 – Trasnparent (Marco de etiqueta transparente) y 1 – Opaque (Marco de etiqueta “Normal”).
BorderStyle	Especifica si el marco de la etiqueta va a aparecer en 3D (1 – Fixed Single) o “Normal” (0 – None).
Caption	Será el nombre que utilizaremos para referirnos al control en el formulario. No tiene nada que ver con la propiedad (<i>Nombre</i>).
Enabled	Si esta propiedad está desactivada (con el valor False), el usuario no podrá seleccionar este control. Éste aparecerá difuminado.
Font	Determina la fuente que se utilizará para representar el texto en la etiqueta.
ForeColor	Establece el color del “mensaje” (los caracteres que componen la etiqueta) de la etiqueta.

Visual Basic

Controles – Etiqueta (Label).

Nombre Propiedad	Descripción
Height	Determina la altura de la etiqueta.
Left	Determina la distancia que hay entre la esquina superior izquierda de la etiqueta y el límite izquierdo del contenedor de la misma (formulario).
Mouselcon	Presenta el cuadro de diálogo estándar <i>Abrir archivo</i> , que te permite seleccionar un archivo que contenga información adecuada para un puntero de ratón.
MousePointer	Presenta una lista con varios punteros para el ratón entre los que elegir. El último valor de la lista hará que el puntero sea el que se seleccione en la propiedad <i>Mouselcon</i> .
Tag	Esta propiedad es muy útil para que otros programadores entiendan como funciona tu programa, tanto si dejas el código para que alguien lo aprenda como si abandonas un proyecto y otras personas tienen que encargarse de él. La información que aparece en esta propiedad es meramente informativa, no afecta al comportamiento del control y no será vista por el usuario. Otra solución, quizás mejor, es comentar el código. Pero no podemos comentar un control.
Top	Determina la distancia que hay entre la esquina superior izquierda de la etiqueta y el límite superior del contenedor de la misma (formulario).
Visible	Si podemos ver un control cuando ejecutamos un programa, es que su propiedad <i>Visible</i> está a <i>True</i> .
Width	Determina la anchura de la etiqueta (en twips).

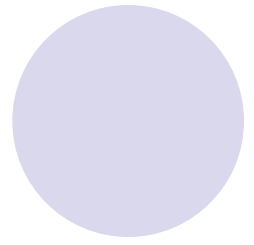
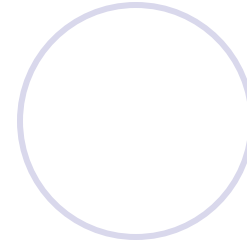
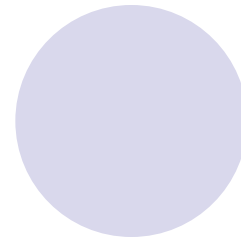
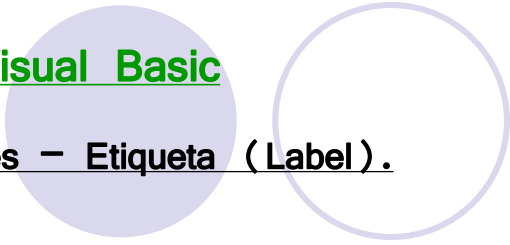
Visual Basic

Controles – Etiqueta (Label).

Nombre Evento	Descripción
Change	Este evento ocurre cada vez que la etiqueta sufre algún cambio.
Click	Este evento tiene lugar cuando hacemos click sobre la etiqueta.
DbIcClick	Este evento tiene lugar cuando hacemos doble-click sobre la etiqueta.
DragDrop	Este evento tiene lugar cuando se completa una operación de arrastrar y colocar, es decir, cuando se suelta el botón del ratón tras arrastrar un objeto hasta un control.
DragOver	Ese evento tiene lugar mientras se está llevando a cabo una operación de arrastrar y colocar. Podemos utilizar este evento para cambiar el puntero del ratón cuando éste se encuentre dentro de los límites de un determinado control (para indicar que el control no lo acepta, por ejemplo).
MouseDown	Este evento se lanza cuando pulsas un botón del ratón.
MouseMove	Este evento se lanza cuando movemos el ratón.
MouseUp	Este evento se lanza cuando soltamos el botón del ratón.

Visual Basic

Controles – Etiqueta (Label).

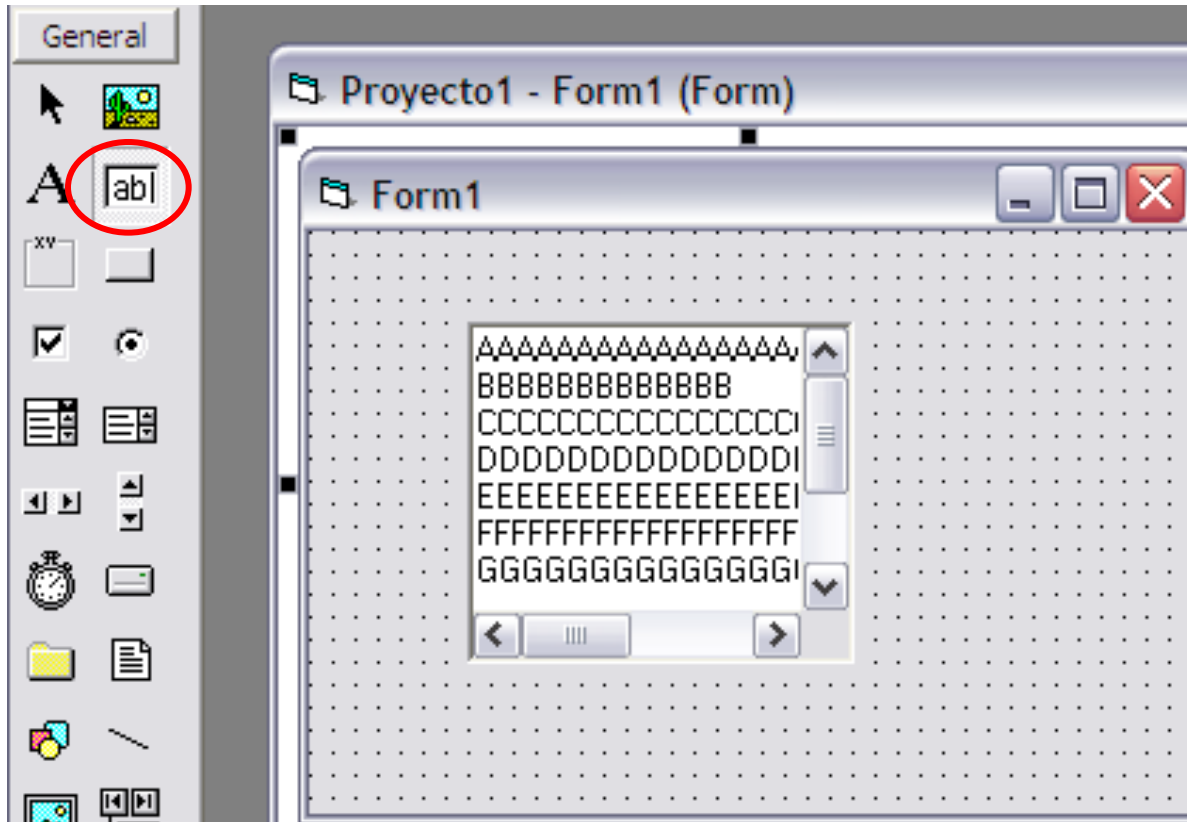


- **Procedimientos y Métodos:**
 - Se encauzan a través de la propiedad CAPTION.

Visual Basic

Controles – Cuadro de Texto (TextBox).

- El control Cuadro de Texto (**TextBox**), nos permitirá solicitar la entrada de datos en un formulario. Un control de este tipo puede tratar una o varias líneas de texto, incorporando funciones de selección, cortar y pegar, barras de desplazamiento para textos largos, etc.



Visual Basic

Controles – Cuadro de Texto (TextBox).

Nombre Propiedad	Descripción
(Nombre)	Será el nombre que utilizaremos para referirnos al control cuadro de texto en el código.
Alignment	Sitúa el texto del cuadro de texto dentro del marco del cuadro de texto. El texto se puede ajustar a la izquierda (0 – Left Justify), a la derecha (1 – Right Justify) o al centro del marco del cuadro de texto (2 – Center).
Appearance	Establece la apariencia que debe tener el cuadro de texto: 1 - 3D o 0 - Flat (“plano”).
BackColor	Establece el color de fondo con el que el cuadro de texto presentará la información.
BackStyle	Especifica el estilo del marco del cuadro de texto: 0 – Transparent (Marco del cuadro de texto transparente) y 1 – Opaque (Marco del cuadro de texto “Normal”).
BorderStyle	Especifica si el marco del cuadro de texto va a aparecer en 3D (1 – Fixed Single) o “Normal” (0 – None).
Enabled	Si esta propiedad está desactivada (con el valor False), el usuario no podrá seleccionar este control. Éste aparecerá difuminado.
Font	Determina la fuente que se utilizará para representar el texto en el cuadro de texto.
ForeColor	Establece el color del “mensaje” (los caracteres que componen el cuadro de texto) del cuadro de texto.
Height	Determina la altura del cuadro de texto.
Left	Determina la distancia que hay entre la esquina superior izquierda del cuadro de texto y el límite izquierdo del contenedor del mismo (formulario).

Visual Basic

Controles – Cuadro de Texto (TextBox).

Nombre PropiedadB	Descripción
MaxLenght	Máxima longitud de entrada de caracteres: 0 – No hay limitaciones, ≠ 0 – El valor que se establezca en MaxLenght.
Mouselcon	Presenta el cuadro de diálogo estándar <i>Abrir archivo</i> , que te permite seleccionar un archivo que contenga información adecuada para un puntero de ratón.
MousePointer	Presenta una lista con varios punteros para el ratón entre los que elegir. El último valor de la lista hará que el puntero sea el que se seleccione en la propiedad <i>Mouselcon</i> .
Multiline	Nos permite introducir más de una línea de texto (True) o no (False).
PasswordChar	Indica el carácter que vamos a utilizar para encriptar el texto.
ScrollBars	Barras de desplazamiento. Para activarla la propiedad <i>Multiline</i> tiene que estar a <i>True</i> : 0 – None (Ninguna), 1 – Horizontal, 2 – Vertical, 3 – Both (Ambas).
SelLenght	Longitud del texto seleccionado para copiarlo en un portapapeles (número de caracteres marcados). Un cuadro de dialogo dispone, automáticamente, sin que nosotros indiquemos nada, de: marcar texto, cortarlo, copiarlo y pegarlo del portapapeles. El cuadro de dialogo dispone, además, de una propiedades por medio de las cuales podremos conocer el texto que está marcado en cada momento, en qué carácter comienza el texto seleccionado y que longitud tiene (No está en la ventana de propiedades).
SelStart	Contiene un número indicando el carácter a partir del cual se ha empezado ha seleccionar le texto (No está en la ventana de propiedades).

Visual Basic

Controles – Cuadro de Texto (TextBox).

Nombre PropiedadB	Descripción
SelText	Contiene la cadena de texto que está seleccionada actualmente (No está en la ventana de propiedades).
Tag	Esta propiedad es muy útil para que otros programadores entiendan como funciona tu programa, tanto si dejas el código para que alguien lo aprenda como si abandonas un proyecto y otras personas tienen que encargarse de él. La información que aparece en esta propiedad es meramente informativa, no afecta al comportamiento del control y no será vista por el usuario. Otra solución, quizás mejor, es comentar el código. Pero no podemos comentar un control.
Text	El valor de esta propiedad será el texto que el cuadro de texto muestre al usuario. Si está definida la opción <i>Multiline</i> como <i>True</i> , para introducir el texto en líneas sucesivas, después de meter una línea. Hay que pulsar CTRL + ENTER y el cursor se colocará en la siguiente línea
Top	Determina la distancia que hay entre la esquina superior izquierda del cuadro de texto y el límite superior del contenedor de la misma (formulario).
Visible	Si podemos ver un control cuando ejecutamos un programa, es que su propiedad <i>Visible</i> está a <i>True</i> .
Width	Determina la anchura del cuadro de texto (en twips).

Visual Basic

Controles – Cuadro de Texto (TextBox).

Nombre Evento	Descripción
Change	Este evento ocurre cada vez que el cuadro de texto sufre algún cambio.
Click	Este evento tiene lugar cuando hacemos click sobre el cuadro de texto.
DbIcClick	Este evento tiene lugar cuando hacemos doble-click sobre el cuadro de texto.
DragDrop	Este evento tiene lugar cuando se completa una operación de arrastrar y colocar, es decir, cuando se suelta el botón del ratón tras arrastrar un objeto hasta un control.
DragOver	Ese evento tiene lugar mientras se está llevando a cabo una operación de arrastrar y colocar. Podemos utilizar este evento para cambiar el puntero del ratón cuando éste se encuentre dentro de los límites de un determinado control (para indicar que el control no lo acepta, por ejemplo).
GotFocus	Cuando el cuadro de texto recibe el enfoque se lanza este evento.
LostFocus	Si <i>GotFocus</i> se lanzaba cuando el cuadro de texto recibía el enfoque, <i>LostFocus</i> se lanza cuando lo pierde. El <i>enfoque</i> es la capacidad de recibir entradas del usuario a través del ratón o el teclado. Cuando un objeto tiene el enfoque, puede recibir entradas del usuario. Por ejemplo, cuando el cuadro de texto tiene el enfoque, el usuario puede escribir texto en él.
KeyDown	Cuando pulsas una tecla, se lanza este evento.
KeyPress	Cuando pulsas una tecla y la sueltas, se lanza este evento.

Visual Basic

Controles – Cuadro de Texto (TextBox).

Nombre Evento	Descripción
KeyUp	Cuando sueltas una tecla, se lanza este evento.
MouseDown	Este evento se lanza cuando pulsas un botón del ratón.
MouseMove	Este evento se lanza cuando movemos el ratón.
MouseUp	Este evento se lanza cuando soltamos el botón del ratón.

Visual Basic

Controles – Cuadro de Texto (TextBox).

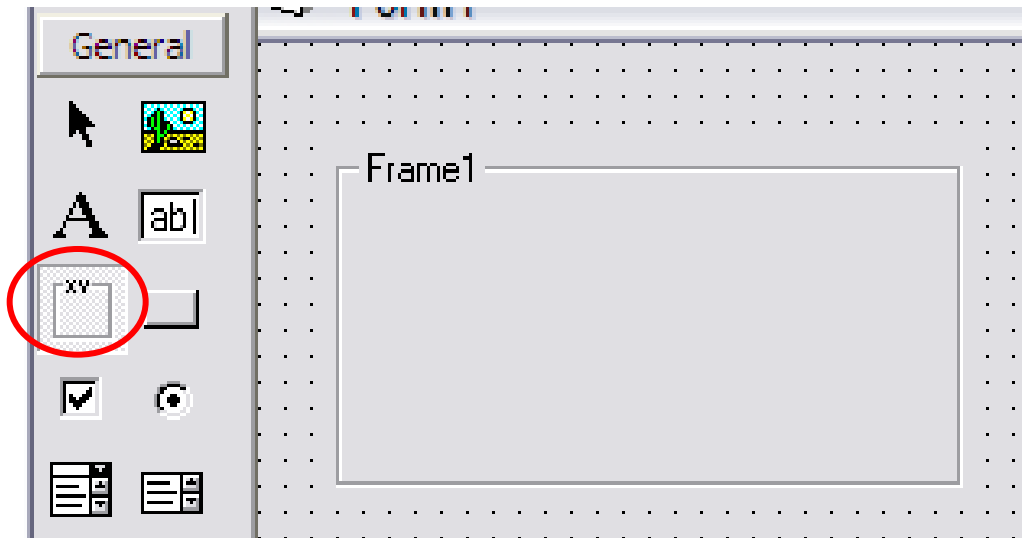
- **Procedimientos y Métodos:**

- **SetFocus:** Sitúa el cursor en un cuadro de texto específico.
[objeto].**SetFocus**

Visual Basic

Controles – Marco (Frame).

- Para crear un grupo de controles lo primero que hay que hacer es insertar el control **Frame**, dándole aproximadamente las dimensiones necesarias. A partir de aquí, cada vez que necesitemos un control deberemos crearlo directamente en el interior del **Frame**. Si creamos un control en cualquier otra zona del formulario y después lo movemos hasta el interior del **Frame**, ese control no formará realmente parte del grupo. Si intentamos moverlos todos veremos como aquellos que no se crearon dentro del grupo se quedan estáticos.



Visual Basic

Controles – Marco (Frame).

- En el caso en el que tengamos cualquier control fuera de un grupo y deseemos ponerlo dentro, existe una solución fácil. En lugar de volver a crearlo y, sobre todo, si ya se han definido para él propiedades o procedimientos, debemos seleccionar dicho control, copiarlo al portapapeles, por ejemplo, pulsando “**Control+Insert**”, borrarlo, pulsando “**Suprimir**”, seguidamente seleccionamos el control **Frame** al que va a pertenecer, pulsamos sobre él con el ratón y por último, pegamos en él el control con, por ejemplo, “**Mayúsculas+Insert**”.
- El control aparecerá en la esquina superior izquierda del marco, y desde ahí podremos moverlo a cualquier posición. Si ahora intentamos desplazar el grupo, comprobaremos como el control ya forma parte de él, y no se queda estático.

Visual Basic

Controles – Marco (Frame).

Nombre Propiedad	Descripción
(Nombre)	Será el nombre que utilizaremos para referirnos al control marco en el código. No tiene nada que ver con la propiedad <i>Caption</i> .
Appearance	Establece la apariencia que debe tener el marco: 1 - 3D o 0 - Flat (“plano”).
BackColor	Establece el color de fondo con el que el marco presentará la información.
BackStyle	Especifica el estilo del marco: 0 – Trasnparent (Marco transparente) y 1 – Opaque (Marco “Normal”).
BorderStyle	Especifica si el marco va a aparecer en 3D (1 – Fixed Single) o “Normal” (0 – None).
Caption	Será el nombre que utilizaremos para referirnos al marco en el formulario. No tiene nada que ver con la propiedad <i>(Nombre)</i> .
Enabled	Si esta propiedad está desactivada (con el valor False), el usuario no podrá seleccionar este control. Éste aparecerá difuminado. Todos los controles que contiene el marco se desactivan
Font	Determina la fuente que se utilizará para representar el texto en el marco.
ForeColor	Establece el color del “mensaje” del marco.
Height	Determina la altura del marco.
Left	Determina la distancia que hay entre la esquina superior izquierda del marco y el límite izquierdo del contenedor de la misma (formulario).
Mouselcon	Presenta el cuadro de diálogo estándar <i>Abrir archivo</i> , que te permite seleccionar un archivo que contenga información adecuada para un puntero de ratón.

Visual Basic

Controles – Marco (Frame).

Nombre Propiedad	Descripción
MousePointer	Presenta una lista con varios punteros para el ratón entre los que elegir. El último valor de la lista hará que el puntero sea el que se seleccione en la propiedad <i>MouseIcon</i> .
Tag	Esta propiedad es muy útil para que otros programadores entiendan como funciona tu programa, tanto si dejas el código para que alguien lo aprenda como si abandonas un proyecto y otras personas tienen que encargarse de él. La información que aparece en esta propiedad es meramente informativa, no afecta al comportamiento del control y no será vista por el usuario. Otra solución, quizás mejor, es comentar el código. Pero no podemos comentar un control.
Top	Determina la distancia que hay entre la esquina superior izquierda del marco y el límite superior del contenedor de la misma (formulario).
Visible	Si podemos ver un control cuando ejecutamos un programa, es que su propiedad <i>Visible</i> está a <i>True</i> .
Width	Determina la anchura del marco (en twips).

Visual Basic

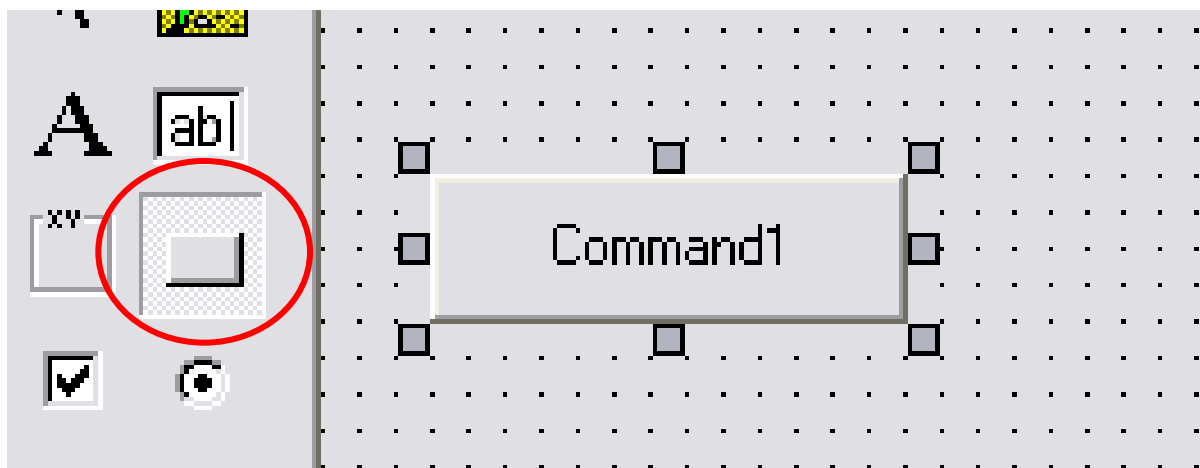
Controles – Marco (Frame).

Nombre Evento	Descripción
Click	Este evento tiene lugar cuando hacemos click sobre el marco.
DbIClick	Este evento tiene lugar cuando hacemos doble-click sobre el marco.
DragDrop	Este evento tiene lugar cuando se completa una operación de arrastrar y colocar, es decir, cuando se suelta el botón del ratón tras arrastrar un objeto hasta un control.
DragOver	Ese evento tiene lugar mientras se está llevando a cabo una operación de arrastrar y colocar. Podemos utilizar este evento para cambiar el puntero del ratón cuando éste se encuentre dentro de los límites de un determinado control (para indicar que el control no lo acepta, por ejemplo).
MouseDown	Este evento se lanza cuando pulsas un botón del ratón.
MouseMove	Este evento se lanza cuando movemos el ratón.
MouseUp	Este evento se lanza cuando soltamos el botón del ratón.

Visual Basic

Controles – Botón de Comando (CommandButton).

- Se trata, seguramente, del elemento más habitual en Windows. Un botón es un control que aparece como un rectángulo o cuadrado con un título en su interior, al pulsarlo el botón parece hundirse, y se ejecuta un código que es la acción asociada al botón (a su evento *Click*, en realidad). El título o texto que aparece en el interior del botón puede contener un carácter destacado (anteponiéndole en la propiedad **Captión** el carácter **&**), que servirá para pulsar el botón mediante la combinación de teclas **Alt + carácter_destacado** (*subrayado*).



Visual Basic

Controles – Botón de Comando (`CommandButton`).

Nombre Propiedad	Descripción
(Nombre)	Será el nombre que utilizaremos para referirnos al botón de comando en el código.
Appearance	Establece la apariencia que debe tener el botón de comando: 1 - 3D o 0 - Flat ("plano").
BackColor	Establece el color de fondo con el que el botón de comando presentará la información.
Cancel	Si <i>Cancel</i> está a <i>True</i> , al pulsar ESC actuará como si pulsáramos el botón de comando.
Caption	Será el nombre que utilizaremos para referirnos al botón de comando en el formulario. No tiene nada que ver con la propiedad (<i>Nombre</i>).
Default	Si <i>Default</i> está a <i>True</i> , al pulsar ENTER actuará como si pulsáramos el botón de comando.
Enabled	Si esta propiedad está desactivada (con el valor <i>False</i>), el usuario no podrá seleccionar este control. Éste aparecerá difuminado.
Font	Determina la fuente que se utilizará para representar el texto en el botón de comando.
Height	Determina la altura del botón de comando.
Left	Determina la distancia que hay entre la esquina superior izquierda del botón de comando y el límite izquierdo del contenedor del mismo (formulario).
Mouselcon	Presenta el cuadro de diálogo estándar <i>Abrir archivo</i> , que te permite seleccionar un archivo que contenga información adecuada para un puntero de ratón.

Visual Basic

Controles – Botón de Comando (`CommandButton`).

Nombre Propiedad	Descripción
MousePointer	Presenta una lista con varios punteros para el ratón entre los que elegir. El último valor de la lista hará que el puntero sea el que se seleccione en la propiedad <i>MouseIcon</i> .
Picture	Si un control tiene esta propiedad, puede mostrar una imagen como fondo.
TabIndex	Su valor nos indicará en que orden se irá accediendo a los distintos controles activos (con TAB o Mayúsculas+TAB).
TabStop	Si está a <i>True</i> , indica que este control está activo para utilizarlo con TabIndex.
Tag	Esta propiedad es muy útil para que otros programadores entiendan como funciona tu programa, tanto si dejas el código para que alguien lo aprenda como si abandonas un proyecto y otras personas tienen que encargarse de él. La información que aparece en esta propiedad es meramente informativa, no afecta al comportamiento del control y no será vista por el usuario. Otra solución, quizás mejor, es comentar el código. Pero no podemos comentar un control.
Top	Determina la distancia que hay entre la esquina superior izquierda del cuadro de texto y el límite superior del contenedor de la misma (formulario).
Visible	Si podemos ver un control cuando ejecutamos un programa, es que su propiedad <i>Visible</i> está a <i>True</i> .
Width	Determina la anchura del cuadro de texto (en twips).

Visual Basic

Controles – Botón de Comando (`CommandButton`).

Nombre Evento	Descripción
Click	Este evento tiene lugar cuando hacemos click sobre el botón de comando.
DragDrop	Este evento tiene lugar cuando se completa una operación de arrastrar y colocar, es decir, cuando se suelta el botón del ratón tras arrastrar un objeto hasta un control.
DragOver	Ese evento tiene lugar mientras se está llevando a cabo una operación de arrastrar y colocar. Podemos utilizar este evento para cambiar el puntero del ratón cuando éste se encuentre dentro de los límites de un determinado control (para indicar que el control no lo acepta, por ejemplo).
GotFocus	Cuando el botón de comando recibe el enfoque se lanza este evento.
LostFocus	Si <i>GotFocus</i> se lanzaba cuando el botón de comando recibía el enfoque, <i>LostFocus</i> se lanza cuando lo pierde. El <i>enfoque</i> es la capacidad de recibir entradas del usuario a través del ratón o el teclado. Cuando un objeto tiene el enfoque, puede recibir entradas del usuario. Por ejemplo, cuando el cuadro de texto tiene el enfoque, el usuario puede escribir texto en él.
KeyDown	Cuando pulsas una tecla, se lanza este evento.
KeyPress	Cuando pulsas una tecla y la sueltas, se lanza este evento.

Visual Basic

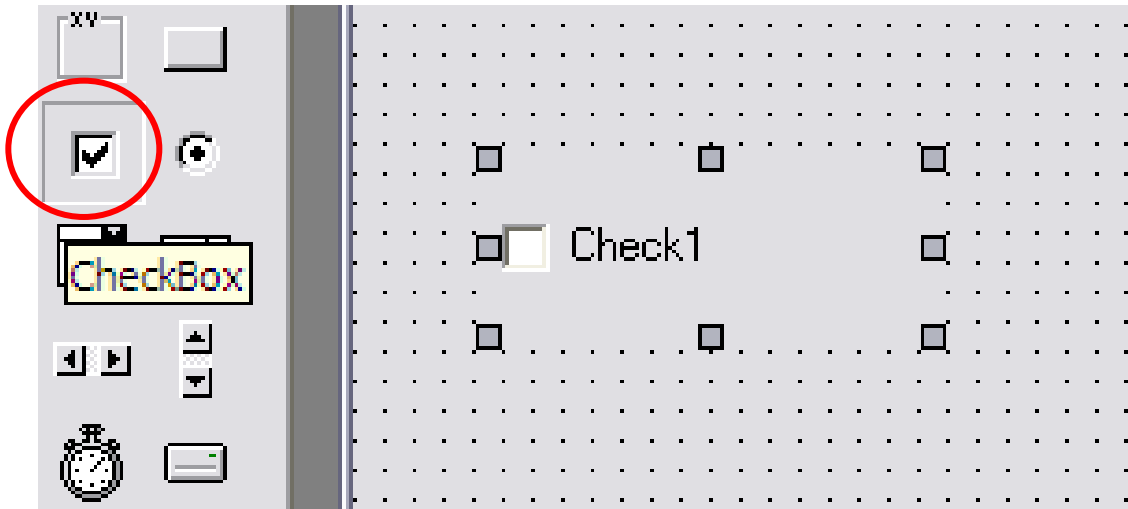
Controles – Botón de Comando (CommandButton).

Nombre Evento	Descripción
KeyUp	Cuando sueltas una tecla, se lanza este evento.
MouseDown	Este evento se lanza cuando pulsas un botón del ratón.
MouseMove	Este evento se lanza cuando movemos el ratón.
MouseUp	Este evento se lanza cuando soltamos el botón del ratón.

Visual Basic

Controles – Cuadro de Control (CheckBox).

- Es un control que indica si una opción particular está activada o desactivada. Cada Cuadro de Control es independiente de los demás, ya que cada uno de ellos tiene su propio nombre (propiedad Name). El número de opciones representadas de esta forma puede ser cualquiera y el usuario puede seleccionar todas las que quiera a la vez.



Visual Basic

Controles – Cuadro de Control (CheckBox).

- Un **CheckBox** almacena su estado actual en la propiedad **Value**, que por defecto es *vbUnchecked*, cuyo valor es 0. Por eso, cuando insertamos un control de este tipo aparece sin la cruz (☐), no está seleccionado. Cuando durante la ejecución del programa el usuario pulsa sobre el cuadro, lo que hace es alternar entre *vbUnchecked* y *vbChecked*, que es el valor que toma cuando aparece la cruz (☑)(cuyo valor es 1). Es posible modificar **Value** directamente desde el código de nuestro programa, y además de marcar y desmarcar también podemos utilizar el valor *vbGrayed*, que hará que el Cuadro de Control aparezca relleno e indisponible para que el usuario trabaje con él (☑) (cuyo valor es 2).

Visual Basic

Controles – Cuadro de Control (CheckBox).

Nombre Propiedad	Descripción
(Nombre)	Será el nombre que utilizaremos para referirnos al cuadro de control en el código.
Alignment	Sitúa el texto del cuadro de control dentro del marco del cuadro de control. El texto se puede ajustar a la izquierda (0 – Left Justify), a la derecha (1 – Right Justify) o al centro del marco del cuadro de control (2 – Center).
Appearance	Establece la apariencia que debe tener el cuadro de control: 1 - 3D o 0 - Flat (“plano”).
BackColor	Establece el color de fondo con el que el cuadro de control presentará la información.
Caption	Será el nombre que utilizaremos para referirnos al cuadro de control en el formulario. No tiene nada que ver con la propiedad (<i>Nombre</i>).
Enabled	Si esta propiedad está desactivada (con el valor False), el usuario no podrá seleccionar este control. Éste aparecerá difuminado.
Font	Determina la fuente que se utilizará para representar el texto en el cuadro de control.
Height	Determina la altura del cuadro de control.
Left	Determina la distancia que hay entre la esquina superior izquierda del cuadro de control y el límite izquierdo del contenedor del mismo (formulario).
Mouselcon	Presenta el cuadro de diálogo estándar <i>Abrir archivo</i> , que te permite seleccionar un archivo que contenga información adecuada para un puntero de ratón.

Visual Basic

Controles – Cuadro de Control (CheckBox).

Nombre Propiedad	Descripción
MousePointer	Presenta una lista con varios punteros para el ratón entre los que elegir. El último valor de la lista hará que el puntero sea el que se seleccione en la propiedad <i>MouseIcon</i> .
Picture	Si un control tiene esta propiedad, puede mostrar una imagen como fondo.
TabIndex	Su valor nos indicará en que orden se irá accediendo a los distintos controles activos (con TAB o Mayúsculas+TAB).
TabStop	Si está a <i>True</i> , indica que este control está activo para utilizarlo con TabIndex.
Tag	Esta propiedad es muy útil para que otros programadores entiendan como funciona tu programa, tanto si dejas el código para que alguien lo aprenda como si abandonas un proyecto y otras personas tienen que encargarse de él. La información que aparece en esta propiedad es meramente informativa, no afecta al comportamiento del control y no será vista por el usuario. Otra solución, quizás mejor, es comentar el código. Pero no podemos comentar un control.
Top	Determina la distancia que hay entre la esquina superior izquierda del cuadro de texto y el límite superior del contenedor de la misma (formulario).
Value	Es el valor que toma el cuadro de control. Puede tomar valor: 0 – UnChecked (Sin Marcar), 1 – Checked (Marcado) y 2 – Grayed (Gris – Deshabilitado).
Visible	Si podemos ver un control cuando ejecutamos un programa, es que su propiedad <i>Visible</i> está a <i>True</i> .
Width	Determina la anchura del cuadro de control (en twips).

Visual Basic

Controles – Cuadro de Control (CheckBox).

Nombre Evento	Descripción
Click	Este evento tiene lugar cuando hacemos click sobre el cuadro de control.
DragDrop	Este evento tiene lugar cuando se completa una operación de arrastrar y colocar, es decir, cuando se suelta el botón del ratón tras arrastrar un objeto hasta un control.
DragOver	Ese evento tiene lugar mientras se está llevando a cabo una operación de arrastrar y colocar. Podemos utilizar este evento para cambiar el puntero del ratón cuando éste se encuentre dentro de los límites de un determinado control (para indicar que el control no lo acepta, por ejemplo).
GotFocus	Cuando el cuadro de control recibe el enfoque se lanza este evento.
LostFocus	Si <i>GotFocus</i> se lanzaba cuando el cuadro de control recibía el enfoque, <i>LostFocus</i> se lanza cuando lo pierde. El <i>enfoque</i> es la capacidad de recibir entradas del usuario a través del ratón o el teclado. Cuando un objeto tiene el enfoque, puede recibir entradas del usuario. Por ejemplo, cuando el cuadro de texto tiene el enfoque, el usuario puede escribir texto en él.
KeyDown	Cuando pulsas una tecla, se lanza este evento.
KeyPress	Cuando pulsas una tecla y la sueltas, se lanza este evento.

Visual Basic

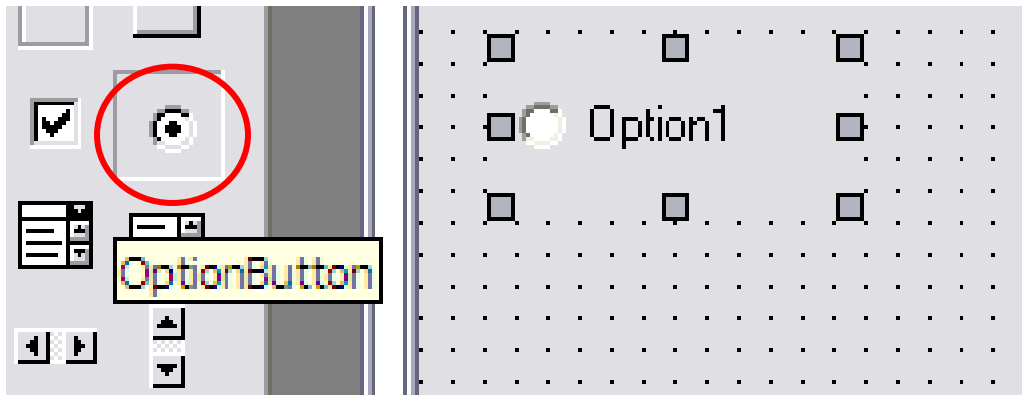
Controles – Cuadro de Control (CheckBox).

Nombre Evento	Descripción
KeyUp	Cuando sueltas una tecla, se lanza este evento.
MouseDown	Este evento se lanza cuando pulsas un botón del ratón.
MouseMove	Este evento se lanza cuando movemos el ratón.
MouseUp	Este evento se lanza cuando soltamos el botón del ratón.

Visual Basic

Controles – Botón de Opciones (OptionButton).

- Se utiliza cuando el usuario de un programa tiene que elegir sólo una opción de un conjunto de opciones y, que obligatoriamente, sea una.
- Operan en conjunto; cuando se activa un botón, los demás botones se desactivan.
- En este caso, **Value** sólo toma dos valores: *True* (botón seleccionado) o *False* (botón No seleccionado).



Visual Basic

Controles – Botón de Opciones (OptionButton).

Nombre Propiedad	Descripción
(Nombre)	Será el nombre que utilizaremos para referirnos al botón de opciones en el código.
Alignment	Sitúa el texto del botón de opciones dentro del marco del botón de opciones . El texto se puede ajustar a la izquierda (0 – Left Justify), a la derecha (1 – Right Justify) o al centro del marco del botón de opciones (2 – Center).
Appearance	Establece la apariencia que debe tener el botón de opciones: 1 - 3D o 0 - Flat (“plano”).
BackColor	Establece el color de fondo con el que el botón de opciones presentará la información.
Caption	Será el nombre que utilizaremos para referirnos al botón de opciones en el formulario. No tiene nada que ver con la propiedad (<i>Nombre</i>).
Enabled	Si esta propiedad está desactivada (con el valor False), el usuario no podrá seleccionar este control. Éste aparecerá difuminado.
Font	Determina la fuente que se utilizará para representar el texto en el botón de opciones.
Height	Determina la altura del botón de opciones.
Left	Determina la distancia que hay entre la esquina superior izquierda del botón de opciones y el límite izquierdo del contenedor del mismo (formulario).
Mouselcon	Presenta el cuadro de diálogo estándar <i>Abrir archivo</i> , que te permite seleccionar un archivo que contenga información adecuada para un puntero de ratón.
MousePointer	Presenta una lista con varios punteros para el ratón entre los que elegir. El último valor de la lista hará que el puntero sea el que se seleccione en la propiedad <i>Mouselcon</i> .

Visual Basic

Controles – Botón de Opciones (OptionButton).

Nombre Propiedad	Descripción
Picture	Si un control tiene esta propiedad, puede mostrar una imagen como fondo.
TabIndex	Su valor nos indicará en que orden se irá accediendo a los distintos controles activos (con TAB o Mayúsculas+TAB).
TabStop	Si está a <i>True</i> , indica que este control está activo para utilizarlo con TabIndex.
Tag	Esta propiedad es muy útil para que otros programadores entiendan como funciona tu programa, tanto si dejas el código para que alguien lo aprenda como si abandonas un proyecto y otras personas tienen que encargarse de él. La información que aparece en esta propiedad es meramente informativa, no afecta al comportamiento del control y no será vista por el usuario. Otra solución, quizás mejor, es comentar el código. Pero no podemos comentar un control.
Top	Determina la distancia que hay entre la esquina superior izquierda del botón de opciones y el límite superior del contenedor de la misma (formulario).
Value	Es el valor que toma el botón de opciones. Puede tomar valor: <i>True</i> – Marcado o <i>False</i> – No Marcado.
Visible	Si podemos ver un control cuando ejecutamos un programa, es que su propiedad <i>Visible</i> está a <i>True</i> .
Width	Determina la anchura del botón de opciones (en twips).

Visual Basic

Controles – Botón de Opciones (OptionButton).

Nombre Evento	Descripción
Click	Este evento tiene lugar cuando hacemos click sobre el botón de comando.
DbIClick	Este evento tiene lugar cuando hacemos doble-click sobre el botón de comando.
DragDrop	Este evento tiene lugar cuando se completa una operación de arrastrar y colocar, es decir, cuando se suelta el botón del ratón tras arrastrar un objeto hasta un control.
DragOver	Ese evento tiene lugar mientras se está llevando a cabo una operación de arrastrar y colocar. Podemos utilizar este evento para cambiar el puntero del ratón cuando éste se encuentre dentro de los límites de un determinado control (para indicar que el control no lo acepta, por ejemplo).
GotFocus	Cuando el botón de comando recibe el enfoque se lanza este evento.
LostFocus	Si <i>GotFocus</i> se lanzaba cuando el botón de comando recibía el enfoque, <i>LostFocus</i> se lanza cuando lo pierde. El <i>enfoque</i> es la capacidad de recibir entradas del usuario a través del ratón o el teclado. Cuando un objeto tiene el enfoque, puede recibir entradas del usuario. Por ejemplo, cuando el cuadro de texto tiene el enfoque, el usuario puede escribir texto en él.
KeyDown	Cuando pulsas una tecla, se lanza este evento.
KeyPress	Cuando pulsas una tecla y la sueltas, se lanza este evento.

Visual Basic

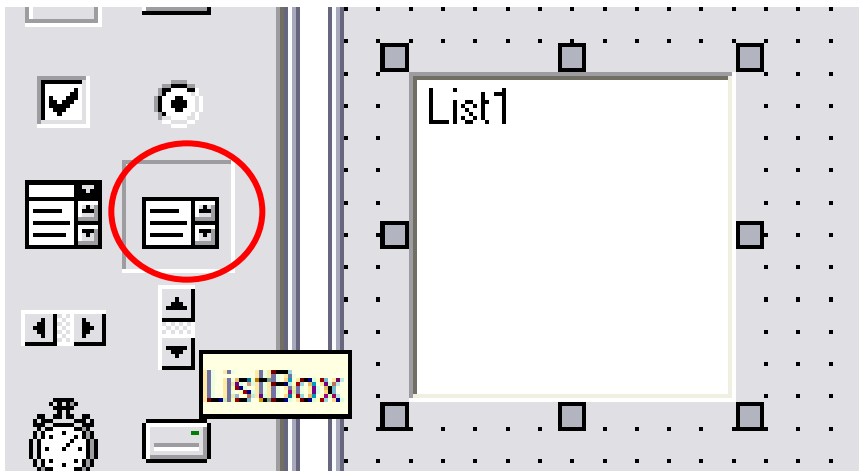
Controles – Botón de Opciones (OptionButton).

Nombre Evento	Descripción
KeyUp	Cuando sueltas una tecla, se lanza este evento.
MouseDown	Este evento se lanza cuando pulsas un botón del ratón.
MouseMove	Este evento se lanza cuando movemos el ratón.
MouseUp	Este evento se lanza cuando soltamos el botón del ratón.

Visual Basic

Controles – Cuadro de Listas (ListBox).

- Presentan una lista de elementos entre los que el usuario puede seleccionar uno o varios. Los Cuadros de Listas presentan al usuario una lista de opciones. De forma predeterminada, las opciones se presentan verticalmente en una única columna, aunque también se pueden establecer múltiples columnas. Si el número de elementos excede de los que se pueden presentar en él, aparecen automáticamente barras de desplazamiento. El usuario puede seleccionar una entrada de la lista haciendo click sobre ella.



Visual Basic

Controles – Cuadro de Listas (ListBox).

Nombre Propiedad	Descripción
(Nombre)	Será el nombre que utilizaremos para referirnos al cuadro de listas en el código.
Appearance	Establece la apariencia que debe tener el cuadro de listas: 1 - 3D o 0 - Flat (“plano”).
BackColor	Establece el color de fondo con el que el botón de opciones presentará la información.
Columns	Muestra los elementos de la lista en tantas columnas como aquí se indiquen.
Enabled	Si esta propiedad está desactivada (con el valor False), el usuario no podrá seleccionar este control. Éste aparecerá difuminado.
Font	Determina la fuente que se utilizará para representar el texto en el cuadro de listas.
Height	Determina la altura del cuadro de listas.
Left	Determina la distancia que hay entre la esquina superior izquierda del cuadro de listas y el límite izquierdo del contenedor del mismo (formulario).
List	Contiene la matriz de todos los valores (elementos) almacenados en el cuadro de listas (CTRL + ENTER) – <i>objeto.List (índice)</i> .
ListCount	Contiene el número de elementos del cuadro de listas.
ListIndex	Contiene el número de orden del elemento cuadro de listas dentro de la lista.
Mouselcon	Presenta el cuadro de diálogo estándar <i>Abrir archivo</i> , que te permite seleccionar un archivo que contenga información adecuada para un puntero de ratón.

Visual Basic

Controles – Cuadro de Listas (ListBox).

Nombre Propiedad	Descripción
MousePointer	Presenta una lista con varios punteros para el ratón entre los que elegir. El último valor de la lista hará que el puntero sea el que se seleccione en la propiedad <i>MouseIcon</i> .
MultiSelect	Controla cuantos elementos se pueden seleccionar. No puede modificarse en tiempo de ejecución. Los valores son: 0 – None, 1 – Simple y 2 – Extended.
Selected	Es una matriz que contiene todos los elementos de la lista: los que están seleccionados (True) y los que no están seleccionados (False).
Sorted	Ordena una lista de elementos del cuadro de lista alfabéticamente (cuando está a True). No puede modificarse en tiempo de ejecución.
TabIndex	Su valor nos indicará en que orden se irá accediendo a los distintos controles activos (con TAB o Mayúsculas+TAB).
TabStop	Si está a <i>True</i> , indica que este control está activo para utilizarlo con TabIndex.
Tag	Esta propiedad es muy útil para que otros programadores entiendan como funciona tu programa, tanto si dejas el código para que alguien lo aprenda como si abandonas un proyecto y otras personas tienen que encargarse de él. La información que aparece en esta propiedad es meramente informativa, no afecta al comportamiento del control y no será vista por el usuario. Otra solución, quizás mejor, es comentar el código. Pero no podemos comentar un control.
Text	Esta propiedad, que no puede modificarse directamente, contiene el texto del elemento más recientemente seleccionado.

Visual Basic

Controles – Cuadro de Listas (ListBox).

Nombre Propiedad	Descripción
Top	Determina la distancia que hay entre la esquina superior izquierda del cuadro de listas y el límite superior del contenedor de la misma (formulario).
Visible	Si podemos ver un control cuando ejecutamos un programa, es que su propiedad <i>Visible</i> está a <i>True</i> .
Width	Determina la anchura del cuadro de listas (en twips).

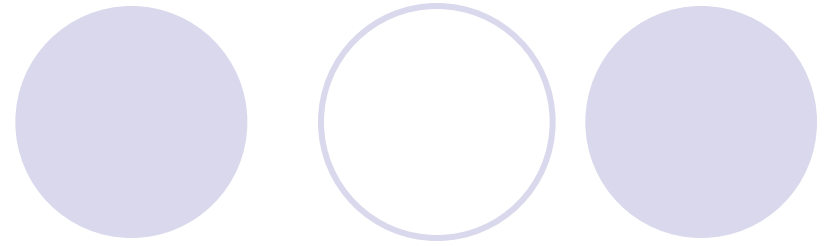
Controles – Cuadro de Listas (ListBox).

Nombre Evento	Descripción
Click	Este evento tiene lugar cuando hacemos click sobre el cuadro de lista.
DbIClick	Este evento tiene lugar cuando hacemos doble-click sobre el cuadro de lista.
DragDrop	Este evento tiene lugar cuando se completa una operación de arrastrar y colocar, es decir, cuando se suelta el botón del ratón tras arrastrar un objeto hasta un control.
DragOver	Ese evento tiene lugar mientras se está llevando a cabo una operación de arrastrar y colocar. Podemos utilizar este evento para cambiar el puntero del ratón cuando éste se encuentre dentro de los límites de un determinado control (para indicar que el control no lo acepta, por ejemplo).
GotFocus	Cuando el cuadro de lista recibe el enfoque se lanza este evento.
LostFocus	Si <i>GotFocus</i> se lanzaba cuando el botón de comando recibía el enfoque, <i>LostFocus</i> se lanza cuando lo pierde. El <i>enfoque</i> es la capacidad de recibir entradas del usuario a través del ratón o el teclado. Cuando un objeto tiene el enfoque, puede recibir entradas del usuario. Por ejemplo, cuando el cuadro de texto tiene el enfoque, el usuario puede escribir texto en él.
KeyDown	Cuando pulsas una tecla, se lanza este evento.
KeyPress	Cuando pulsas una tecla y la sueltas, se lanza este evento.

Controles – Cuadro de Listas (ListBox).

Nombre Evento	Descripción
KeyUp	Cuando sueltas una tecla, se lanza este evento.
MouseDown	Este evento se lanza cuando pulsas un botón del ratón.
MouseMove	Este evento se lanza cuando movemos el ratón.
MouseUp	Este evento se lanza cuando soltamos el botón del ratón.

Controles – Cuadro de Listas (ListBox).

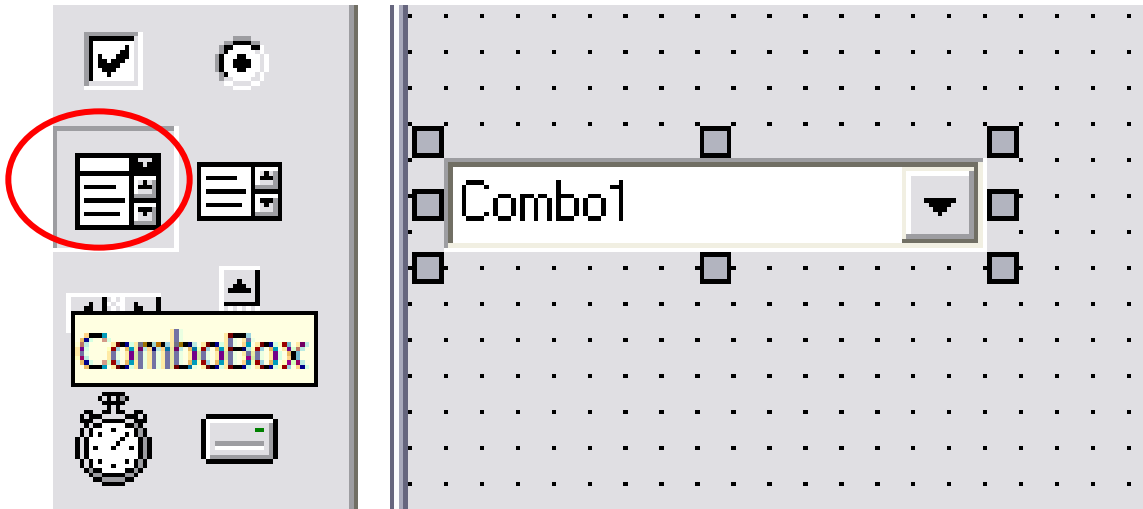


- **Procedimientos y Métodos:**
- Permite modificar el contenido del cuadro de una lista durante la ejecución del programa.
 - **AddItem** (Añadir un elemento): Añade un elemento a una lista determinada.
 - Formato:
Objeto.**AddItem** "texto" [,índice]
 - Si **Sorted** = True → No poner *índice*.
 - Si no hay *índice* → El elemento se inserta ("añade") en el último lugar.
 - El *índice* empieza por cero.
 - **Clear** (Borrar): Borra todos los elementos de una lista determinada.
 - Formato:
Objeto.**Clear**
 - **RemoveItem** (Eliminar un elemento): Permite eliminar un elemento de una lista determinada.
 - Formato:
Objeto.**RemoveItem** *índice*

Visual Basic

Controles – Listas Combinadas (ComboBox).

- El control **ComboBox** combina las características de un Cuadro de Texto y un Cuadro de Listas. Este control permite que el usuario seleccione un elemento; para ello, deberemos escribir el texto en el cuadro de texto y lo seleccionaremos después en la lista.



Visual Basic

Controles – Listas Combinadas (ComboBox).

Nombre Propiedad	Descripción
(Nombre)	Será el nombre que utilizaremos para referirnos a la lista combinadas en el código.
Appearance	Establece la apariencia que debe tener la lista combinada: 1 - 3D o 0 - Flat (“plano”).
BackColor	Establece el color de fondo con el que la lista combinada presentará la información.
Enabled	Si esta propiedad está desactivada (con el valor False), el usuario no podrá seleccionar este control. Éste aparecerá difuminado.
Font	Determina la fuente que se utilizará para representar el texto en el cuadro de listas.
Height	Determina la altura de la lista combinada.
Left	Determina la distancia que hay entre la esquina superior izquierda de la lista combinada y el límite izquierdo del contenedor del mismo (formulario).
List	Contiene la matriz de todos los valores (elementos) almacenados en la lista combinada (CTRL + ENTER) – <i>objeto.List (índice)</i> .
ListCount	Contiene el número de elementos de la lista combinada.
ListIndex	Contiene el número de orden del elemento de la lista combinada dentro de la lista.
Mouselcon	Presenta el cuadro de diálogo estándar <i>Abrir archivo</i> , que te permite seleccionar un archivo que contenga información adecuada para un puntero de ratón.

Visual Basic

Controles – Listas Combinadas (ComboBox).

Nombre Propiedad	Descripción
MousePointer	Presenta una lista con varios punteros para el ratón entre los que elegir. El último valor de la lista hará que el puntero sea el que se seleccione en la propiedad <i>MouseIcon</i> .
Selected	Es una matriz que contiene todos los elementos de la lista: los que están seleccionados (True) y los que no están seleccionados (False).
Sorted	Ordena una lista de elementos de la lista combinada alfabéticamente (cuando está a True). No puede modificarse en tiempo de ejecución.
Style	Nos permite elegir entre 3 tipos distintos de listas combinadas: 0 – DropDown Combo (Una lista combinada que aparece inicialmente como una caja de texto, en la que el usuario puede introducir información, con una flecha a su derecha, en la que pulsando desplegaremos una lista de la que podremos seleccionar un elemento). 1 – Simple Combo (Igual que la anterior, con la diferencia de que la lista NO es desplegable , sino que aparece siempre abierta). 2 – DropDown List (Aparece como una etiqueta de texto, en la que se muestra el elemento actualmente seleccionado de una lista. Es posible abrir la lista para seleccionar cualquier otro, pulsando la flecha que hay a la derecha, pero no es posible teclear información directamente, al no existir una caja de texto).
TabIndex	Su valor nos indicará en que orden se irá accediendo a los distintos controles activos (con TAB o Mayúsculas+TAB).
TabStop	Si está a <i>True</i> , indica que este control está activo para utilizarlo con TabIndex.

Visual Basic

Controles – Listas Combinadas (ComboBox).

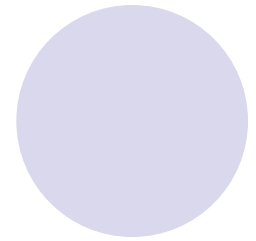
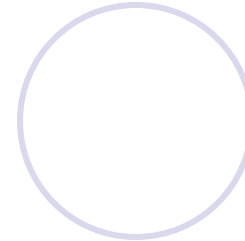
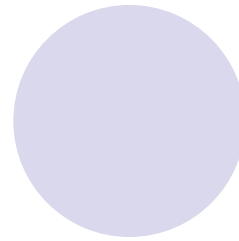
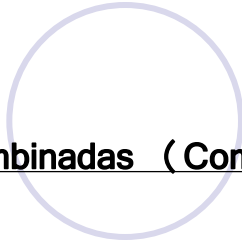
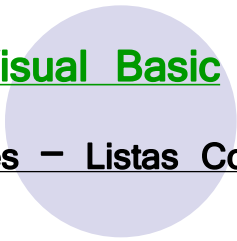
Nombre Propiedad	Descripción
Tag	Esta propiedad es muy útil para que otros programadores entiendan como funciona tu programa, tanto si dejas el código para que alguien lo aprenda como si abandonas un proyecto y otras personas tienen que encargarse de él. La información que aparece en esta propiedad es meramente informativa, no afecta al comportamiento del control y no será vista por el usuario. Otra solución, quizás mejor, es comentar el código. Pero no podemos comentar un control.
Text	Esta propiedad contiene el texto del elemento más recientemente seleccionado.
Top	Determina la distancia que hay entre la esquina superior izquierda de la lista combinada y el límite superior del contenedor de la misma (formulario).
Visible	Si podemos ver un control cuando ejecutamos un programa, es que su propiedad <i>Visible</i> está a <i>True</i> .
Width	Determina la anchura de la lista combinada (en twips).

Controles – Listas Combinadas (ComboBox).

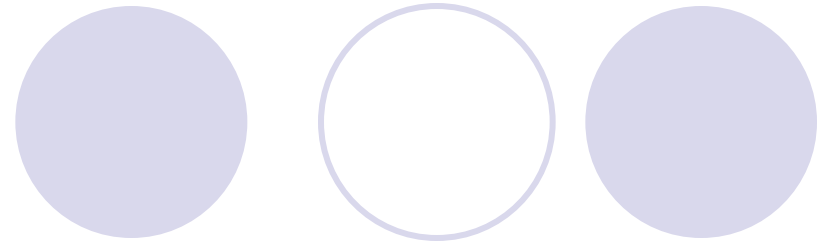
Nombre Evento	Descripción
Click	Este evento tiene lugar cuando hacemos click sobre la lista combinada.
Change	Este evento ocurre cada vez que la lista combinada sufre algún cambio.
DbIcClick	Este evento tiene lugar cuando hacemos doble-click sobre la lista combinada.
DragDrop	Este evento tiene lugar cuando se completa una operación de arrastrar y colocar, es decir, cuando se suelta el botón del ratón tras arrastrar un objeto hasta un control.
DragOver	Ese evento tiene lugar mientras se está llevando a cabo una operación de arrastrar y colocar. Podemos utilizar este evento para cambiar el puntero del ratón cuando éste se encuentre dentro de los límites de un determinado control (para indicar que el control no lo acepta, por ejemplo).
GotFocus	Cuando la lista combinada recibe el enfoque se lanza este evento.
LostFocus	Si <i>GotFocus</i> se lanzaba cuando el botón de comando recibía el enfoque, <i>LostFocus</i> se lanza cuando lo pierde. El <i>enfoque</i> es la capacidad de recibir entradas del usuario a través del ratón o el teclado. Cuando un objeto tiene el enfoque, puede recibir entradas del usuario. Por ejemplo, cuando el cuadro de texto tiene el enfoque, el usuario puede escribir texto en él.
KeyDown	Cuando pulsas una tecla, se lanza este evento.
KeyPress	Cuando pulsas una tecla y la sueltas, se lanza este evento.

Visual Basic

Controles – Listas Combinadas (ComboBox).



Nombre Evento	Descripción
KeyUp	Cuando sueltas una tecla, se lanza este evento.



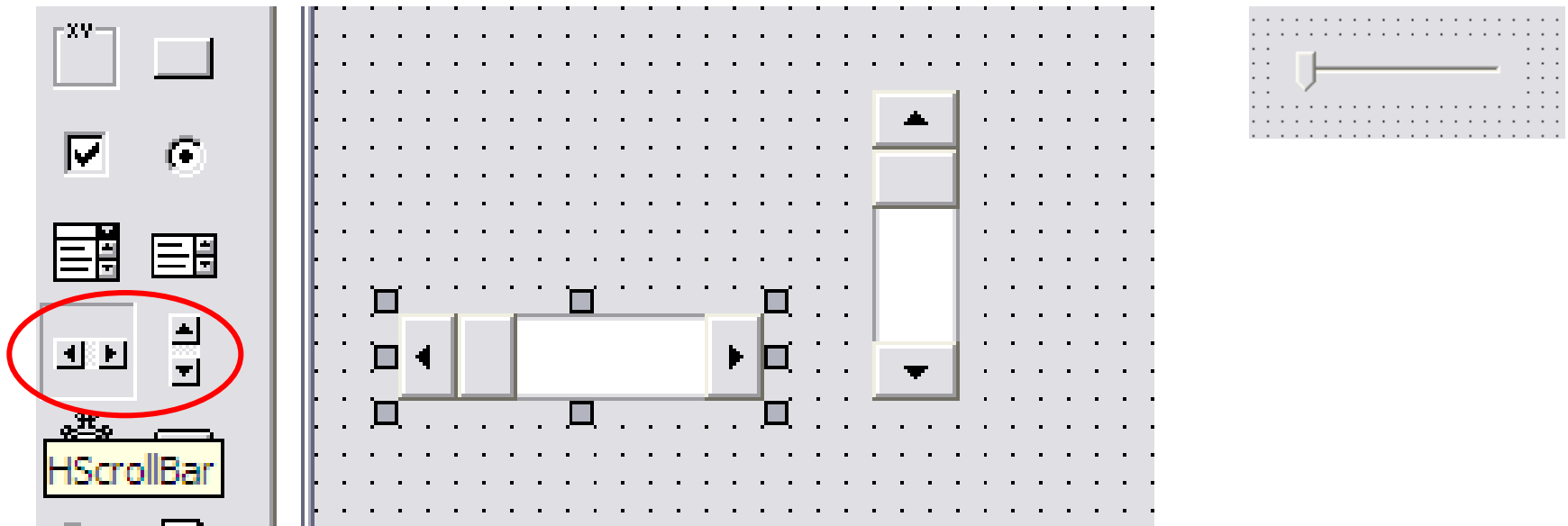
- **Procedimientos y Métodos:**

- Permite modificar el contenido de una lista combinada durante la ejecución del programa.
 - **AddItem** (Añadir un elemento): Añade un elemento a una lista determinada.
 - Formato:
Objeto.**AddItem** “texto” [,índice]
 - Si **Sorted** = True → No poner *índice*.
 - Si no hay *índice* → El elemento se inserta (“añade”) en el último lugar.
 - El *índice* empieza por cero.
 - **Clear** (Borrar): Borra todos los elementos de una lista determinada.
 - Formato:
Objeto.**Clear**
 - **RemoveItem** (Eliminar un elemento): Permite eliminar un elemento de una lista determinada.
 - Formato:
Objeto.**RemoveItem** *índice*

Visual Basic

Controles – Barras de desplazamiento Horizontales y Verticales (HScrollBar y VScrollBar).

- Informan de la posición del cuadro de desplazamiento dentro de la barra. El usuario tiene control sobre el rango de la barra de desplazamiento y sobre los incrementos en los que el cuadro de desplazamiento puede avanzar. Actualmente se recomienda más utilizar Sliders como dispositivos de entrada en lugar de las ScrollBars.



Visual Basic

Controles – Barras de desplazamiento Horizontales y Verticales (HScrollBar y VScrollBar).

Nombre Propiedad	Descripción
(Nombre)	Será el nombre que utilizaremos para referirnos a los scrolls en el código.
Enabled	Si esta propiedad está desactivada (con el valor False), el usuario no podrá seleccionar este control. Éste aparecerá difuminado.
Font	Determina la fuente que se utilizará para representar el texto en el cuadro de listas.
Height	Determina la altura de la barra de desplazamiento.
LargeChange	Valor que se le suma a “Value” cuando se hace click dentro de la Barra de Desplazamiento.
Left	Determina la distancia que hay entre la esquina superior izquierda de la barra de desplazamiento y el límite izquierdo del contenedor del mismo (formulario).
Max	El valor máximo que puede tomar el Cuadro de Desplazamiento dentro de la Barra de Desplazamiento ($-32768 \leq \text{valor} \leq 32767$).
Min	El valor mínimo que puede tomar el Cuadro de Desplazamiento dentro de la Barra de Desplazamiento ($-32768 \leq \text{valor} \leq 32767$).
Mouselcon	Presenta el cuadro de diálogo estándar <i>Abrir archivo</i> , que te permite seleccionar un archivo que contenga información adecuada para un puntero de ratón.

Visual Basic

Controles – Barras de desplazamiento Horizontales y Verticales (HScrollBar y VScrollBar).

Nombre Propiedad	Descripción
MousePointer	Presenta una lista con varios punteros para el ratón entre los que elegir. El último valor de la lista hará que el puntero sea el que se seleccione en la propiedad <i>MouseIcon</i> .
SmallChange	Valor que se le suma a “Value” cuando se hace click en una de las Flechas de Desplazamiento.
TabIndex	Su valor nos indicará en que orden se irá accediendo a los distintos controles activos (con TAB o Mayúsculas+TAB).
TabStop	Si está a <i>True</i> , indica que este control está activo para utilizarlo con TabIndex.
Tag	Esta propiedad es muy útil para que otros programadores entiendan como funciona tu programa, tanto si dejas el código para que alguien lo aprenda como si abandonas un proyecto y otras personas tienen que encargarse de él. La información que aparece en esta propiedad es meramente informativa, no afecta al comportamiento del control y no será vista por el usuario. Otra solución, quizás mejor, es comentar el código. Pero no podemos comentar un control.
Top	Determina la distancia que hay entre la esquina superior izquierda de la barra de desplazamiento y el límite superior del contenedor de la misma (formulario).
Value	Número que representa la posición actual del Cuadro de Desplazamiento en el interior de la Barra de Desplazamiento ($\text{Min} \leq \text{valor} \leq \text{Max}$).
Visible	Si podemos ver un control cuando ejecutamos un programa, es que su propiedad <i>Visible</i> está a <i>True</i> .
Width	Determina la anchura de la barra de desplazamiento (en twips).

Visual Basic

Controles – Barras de desplazamiento Horizontales y Verticales (HScrollBar y VScrollBar).

Nombre Evento	Descripción
Change	Se produce después de que se haya modificado la posición del cuadro de desplazamiento.
DragDrop	Este evento tiene lugar cuando se completa una operación de arrastrar y colocar, es decir, cuando se suelta el botón del ratón tras arrastrar un objeto hasta un control.
DragOver	Ese evento tiene lugar mientras se está llevando a cabo una operación de arrastrar y colocar. Podemos utilizar este evento para cambiar el puntero del ratón cuando éste se encuentre dentro de los límites de un determinado control (para indicar que el control no lo acepta, por ejemplo).
GotFocus	Cuando la lista combinada recibe el enfoque se lanza este evento.
LostFocus	Si <i>GotFocus</i> se lanzaba cuando el botón de comando recibía el enfoque, <i>LostFocus</i> se lanza cuando lo pierde. El <i>enfoque</i> es la capacidad de recibir entradas del usuario a través del ratón o el teclado. Cuando un objeto tiene el enfoque, puede recibir entradas del usuario. Por ejemplo, cuando el cuadro de texto tiene el enfoque, el usuario puede escribir texto en él.
KeyDown	Cuando pulsas una tecla, se lanza este evento.
KeyPress	Cuando pulsas una tecla y la sueltas, se lanza este evento.

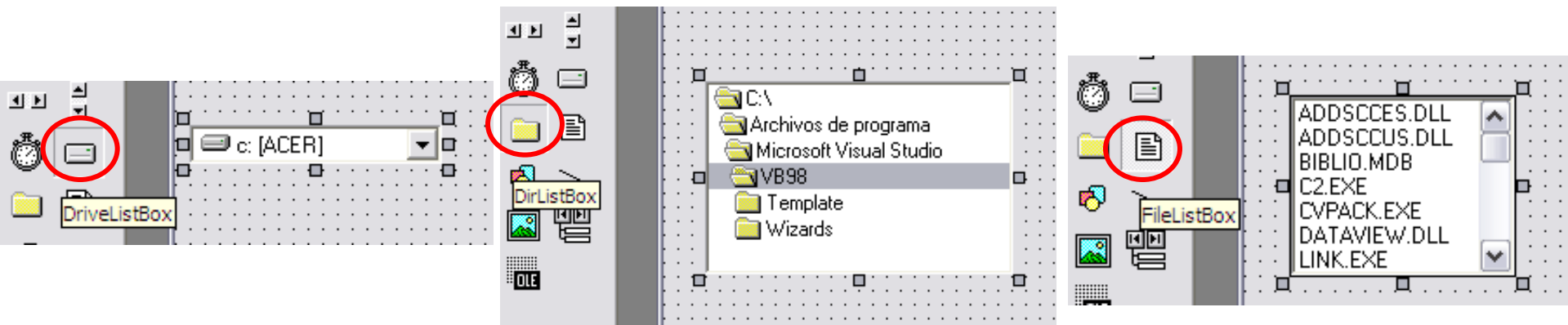
Controles – Barras de desplazamiento Horizontales y Verticales (HScrollBar y VScrollBar).

Nombre Evento	Descripción
KeyUp	Cuando sueltas una tecla, se lanza este evento.
Scroll	Se produce mientras que el Cuadro de Desplazamiento está siendo arrastrado por el interior de la Barra de Desplazamiento.

Visual Basic

Controles – Controles del Sistema de Archivos (Cuadro de Lista de Unidades (DriveListBox), Cuadro de Lista de Directorios (DirListBox) y Cuadro de Lista de Ficheros (FileListBox)).

- Con DriveListBox obtendremos una lista combinada, que al desplegarse nos permite seleccionar entre las distintas unidades que tenga el sistema.
- Con DirListBox obtenemos una lista de directorios en forma de árbol, pudiendo cerrar y abrir varios niveles de forma similar al Administrador de archivos o explorador.
- Con FileListBox obtenemos una lista de archivos, que contiene sólo su nombre y su extensión.



Visual Basic

Controles – Controles del Sistema de Archivos (Cuadro de Lista de Unidades (DriveListBox), Cuadro de Lista de Directorios (DirListBox) y Cuadro de Lista de Ficheros (FileListBox)).

Nombre Propiedad	Descripción
(Nombre)	Será el nombre que utilizaremos para referirnos a los cuadros de lista en el código.
Appearance	Establece la apariencia que debe tener el cuadros de listas: 1 - 3D o 0 - Flat (“plano”).
Archive	Si esta propiedad está a True, en el objeto FileListBox se mostrarán los “ <i>archivos modificados</i> ”.
BackColor	Establece el color de fondo con el que el botón de opciones presentará la información.
ChDrive	Podemos cambiar las unidades de DriveListBox a nivel de sistema operativo si especificamos la propiedad Drive como argumento de la instrucción ChDrive (ChDrive Objeto.Drive).
Drive	Podemos tanto obtener la selección actual como modificarla, facilitando la letra de la nueva unidad en DriveListBox – Objeto.Drive.
Enabled	Si esta propiedad está desactivada (con el valor False), el usuario no podrá seleccionar este control. Éste aparecerá difuminado.
FileDateTime	Obtiene la fecha y hora de creación, o última modificación del fichero seleccionado en la lista FileListBox.
FileLen	Obtiene el tamaño en bytes del archivo seleccionado en la lista FileListBox.
FileName	Contiene el nombre del archivo seleccionado en la lista FileListBox.
Font	Determina la fuente que se utilizará para representar el texto en el cuadro de lista.

Visual Basic

Controles – Controles del Sistema de Archivos (Cuadro de Lista de Unidades (DriveListBox), Cuadro de Lista de Directorios (DirListBox) y Cuadro de Lista de Ficheros (FileListBox)).

Nombre Propiedad	Descripción
GetAttr	Nos permite obtener los atributos del archivo seleccionado en la lista FileListBox.
Height	Determina la altura del cuadro de lista.
Hidden	Si esta propiedad está a True, en el objeto FileListBox se mostrarán los “ <i>archivos ocultos</i> ”.
Left	Determina la distancia que hay entre la esquina superior izquierda del cuadro de lista y el límite izquierdo del contenedor del mismo (formulario).
List	Contiene la matriz de todos los valores (elementos) almacenados en el cuadro de lista (CTRL + ENTER) – <i>objeto.List (índice)</i> .
ListCount	Contiene el número de elementos del cuadro de lista.
ListIndex	Contiene el número de orden del elemento cuadro de lista dentro de la lista.
Mouselcon	Presenta el cuadro de diálogo estándar <i>Abrir archivo</i> , que te permite seleccionar un archivo que contenga información adecuada para un puntero de ratón.
MousePointer	Presenta una lista con varios punteros para el ratón entre los que elegir. El último valor de la lista hará que el puntero sea el que se seleccione en la propiedad <i>Mouselcon</i> .

Visual Basic

Controles – Controles del Sistema de Archivos (Cuadro de Lista de Unidades (DriveListBox), Cuadro de Lista de Directorios (DirListBox) y Cuadro de Lista de Ficheros (FileListBox)).

Nombre Propiedad	Descripción
MultiSelect	Controla cuantos elementos se pueden seleccionaren en FileListBox. No puede modificarse en tiempo de ejecución. Los valores son: 0 – None, 1 – Simple y 2 – Extended.
Normal	Si esta propiedad está a True, en el objeto FileListBox se mostrarán los “archivos normales”.
Path	Permite obtener el camino actual completo, así como modificarlo en DirListBox. El directorio especificado por la propiedad Path (Objeto.Path) siempre tiene el valor -1 en ListIndex. El directorio inmediatamente superior a él tiene un valor de -2 en ListIndex, el superior a éste tiene -3 y así sucesivamente hasta el directorio raíz. El primer subdirectorio de Objeto.Path tiene un valor de 0 en ListIndex. Si hay múltiples directorios en el nivel del primer subdirectorio, el siguiente tiene un valor de 1 en ListIndex; si hay más en este nivel ListIndex tomaría el valor 2 y así sucesivamente.
Pattern	Indica que tipo de archivos van a aparecer en el objeto FileListBox (por defecto todos - *.*).
ReadOnly	Si esta propiedad está a True, en el objeto FileListBox se mostrarán los “archivos de sólo lectura”.
System	Si esta propiedad está a True, en el objeto FileListBox se mostrarán los “archivos de sistema”.
TabIndex	Su valor nos indicará en que orden se irá accediendo a los distintos controles activos (con TAB o Mayúsculas+TAB).

Visual Basic

Controles – Controles del Sistema de Archivos (Cuadro de Lista de Unidades (DriveListBox), Cuadro de Lista de Directorios (DirListBox) y Cuadro de Lista de Ficheros (FileListBox)).

Nombre Propiedad	Descripción
TabStop	Si está a <i>True</i> , indica que este control está activo para utilizarlo con TabIndex.
Tag	Esta propiedad es muy útil para que otros programadores entiendan como funciona tu programa, tanto si dejas el código para que alguien lo aprenda como si abandonas un proyecto y otras personas tienen que encargarse de él. La información que aparece en esta propiedad es meramente informativa, no afecta al comportamiento del control y no será vista por el usuario. Otra solución, quizás mejor, es comentar el código. Pero no podemos comentar un control.
Top	Determina la distancia que hay entre la esquina superior izquierda del cuadro de lista y el límite superior del contenedor de la misma (formulario).
Visible	Si podemos ver un control cuando ejecutamos un programa, es que su propiedad <i>Visible</i> está a <i>True</i> .
Width	Determina la anchura del cuadro de lista (en twips).

Visual Basic

Controles – Controles del Sistema de Archivos (Cuadro de Lista de Unidades (DriveListBox), Cuadro de Lista de Directorios (DirListBox) y Cuadro de Lista de Ficheros (FileListBox)).

Nombre Evento	Descripción
Change	Se produce después de que se haya modificado la posición en DiveListBox y en DirListBox.
Click	Este evento tiene lugar cuando hacemos click sobre el cuadro de lista en FileListBox y en DirListBox.
DbIcClick	Este evento tiene lugar cuando hacemos doble-click sobre el cuadro de lista en FileListBox.
DragDrop	Este evento tiene lugar cuando se completa una operación de arrastrar y colocar, es decir, cuando se suelta el botón del ratón tras arrastrar un objeto hasta un control.
DragOver	Ese evento tiene lugar mientras se está llevando a cabo una operación de arrastrar y colocar. Podemos utilizar este evento para cambiar el puntero del ratón cuando éste se encuentre dentro de los límites de un determinado control (para indicar que el control no lo acepta, por ejemplo).
GotFocus	Cuando el cuadro de lista recibe el enfoque se lanza este evento.
LostFocus	Si <i>GotFocus</i> se lanzaba cuando el botón de comando recibía el enfoque, <i>LostFocus</i> se lanza cuando lo pierde. El <i>enfoque</i> es la capacidad de recibir entradas del usuario a través del ratón o el teclado. Cuando un objeto tiene el enfoque, puede recibir entradas del usuario. Por ejemplo, cuando el cuadro de texto tiene el enfoque, el usuario puede escribir texto en él.

Visual Basic

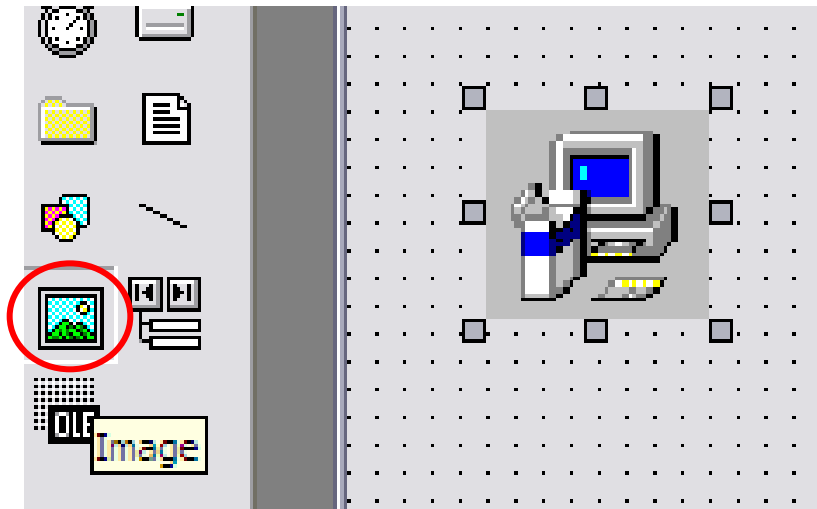
Controles – Controles del Sistema de Archivos (Cuadro de Lista de Unidades (DriveListBox), Cuadro de Lista de Directorios (DirListBox) y Cuadro de Lista de Ficheros (FileListBox)).

Nombre Evento	Descripción
KeyDown	Cuando pulsas una tecla, se lanza este evento.
KeyPress	Cuando pulsas una tecla y la sueltas, se lanza este evento.
KeyUp	Cuando sueltas una tecla, se lanza este evento.
MouseDown	Este evento se lanza cuando pulsas un botón del ratón en DirListBox y FileListBox.
MouseMove	Este evento se lanza cuando movemos el ratón en DirListBox y FileListBox.
MouseUp	Este evento se lanza cuando soltamos el botón del ratón en DirListBox y FileListBox.

Visual Basic

Controles – Imagen (Image).

- El control **Image** se utiliza para presentar gráficos. Los controles **Image** pueden presentar gráficos en los siguientes formatos: mapa de bits (.bmp o .dib), icono (.ico), metarchivo (.wmf), metarchivo mejorado (.emf) o archivos JPEG o GIF. Nos permite crear animaciones sencillas.
- El control **Image** utiliza menos recursos del sistema y se dibuja con más rapidez que el control **PictureBox**, pero sólo acepta algunos de los métodos, las propiedades y los eventos que acepta el control **PictureBox**.



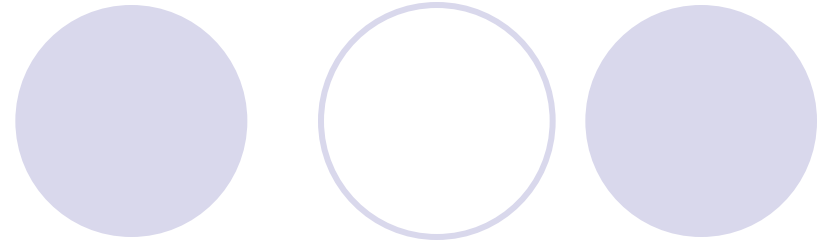
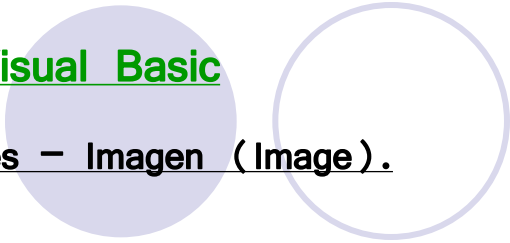
Visual Basic

Controles – Imagen (Image).

- Ambos controles aceptan los mismos formatos de imagen. Sin embargo, en los controles **Image** podemos ajustar las imágenes al tamaño del control para que quepa en él. No podemos hacer lo mismo con los controles **PictureBox**.

Visual Basic

Controles – Imagen (Image).

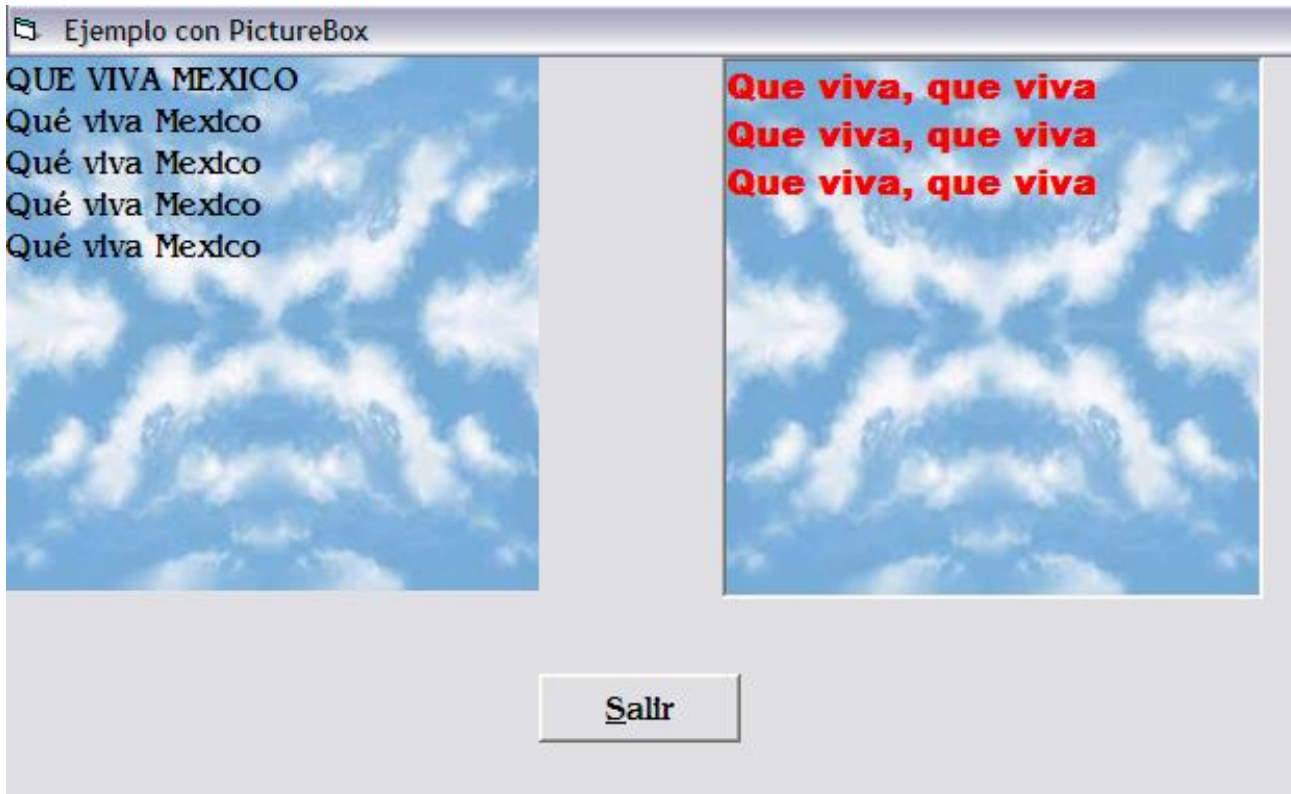


- **Propiedad:**
 - La propiedad **Stretch** determina si la imagen se va a reducir al cambiar el tamaño del control **Image** en tiempo de diseño. Si se establece a True, se reducirá la imagen cargada en el control **Image** mediante la propiedad **Picture**.

Visual Basic

Ejemplo 1.

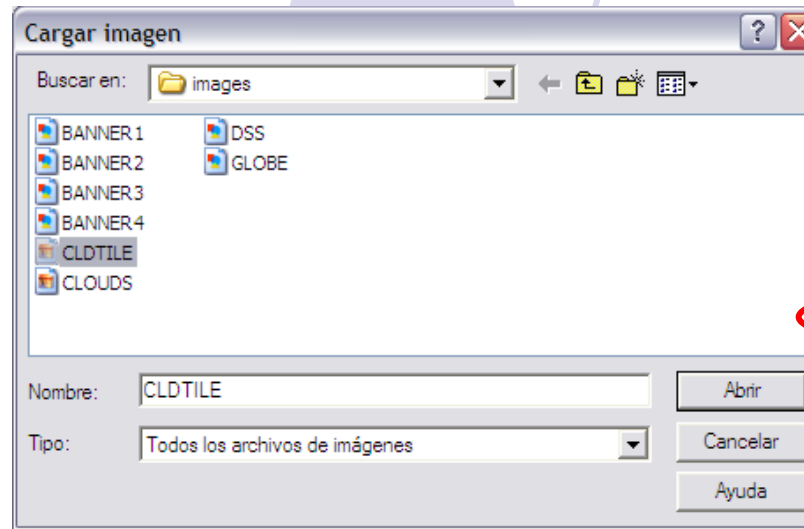
- Utilizando un objeto **PictureBox** vamos a situar la misma imagen en el formulario y en el **PictureBox**. Al principio se cargará la frase en el formulario “QUE VIVA MEXICO”. Después, dependiendo de si pulsamos en el PictureBox (que aparecerá: “**Que viva, que viva**” en rojo) o en el formulario de nuevo (que aparecerá: “Qué viva Mexico”) aparecerá un mensaje u otro. Para salir, botón **Salir**.



Visual Basic

C:\Documents and Settings\user\Mis documentos\Mis imágenes\Galería multimedia de Microsoft

Ejemplo 1.



LINKTOPIC	Form1
MaxButton	True
MDIChild	False
MinButton	True
MouseIcon	(Ninguno)
MousePointer	0 - Default
Moveable	True
NegotiateMenus	True
OLEDropMode	0 - None
Palette	(Ninguno)
PaletteMode	0 - HalfTone
Picture	(Mapa de bits)
RightToLeft	False
ScaleHeight	5355

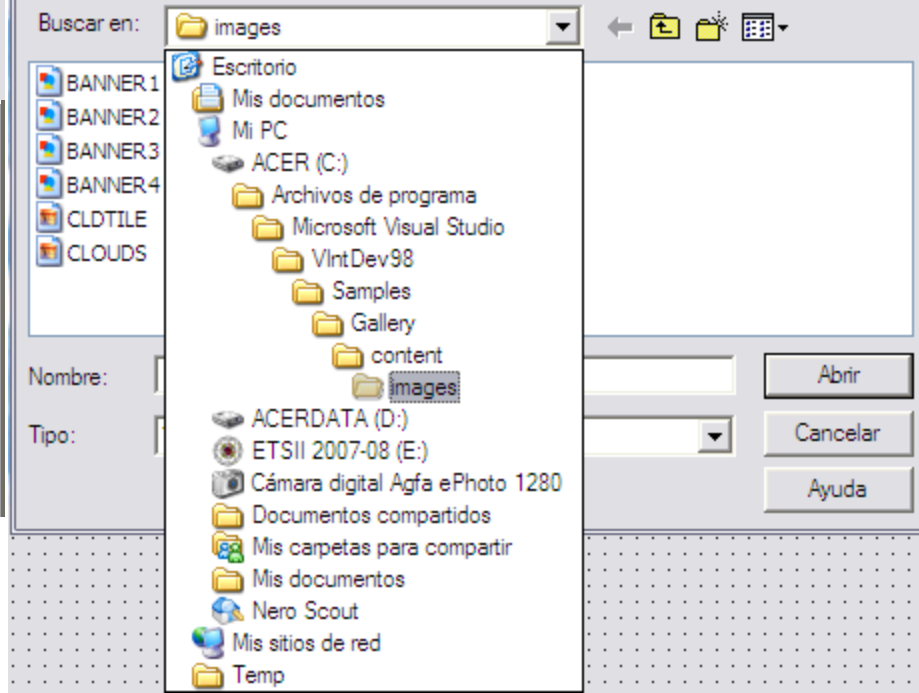
Picture
Devuelve o establece el control.

Formulario Form

Alfabética | Por categorías

(Nombre)	Formulario
Appearance	1 - 3D
AutoRedraw	True
BackColor	<input type="checkbox"/> &H8000000F&
BorderStyle	2 - Sizable
Caption	Ejemplo con PictureBox
ClipControls	True

Cargar imagen



Visual Basic

Ejemplo 1.

Propiedades - Dibujo

Dibujo PictureBox

Alfabética

Por categorías

MousePointer	0 - Default
Negotiate	False
OLEDragMode	0 - Manual
OLEDropMode	0 - None
Picture	(Mapa de bits)
RightToLeft	False
ScaleHeight	3840

Dibujo PictureBox

Alfabética

Por categorías

(Nombre)	Dibujo
Align	0 - None
Appearance	1 - 3D
AutoRedraw	True
AutoSize	True
BackColor	<input type="checkbox"/> &H80000000F&
BorderStyle	1 - Fixed Single
CausesValidation	True
ClipControls	True
DataField	
DataFormat	
DataMember	
DataSource	
DragIcon	(Ninguno)
DragMode	0 - Manual
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid
DrawWidth	1
Enabled	True
FillColor	<input checked="" type="checkbox"/> &H00000000&
FillStyle	1 - Transparent
Font	Arial Black
FontTransparent	True
ForeColor	<input checked="" type="checkbox"/> &H000000FF&
HasDC	True

Visual Basic

Ejemplo 1.

Option Explicit

```
Private Sub Form_Load()  
    Print "QUE VIVA MEXICO"  
    ' Nada más que empezar, se carga(LOAD) en el  
    ' Formulario: "QUE VIVA MEXICO".  
End Sub
```

```
Private Sub Form_Click()  
    Print "Qué viva Mexico"  
    ' Si hacemos Click con el ratón sobre el  
    ' Formulario, se escribirá: "Qué viva Mexico".  
End Sub
```

```
Private Sub Dibujo_Click()  
    Dibujo.Print "Que viva, que viva"  
    ' Si hacemos click con el ratón sobre el  
    ' PictureBox, se escribira: "Que viva, que viva"  
    ' en rojo.  
End Sub
```

```
Private Sub Salir_Click()  
    End  
    ' Para Salir.  
End Sub
```


Visual Basic

Ejemplo 2.

- Vamos a comprobar el evento **CHANGE**. Cada vez que “***cambiamos algo***” en el cuadro de texto (Name = *Texto*) se lanzará un evento que provocará que escribamos el literal del cuadro de texto en el formulario.

The screenshot shows a Visual Basic form titled "Form1". On the left side, there is a list of text box values: C, Ca, Cas, Casl, Caslt, Casita, Caslt, Casl, Cas, and Ca. In the center, the text "Introduce un carácter:" is displayed. To the right of this text is a text box containing the value "Ca". Below the text box is a button labeled "Salir".

Visual Basic

Ejemplo 2.

Formulario Form

Alfabetica | Por categorías

(Nombre)	Formulario
Appearance	1 - 3D
AutoRedraw	True
BackColor	<input type="checkbox"/> &H80000000F&
BorderStyle	2 - Sizable
Caption	Form1
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid
DrawWidth	1
Enabled	True
FillColor	<input checked="" type="checkbox"/> &H000000000&
FillStyle	1 - Transparent
Font	Bangle
FontTransparent	True
ForeColor	<input checked="" type="checkbox"/> &H008000000&
HasDC	True

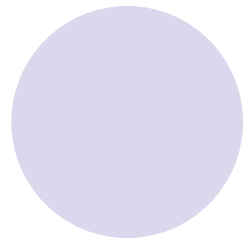
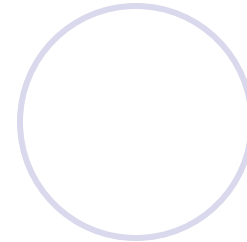
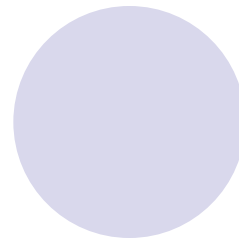
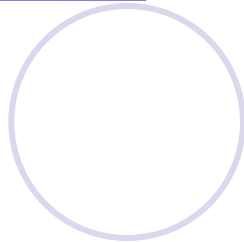
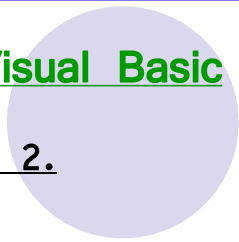
Etiqueta Label

Alfabetica | Por categorías

(Nombre)	Etiqueta
Alignment	0 - Left Justify
Appearance	1 - 3D
AutoSize	True
BackColor	<input type="checkbox"/> &H80000000F&
BackStyle	1 - Opaque
BorderStyle	0 - None
Caption	Introduce un carácter:
DataField	

Visual Basic

Ejemplo 2.



Option Explicit

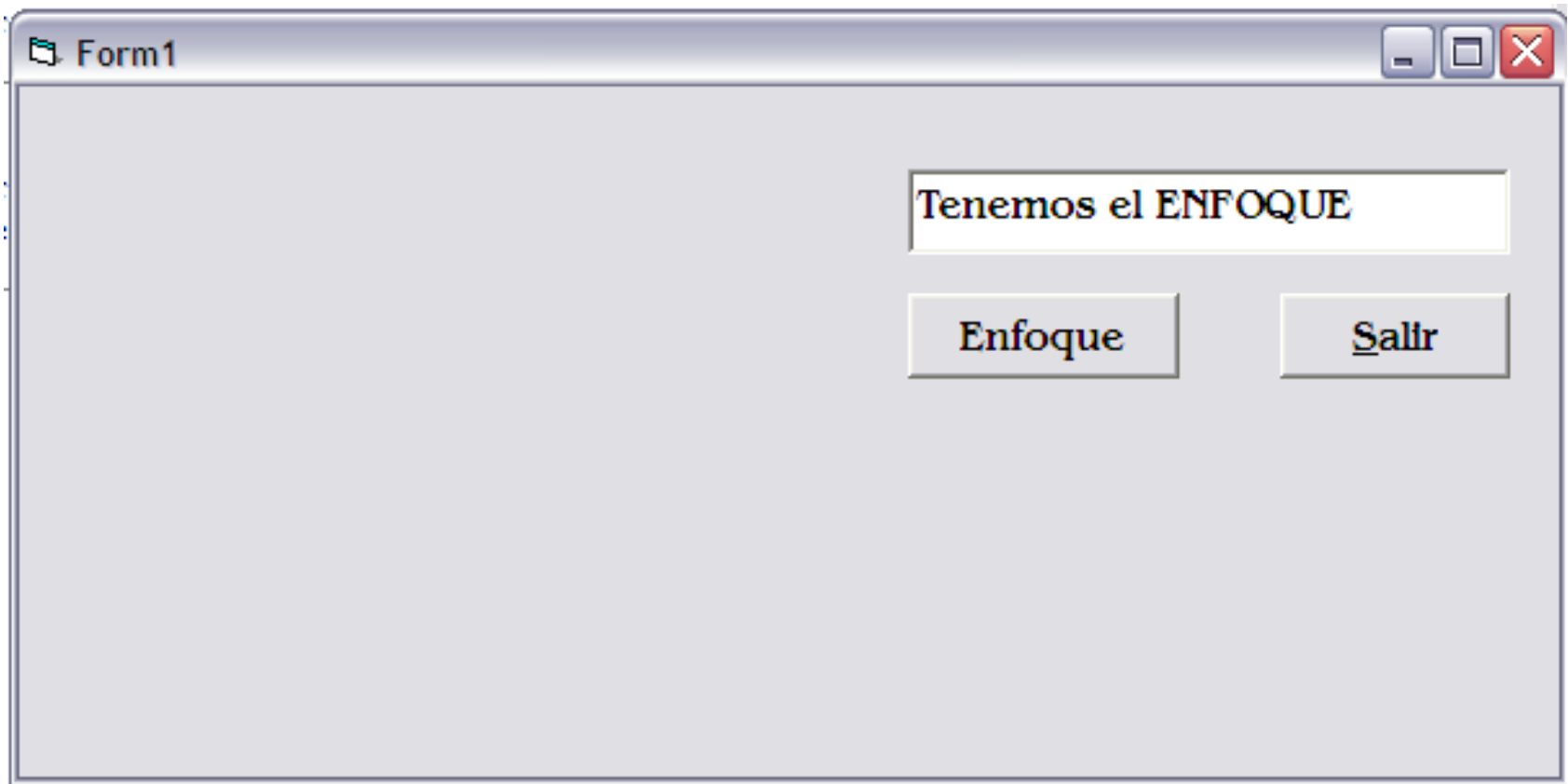
```
Private Sub Texto_Change()  
    Formulario.Print Texto.Text  
    ' Utilizamos el evento CHANGE para que cada vez  
    ' que escribamos o borremos en el cuadro de  
    ' texto "Texto" lo que haya, se escriba en el  
    ' formulario.  
End Sub
```

```
Private Sub Salir_Click()  
    End  
End Sub
```

Visual Basic

Ejemplo 3.

- Vamos a comprobar el evento **GotFocus** y **LostFocus**. Cada vez que pulsemos el botón *Enfoque*, se escribirá en el formulario, en rojo, “El Botón tiene el ENFOQUE”. Y cada vez que situemos el cursor en la caja de texto se visualizará el mensaje “Tenemos el ENFOQUE”.



Visual BasicEjemplo 3.**Formulario** Form

Alfabética Por categorías

(Nombre)	Formulario
Appearance	1 - 3D
AutoRedraw	True
BackColor	<input type="checkbox"/> &H80000000F&
BorderStyle	2 - Sizable
Caption	Form1
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid
DrawWidth	1
Enabled	True
FillColor	<input checked="" type="checkbox"/> &H00000000&
FillStyle	1 - Transparent
Font	Bangle
FontTransparent	True
ForeColor	<input checked="" type="checkbox"/> &H000000FF&
HasDC	True

Texto TextBox

Alfabética Por categorías

(Nombre)	Texto
Alignment	0 - Left Justify
Appearance	1 - 3D

Visual Basic

Ejemplo 3.

Option Explicit

```
Private Sub Texto_GotFocus()  
    Texto.Text = "Tenemos el ENFOQUE"  
    ' Cada vez que nos situamos con el cursor sobre  
    ' la caja de texto, TENEMOS EL FOCO.  
    ' Por lo tanto escribimos "Tenemos el ENFOQUE".  
    Formulario.Cls  
End Sub
```

```
Private Sub Texto_LostFocus()  
    Formulario.Print "El Botón tiene el ENFOQUE"  
    ' Cada vez que pulsamos el botón "Enfoque" perdemos  
    ' el ENFOQUE.  
    ' Por lo tanto escribimos "El Botón tiene el ENFOQUE".  
    Texto.Text = ""  
End Sub
```

```
Private Sub Salir_Click()  
    End  
End Sub
```

Visual Basic

Ejemplo 4.

- El programa nos da la posibilidad de seleccionar varias Opciones Hardware del marco. Una vez seleccionadas, pulsamos Aceptar y en el cuadro texto deben aparecer comentarios indicando dicha selección.

The image shows a Windows-style dialog box titled "Elige la configuración de tu ordenador y pulsa Aceptar:". It contains a group box labeled "OPciones Hardware" with four items: "CD-Rom" (unchecked), "Adaptador Gráfico" (checked), "Salidas Midi" (unchecked), and "Unidad M.O." (checked). At the bottom are "Aceptar" and "Salir" buttons. To the right, a text area displays the resulting configuration: "La Configuración de tu ordenador es:", "NO tienes CD-ROM.", "Tienes Adaptador de Vídeo.", "NO tienes MIDI", and "Tienes unidad M.O.".

Elige la configuración de tu ordenador y pulsa Aceptar:

OPciones Hardware

☐ CD-Rom

☒ Adaptador Gráfico

☐ Salidas Midi

☒ Unidad M.O.

Aceptar Salir

La Configuración de tu ordenador es:

NO tienes CD-ROM.
Tienes Adaptador de Vídeo.
NO tienes MIDI
Tienes unidad M.O.

Visual Basic

4.

Visual Basic

4.

Visual Basic

4.

Visual Basic

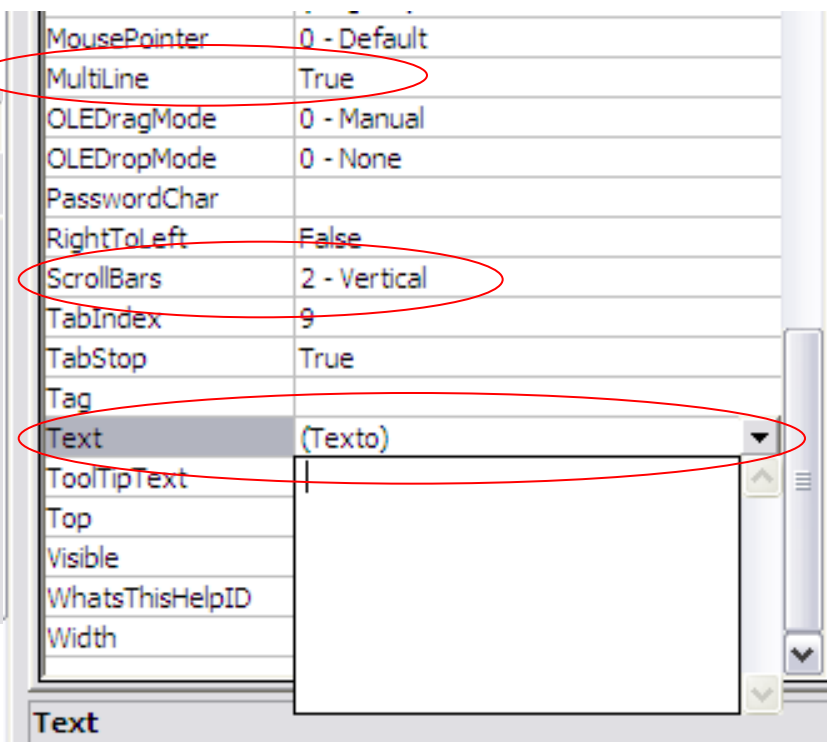
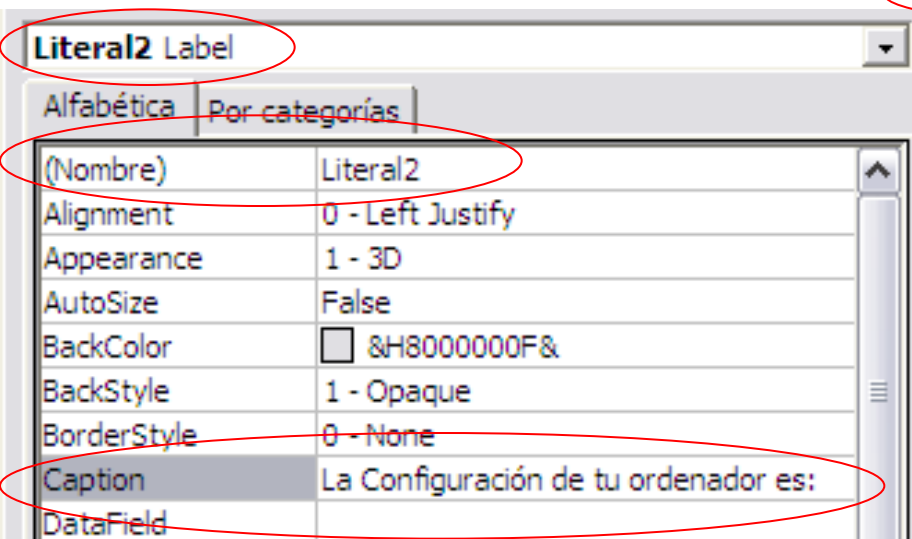
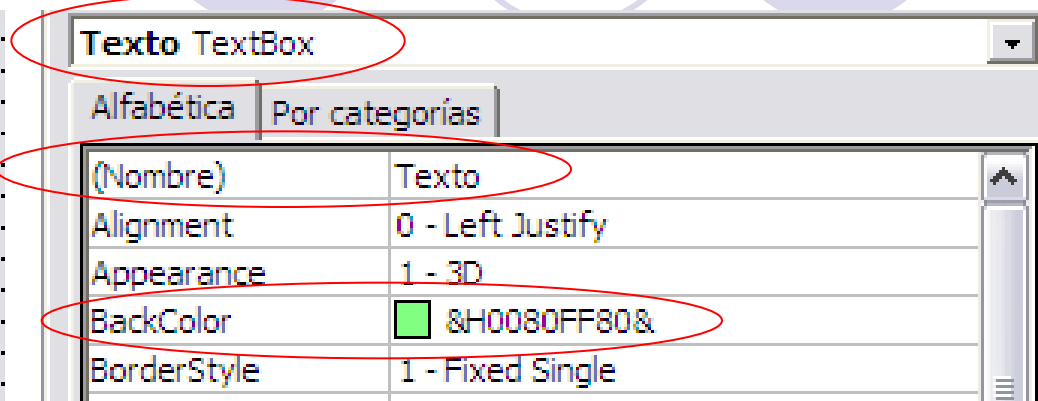
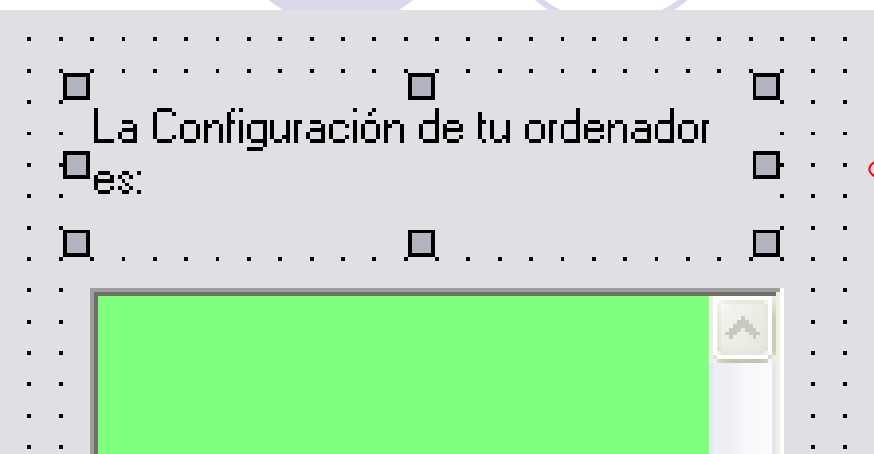
4.

Visual Basic

4.

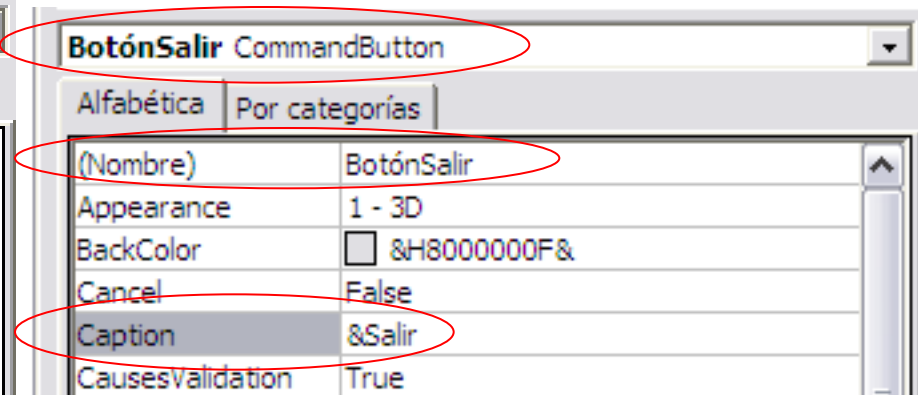
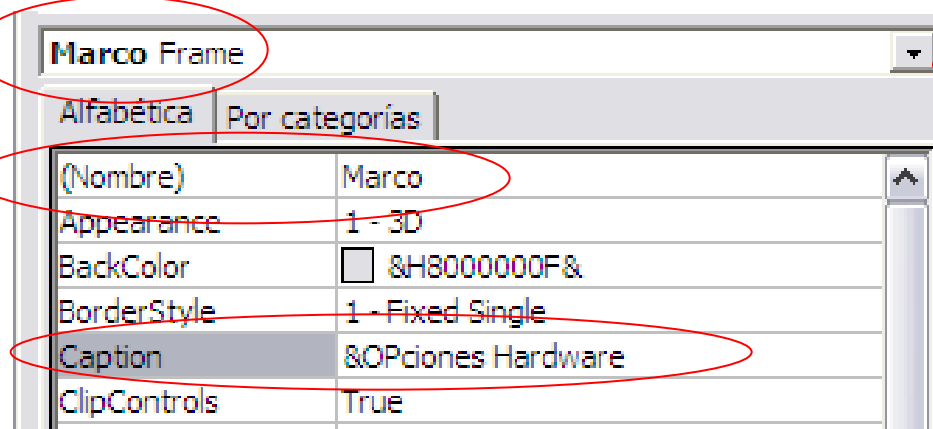
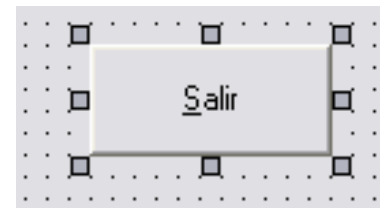
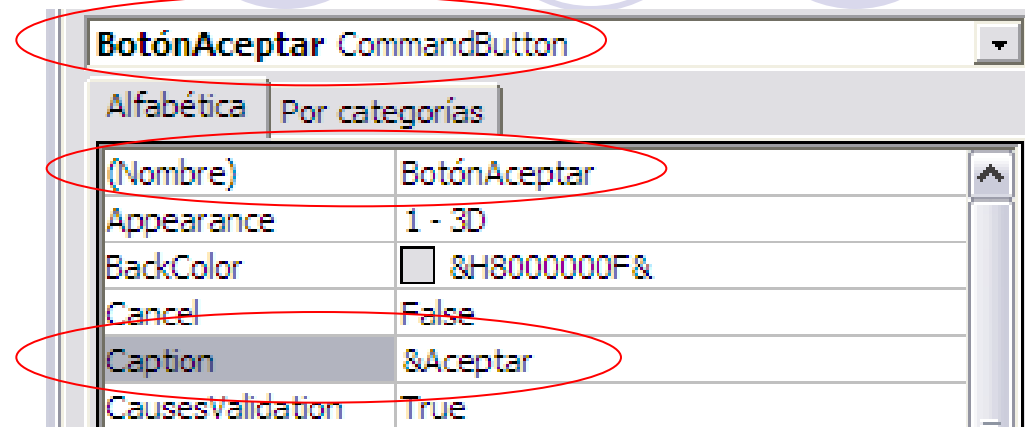
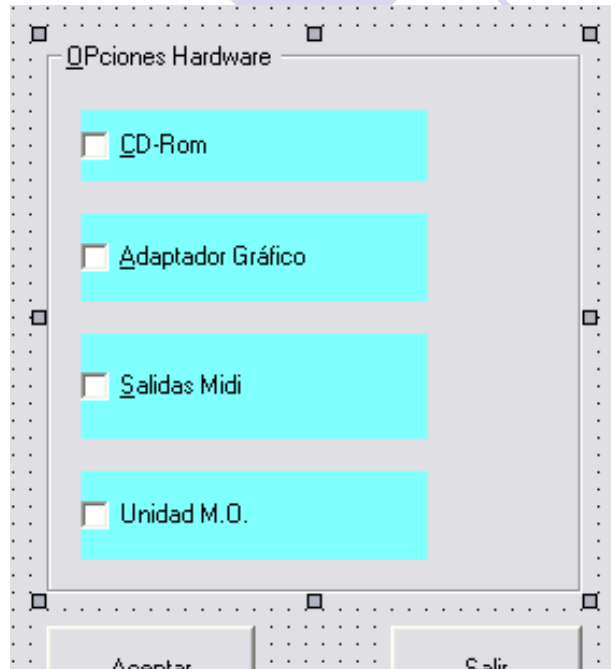
Visual Basic

Ejemplo 4.



Visual Basic

Ejemplo 4.



Visual Basic

Ejemplo 4.



Adaptador Gráfico

OpciónAdaptador CheckBox

Alfabética | Por categorías

(Nombre) OpciónAdaptador

Alignment 0 - Left Justify

Appearance 1 - 3D

BackColor  &H00FFFF80&

Caption &Adaptador Gráfico

CausesValidation True



CD-Rom

OpciónCD CheckBox

Alfabética | Por categorías

(Nombre) OpciónCD

Alignment 0 - Left Justify

Appearance 1 - 3D

BackColor  &H00FFFF80&

Caption &CD-Rom

CausesValidation True

Visual Basic

Ejemplo 4.

☐ Salidas Midi

☐ Unidad M.O.

OpciónMO CheckBox

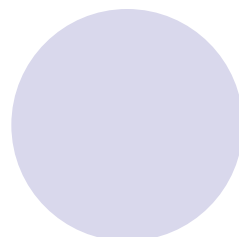
Alfabética Por categorías

(Nombre)	OpciónMO
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H00FFFF80&
Caption	Unidad M.O.
CausesValidation	True

OpciónMidi CheckBox

Alfabética Por categorías

(Nombre)	OpciónMidi
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H00FFFF80&
Caption	&Salidas Midi
CausesValidation	True



```
Private Sub BotónAceptar_Click()  
    Dim Cadena As String  
    Dim Retorno As String  
  
    Retorno = Chr$(13) & Chr$(10)  
        ' Retorno de Carro  
    Cadena = ""  
        ' Inicializamos la cadena de salida a espacios.  
    If OpciónCD.Value = 1 Then  
        ' Si el ChekBox "OpciónCD" tiene la propiedad Value = 1  
        ' quiere decir que está marcado.  
        Cadena = "Tienes CD-Rom." & Retorno  
            ' Escribimos: "Tienes CD-Rom" y saltamos a la línea siguiente.  
    Else  
        Cadena = "NO tienes CD-ROM." & Retorno  
            ' Escribimos "NO tienes CD-Rom" y saltamos a la línea siguiente.  
    End If  
    If OpciónAdaptador.Value = 1 Then  
        Cadena = Cadena & "Tienes Adaptador de Vídeo." & Retorno  
    Else  
        Cadena = Cadena & "NO tienes Adaptador de Vídeo." & Retorno  
    End If  
    If OpciónMidi.Value = 1 Then  
        Cadena = Cadena & "Tienes MIDI." & Retorno  
    Else  
        Cadena = Cadena & "NO tienes MIDI" & Retorno  
    End If  
    If OpciónMO.Value = 1 Then  
        Cadena = Cadena & "Tienes unidad M.O." & Retorno  
    Else  
        Cadena = Cadena & "NO tienes unidad M.O." & Retorno  
    End If
```



```
Texto.Text = Cadena
    ' Pasamos lo que hay en "Cadena" al Cuadro de Texto "Texto".
Literal2.Enabled = True
    ' Activamos la etiqueta "Literal2" que estaba Desactivada.
Texto.Enabled = True
    ' Activamos el texto "Texto" que estaba Desactivado.
End Sub
```

```
Private Sub OpciónCD_Click()
    Texto.Enabled = False
    ' Desactivamos el Cuadro de Texto "Texto" para que mientras
    ' se manejan los botones, no se pueda escribir en él.
    Texto.Text = ""
    ' Inicializamos el texto de "Texto" a espacios.
    Literal2.Enabled = False
    ' Desactivamos la etiqueta "Literal2" para que nadie la pueda
    ' modificar.
End Sub
```

```
Private Sub OpciónAdaptador_Click()
    Texto.Enabled = False
    ' Desactivamos el Cuadro de Texto "Texto" para que mientras
    ' se manejan los botones, no se pueda escribir en él.
    Texto.Text = ""
    ' Inicializamos el texto de "Texto" a espacios.
    Literal2.Enabled = False
    ' Desactivamos la etiqueta "Literal2" para que nadie la pueda
    ' modificar.
End Sub
```

```
Private Sub OpciónMidi_Click()  
    Texto.Enabled = False  
        ' Desactivamos el Cuadro de Texto "Texto" para que mientras  
        ' se manejan los botones, no se pueda escribir en él.  
    Texto.Text = ""  
        ' Inicializamos el texto de "Texto" a espacios.  
    Literal2.Enabled = False  
        ' Desactivamos la etiqueta "Literal2" para que nadie la pueda  
        ' modificar.  
End Sub
```

```
Private Sub OpciónMO_Click()  
    Texto.Enabled = False  
        ' Desactivamos el Cuadro de Texto "Texto" para que mientras  
        ' se manejan los botones, no se pueda escribir en él.  
    Texto.Text = ""  
        ' Inicializamos el texto de "Texto" a espacios.  
    Literal2.Enabled = False  
        ' Desactivamos la etiqueta "Literal2" para que nadie la pueda  
        ' modificar.  
End Sub
```

```
Private Sub BotónSalir_Click()  
    End  
End Sub
```

Visual Basic

Ejemplo 44.

- El programa nos da la posibilidad de cambiar el color de un texto utilizando las opciones de los Cuadros de Texto (R, V y A), las opciones de los Botones y las Barras de Desplazamiento.

Form1

Color de Texto

Rojo:

Verde:

Azul:

Ro+ **Ve+** **Az+**

209 **129** **70**

Ro- **Ve-** **Az-**

Salir

Visual Basic

Ejemplo 44.

Color de Texto

Propiedades - Form1

Form1 Form

Alfabética | Por categorías

Palette	(Ninguno)
PaletteMode	0 - Halftone
Picture	(Ninguno)
RightToLeft	False
ScaleHeight	11115
ScaleLeft	0
ScaleMode	1 - Twip
ScaleTop	0
ScaleWidth	15240
ShowInTaskbar	True
StartPosition	3 - Windows Defau
Tag	
Top	0
Visible	True
WhatsThisButton	False
WhatsThisHelp	False
Width	15360
WindowState	2 - Maximized

Propiedades - Text1

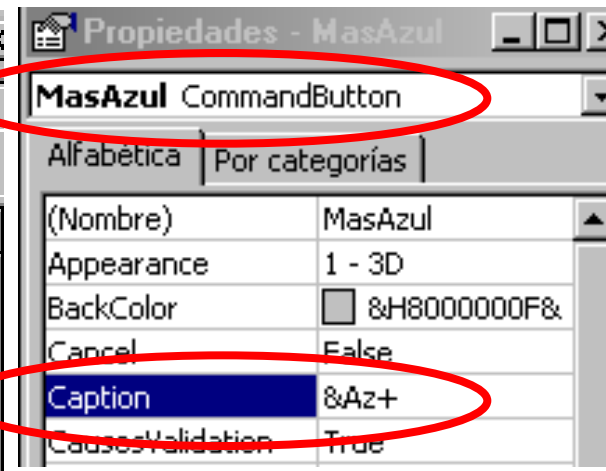
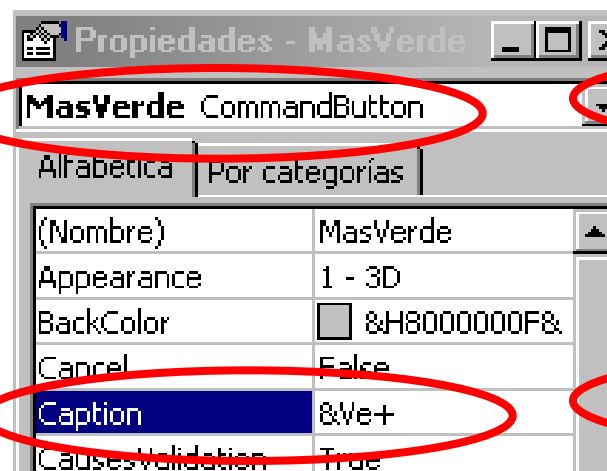
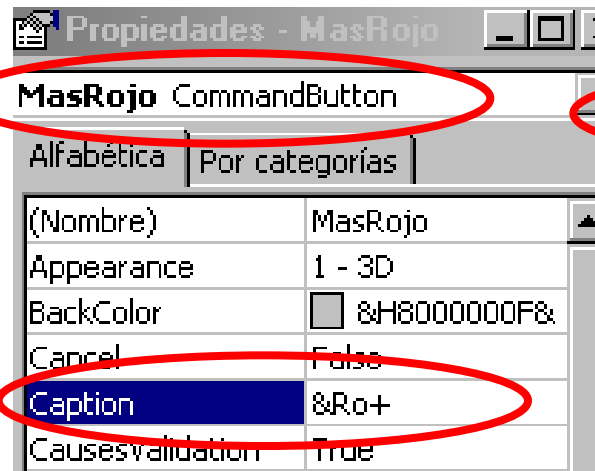
Text1 TextBox

Alfabética | Por categorías

MaxLength	0
MouseIcon	(Ninguno)
MousePointer	0 - Default
MultiLine	False
OLEDragMode	0 - Manual
OLEDropMode	0 - None
PasswordChar	
RightToLeft	False
ScrollBars	0 - None
TabIndex	0
TabStop	True
Tag	
Text	Color de Texto
ToolTipText	

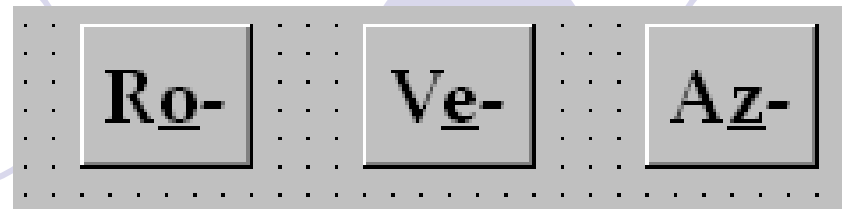
Visual Basic

Ejemplo 44.



Visual Basic

Ejemplo 44.



Propiedades - MenosRojo

MenosRojo CommandButton

Alfabética	Por categorías
(Nombre)	MenosRojo
Appearance	1 - 3D
BackColor	&H80000000F&
Cancel	False
Caption	R&o-
CausesValidation	True

Propiedades - MenosV...

MenosVerde CommandButton

Alfabética	Por categorías
(Nombre)	MenosVerde
Appearance	1 - 3D
BackColor	&H80000000F&
Cancel	False
Caption	V&e-
CausesValidation	True

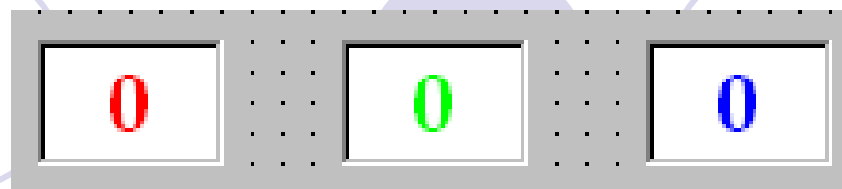
Propiedades - MenosAzul

MenosAzul CommandButton

Alfabética	Por categorías
(Nombre)	MenosAzul
Appearance	1 - 3D
BackColor	&H80000000F&
Cancel	False
Caption	A&z-
Causesvalidation	True

Visual Basic

Ejemplo 44.



Propiedades - TextoRojo	
TextoRojo TextBox	
Alfabética	Por categorías
MaxLength	0
MouseIcon	(Ninguno)
MousePointer	0 - Default
MultiLine	False
OLEDragMode	0 - Manual
OLEDropMode	0 - None
PasswordChar	
RightToLeft	False
ScrollBars	0 - None
TabIndex	7
TabStop	True
Tag	
Text	0
ToolTipText	

Propiedades - TextoVerde	
TextoVerde TextBox	
Alfabética	Por categorías
MaxLength	0
MouseIcon	(Ninguno)
MousePointer	0 - Default
MultiLine	False
OLEDragMode	0 - Manual
OLEDropMode	0 - None
PasswordChar	
RightToLeft	False
ScrollBars	0 - None
TabIndex	8
TabStop	True
Tag	
Text	0
ToolTipText	

Propiedades - TextoAzul	
TextoAzul TextBox	
Alfabética	Por categorías
MaxLength	0
MouseIcon	(Ninguno)
MousePointer	0 - Default
MultiLine	False
OLEDragMode	0 - Manual
OLEDropMode	0 - None
PasswordChar	
RightToLeft	False
ScrollBars	0 - None
TabIndex	9
TabStop	True
Tag	
Text	0
ToolTipText	

Visual Basic



Ejemplo 44.



Rojo:



Verde:


Azul:

Salir

Propiedades - Etiqueta...	
EtiquetaRojo Label	
Alfabética	Por categorías
(Nombre)	EtiquetaRojo
Alignment	0 - Left Justify
Appearance	1 - 3D
AutoSize	True
BackColor	 &H8000000F&
BackStyle	1 - Opaque
BorderStyle	0 - None
Caption	Rojo:
DataField	
DataFormat	
DataMember	
DataSource	
DragIcon	(Ninguno)
DragMode	0 - Manual
Enabled	True
Font	Palatino Linotype
ForeColor	 &H000000FF&

Propiedades - Etiqueta...	
EtiquetaVerde Label	
Alfabética	Por categorías
(Nombre)	EtiquetaVerde
Alignment	0 - Left Justify
Appearance	1 - 3D
AutoSize	True
BackColor	 &H8000000F&
BackStyle	1 - Opaque
BorderStyle	0 - None
Caption	Verde:
DataField	
DataFormat	
DataMember	
DataSource	
DragIcon	(Ninguno)
DragMode	0 - Manual
Enabled	True
Font	Palatino Linotype
ForeColor	 &H0000C000&

Propiedades - Etiqueta...	
EtiquetaAzul Label	
Alfabética	Por categorías
(Nombre)	EtiquetaAzul
Alignment	0 - Left Justify
Appearance	1 - 3D
AutoSize	True
BackColor	 &H8000000F&
BackStyle	1 - Opaque
BorderStyle	0 - None
Caption	Azul:
DataField	
DataFormat	
DataMember	
DataSource	
DragIcon	(Ninguno)
DragMode	0 - Manual
Enabled	True
Font	Palatino Linotype
ForeColor	 &H00FF0000&

Propiedades - BotonSalir	
BotonSalir CommandButton	
Alfabética	Por categorías
(Nombre)	BotonSalir
Appearance	1 - 3D
BackColor	 &H8000000F&
Cancel	False
Caption	Salir
CausesValidation	True

Visual Basic

Ejemplo 44.

Rojo:

Verde:

Azul:

Propiedades - Horizont...

HorizontalRojo HScrollBar

Alfabética | Por categorías

HelpContextID	0
Index	
LargeChange	10
Left	1440
Max	255
Min	0
MouseIcon	(Ninguno)
MousePointer	0 - Default
RightToLeft	False
SmallChange	1
TabIndex	15

Propiedades - Horizont...

HorizontalVerde HScrollBar

Alfabética | Por categorías

HelpContextID	0
Index	
LargeChange	10
Left	1440
Max	255
Min	0
MouseIcon	(Ninguno)
MousePointer	0 - Default
RightToLeft	False
SmallChange	1
TabIndex	11

Propiedades - Horizont...

HorizontalAzul HScrollBar

Alfabética | Por categorías

HelpContextID	0
Index	
LargeChange	10
Left	1440
Max	255
Min	0
MouseIcon	(Ninguno)
MousePointer	0 - Default
RightToLeft	False
SmallChange	1
TabIndex	16

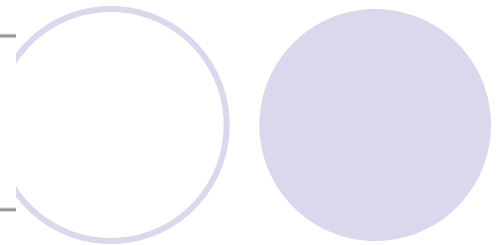
```
Option Explicit
Dim Rojo As Integer, Verde As Integer, Azul As Integer
```

```
Private Sub BotonSalir_Click()
    End
End Sub
```

```
Private Sub MasRojo_Click()
    If (Rojo < 255) Then
        Rojo = Rojo + 1
        Text1.ForeColor = RGB(Rojo, Verde, Azul)
        TextoRojo.Text = CStr(Rojo)
        HorizontalRojo.Value = Rojo
    End If
End Sub
```

```
Private Sub MenosRojo_Click()
    If (Rojo > 0) Then
        Rojo = Rojo - 1
        Text1.ForeColor = RGB(Rojo, Verde, Azul)
        TextoRojo.Text = CStr(Rojo)
        HorizontalRojo.Value = Rojo
    End If
End Sub
```

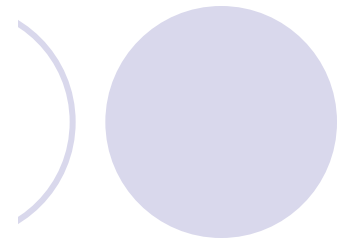
```
Private Sub MasVerde_Click()
    If (Verde < 255) Then
        Verde = Verde + 1
        Text1.ForeColor = RGB(Rojo, Verde, Azul)
        TextoVerde.Text = CStr(Verde)
        HorizontalVerde.Value = Verde
    End If
End Sub
```



```
Private Sub MenosVerde_Click()  
    If (Verde > 0) Then  
        Verde = Verde - 1  
        Text1.ForeColor = RGB(Rojo, Verde, Azul)  
        TextoVerde.Text = CStr(Verde)  
        HorizontalVerde.Value = Verde  
    End If  
End Sub
```

```
Private Sub MasAzul_Click()  
    If (Azul < 255) Then  
        Azul = Azul + 1  
        Text1.ForeColor = RGB(Rojo, Verde, Azul)  
        TextoAzul.Text = CStr(Azul)  
        HorizontalAzul.Value = Azul  
    End If  
End Sub
```

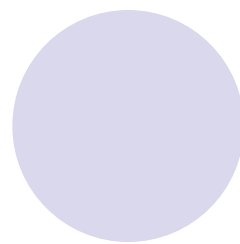
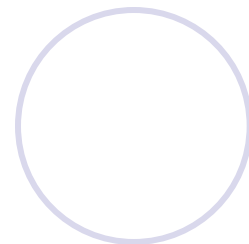
```
Private Sub MenosAzul_Click()  
    If (Azul > 0) Then  
        Azul = Azul - 1  
        Text1.ForeColor = RGB(Rojo, Verde, Azul)  
        TextoAzul.Text = CStr(Azul)  
        HorizontalAzul.Value = Azul  
    End If  
End Sub
```




```
Private Sub TextoRojo_KeyPress(KeyAscii As Integer)
If KeyAscii <> 13 Then
    Exit Sub
End If
Rojo = CInt(TextoRojo.Text)
If Rojo < 256 And Rojo >= 0 Then
    Text1.ForeColor = RGB(Rojo, Verde, Azul)
    HorizontalRojo.Value = Rojo
End If
End Sub
```

```
Private Sub TextoVerde_KeyPress(KeyAscii As Integer)
If KeyAscii <> 13 Then
    Exit Sub
End If
Verde = CInt(TextoVerde.Text)
If Verde < 256 And Verde >= 0 Then
    Text1.ForeColor = RGB(Rojo, Verde, Azul)
    HorizontalVerde.Value = Verde
End If
End Sub
```

```
Private Sub TextoAzul_KeyPress(KeyAscii As Integer)
If KeyAscii <> 13 Then
    Exit Sub
End If
Azul = CInt(TextoAzul.Text)
If Azul < 256 And Azul >= 0 Then
    Text1.ForeColor = RGB(Rojo, Verde, Azul)
    HorizontalAzul.Value = Azul
End If
End Sub
```



```
Private Sub HorizontalRojo_Change()  
    TextoRojo.Text = HorizontalRojo.Value  
    Rojo = CInt(TextoRojo.Text)  
    Text1.ForeColor = RGB(Rojo, Verde, Azul)  
End Sub
```

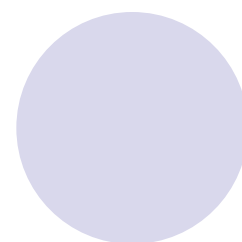
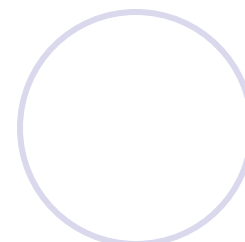
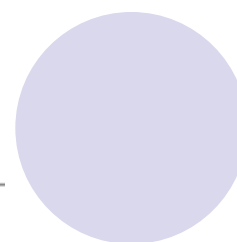
```
Private Sub HorizontalRojo_Scroll()  
    TextoRojo.Text = HorizontalRojo.Value  
    Rojo = CInt(TextoRojo.Text)  
    Text1.ForeColor = RGB(Rojo, Verde, Azul)  
End Sub
```

```
Private Sub HorizontalVerde_Change()  
    TextoVerde.Text = HorizontalVerde.Value  
    Verde = CInt(TextoVerde.Text)  
    Text1.ForeColor = RGB(Rojo, Verde, Azul)  
End Sub
```

```
Private Sub HorizontalVerde_Scroll()  
    TextoVerde.Text = HorizontalVerde.Value  
    Verde = CInt(TextoVerde.Text)  
    Text1.ForeColor = RGB(Rojo, Verde, Azul)  
End Sub
```

```
Private Sub HorizontalAzul_Change()  
    TextoAzul.Text = HorizontalAzul.Value  
    Azul = CInt(TextoAzul.Text)  
    Text1.ForeColor = RGB(Rojo, Verde, Azul)  
End Sub
```

```
Private Sub HorizontalAzul_Scroll()  
    TextoAzul.Text = HorizontalAzul.Value  
    Azul = CInt(TextoAzul.Text)  
    Text1.ForeColor = RGB(Rojo, Verde, Azul)  
End Sub
```



Visual Basic

Ejemplo 5.

- El programa debe buscar la composición de nuestro ordenador. Para ello podemos seleccionar sólo una de las opciones en cada uno de los marcos. Al final, escribiremos dicha composición.

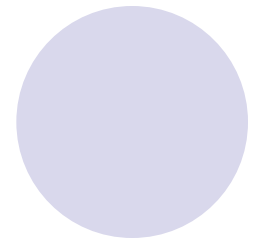
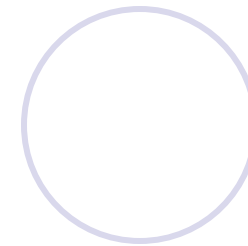
Form1

Procesador	Memoria	Discos
<input checked="" type="radio"/> Pentium 2 GHz	<input type="radio"/> 512 MB	<input type="radio"/> 80 GB
<input type="radio"/> Pentium 8 GHz	<input type="radio"/> 1024 MB	<input checked="" type="radio"/> 160 GB
<input type="radio"/> Pentium 16 GHz	<input checked="" type="radio"/> 2048 MB	<input type="radio"/> 320 GB

Equipo compuesto por un Procesador Pentium a 2 GHz con una memoria RAM de 2048 Megas, y con un Disco Duro de 160 GB.

Visual Basic

Ejemplo 5.



Propiedades - Form1

Form1 Form

Alfabética Por categorías

(Nombre)	Form1
Appearance	1 - 3D
AutoRedraw	True
BackColor	<input type="checkbox"/> &H8000000F&

WhatsThisHelp	False
Width	15360
WindowState	2 - Maximized

BotónSeleccionar CommandButton

Alfabética Por categorías

(Nombre)	BotónSeleccionar
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	&Seleccionar
CausesValidation	True



BotónSalir CommandButton

Alfabética Por categorías

(Nombre)	BotónSalir
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	S&alir
CausesValidation	True

Visual Basic

Ejemplo 5.

Procesador

☒ Pentium 2 GHz

☐ Pentium 8 GHz

☐ Pentium 16 GHz

P2 OptionButton

Alfabética | Por categorías

(Nombre)	P2
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Caption	Pentium 2 GHz
CausesValidation	True
UseMaskColor	False
Value	True
Visible	True

P8 OptionButton

Alfabética | Por categorías

(Nombre)	P8
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Caption	Pentium 8 GHz
CausesValidation	True
UseMaskColor	False
Value	False
Visible	True

Marco1 Frame

Alfabética | Por categorías

(Nombre)	Marco1
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
BorderStyle	1 - Fixed Single
Caption	Procesador
ClipControls	True

☐ Pentium 16 GHz

Propiedades - P16

P16 OptionButton

Alfabética | Por categorías

(Nombre)	P16
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Caption	Pentium 16 GHz
CausesValidation	True

UseMaskColor	False
Value	False
Visible	True

Visual Basic

Ejemplo 5.

Memoria

☐ 512 MB

☐ 1024 MB

☒ 2048 MB

P2 OptionButton

Alfabética | Por categorías

(Nombre)	P2
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Caption	Pentium 2 GHz
CausesValidation	True
UseMaskColor	False
Value	False
Visible	True

☐ 1024 MB

M1024 OptionButton

Alfabética | Por categorías

(Nombre)	M1024
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Caption	1024 MB
CausesValidation	True
UseMaskColor	False
Value	False
Visible	True

Marco2 Frame

Alfabética | Por categorías

(Nombre)	Marco2
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
BorderStyle	1 - Fixed Single
Caption	Memoria
ClipControls	True

☒ 2048 MB

M2048 OptionButton

Alfabética | Por categorías

(Nombre)	M2048
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Caption	2048 MB
CausesValidation	True

UseMaskColor	False
Value	True
Visible	True

Visual Basic

Ejemplo 5.

Discos

☐ 80 GB

☒ 160 GB

☐ 320 GB

☐ 80 GB

D80 OptionButton

Alfabética | Por categorías

(Nombre)	D80
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	&H8000000F&
Caption	80 GB
CausesValidation	True
UseMaskColor	False
Value	False
Visible	True

☒ 160 GB

D160 OptionButton

Alfabética | Por categorías

(Nombre)	D160
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	&H8000000F&
Caption	160 GB
CausesValidation	True
UseMaskColor	False
Value	True
Visible	True

Marco3 Frame

Alfabética | Por categorías

(Nombre)	Marco3
Appearance	1 - 3D
BackColor	&H8000000F&
BorderStyle	1 - Fixed Single
Caption	Discos
CloControls	True

☐ 320 GB

D320 OptionButton

Alfabética | Por categorías

(Nombre)	D320
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	&H8000000F&
Caption	320 GB
CausesValidation	True

UseMaskColor	False
Value	False
Visible	True

Visual Basic

5.

Visual Basic

5.

Visual Basic

5.

Visual Basic

5.

Visual Basic

5.


```
Private Sub BotónSalir_Click()  
    End  
End Sub
```

```
Private Sub BotónSeleccionar_Click()  
    Dim Procesador As String  
    Dim Memoria As Integer  
    Dim Megas As Integer  
  
    If P2.Value Then  
        ' Si P2.Value es igual a True, esta opción esta seleccionada.  
        Procesador = "Pentium a 2 GHz"  
    Else  
        If P8.Value Then  
            Procesador = "Pentium a 8 GHz"  
        Else  
            Procesador = "Pentium a 16 GHz"  
        End If  
    End If  
  
    If M512.Value Then  
        Memoria = 512  
        ' Como la propiedad Values es la propiedad por defecto  
        ' la podemos OMITIR.  
    Else  
        If M1024 Then  
            Memoria = 1024  
        Else  
            Memoria = 2048  
        End If  
    End If
```



```
If D80 Then
    Megas = 80
Else
    If D160 Then
        Megas = 160
    Else
        Megas = 320
    End If
End If

Texto.Text = "Equipo compuesto por un Procesador " & Procesador & _
            " con una memoria RAM de " & Memoria & _
            " Megas, y con un Disco Duro de " & Megas & " GB."

End Sub
```

Visual Basic

Ejemplo 6.

- Consiste en crear un interfaz donde en un lista de ficheros que hayamos introducido previamente podamos realizar las tareas de “Añadir”, “Borrar un elemento”, “Borrar toda la lista” y “Salir”.

Form1

Selecciona varios elementos de la Lista de Ficheros y pulsa: "Agregar > > >"

Lista de Ficheros:

Command.COM	Diskcomp.COM
Config.SYS	Format.COM
Autoexec.BAT	Scandisk.EXE
Diskcopy.COM	Help.COM
Edit.COM	Scan.EXE
Xcopy.COM	Clean.EXE
Tree.COM	Himmem.SYS

Lista 2:

Autoexec.BAT
Diskcopy.COM
Diskcomp.COM
Format.COM
Scandisk.EXE

Número de Ficheros de la Lista 2: 5

Agregar > > >

Borrar-Elemento

Borrar-Lista

Salir

Visual Basic

Ejemplo 6.

Form1 Form

Alfabética | Por categorías

(Nombre)	Form1
Appearance	1 - 3D
AutoRedraw	True
BackColor	False
Width	15360
WindowState	2 - Maximized

Borrar-Elemento

B_BElemento CommandButton

Alfabética | Por categorías

(Nombre)	B_BElemento
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	Borrar-Elemento

Agregar > > >

B_Agregar CommandButton

Alfabética | Por categorías

(Nombre)	B_Agregar
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	Agregar > > >

Borrar-Lista

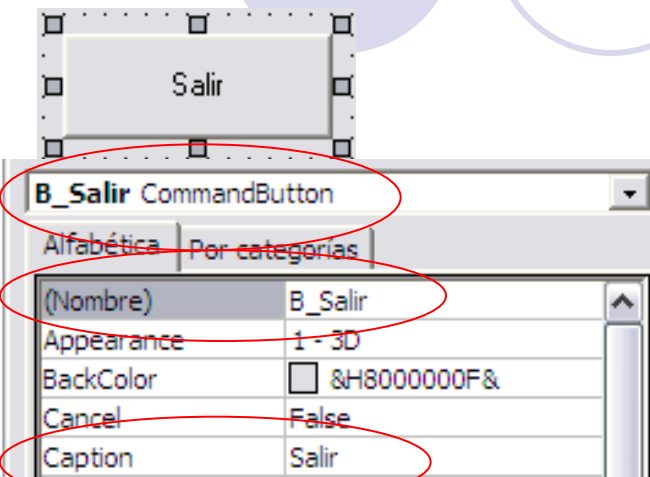
B_BLista CommandButton

Alfabética | Por categorías

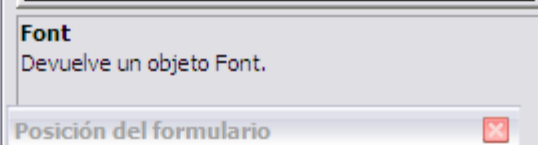
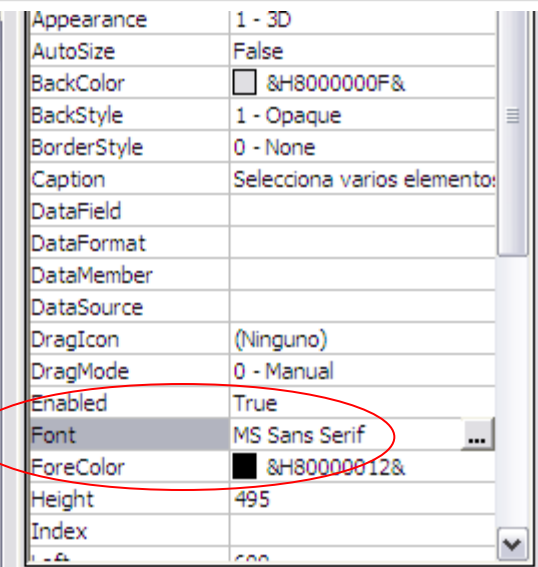
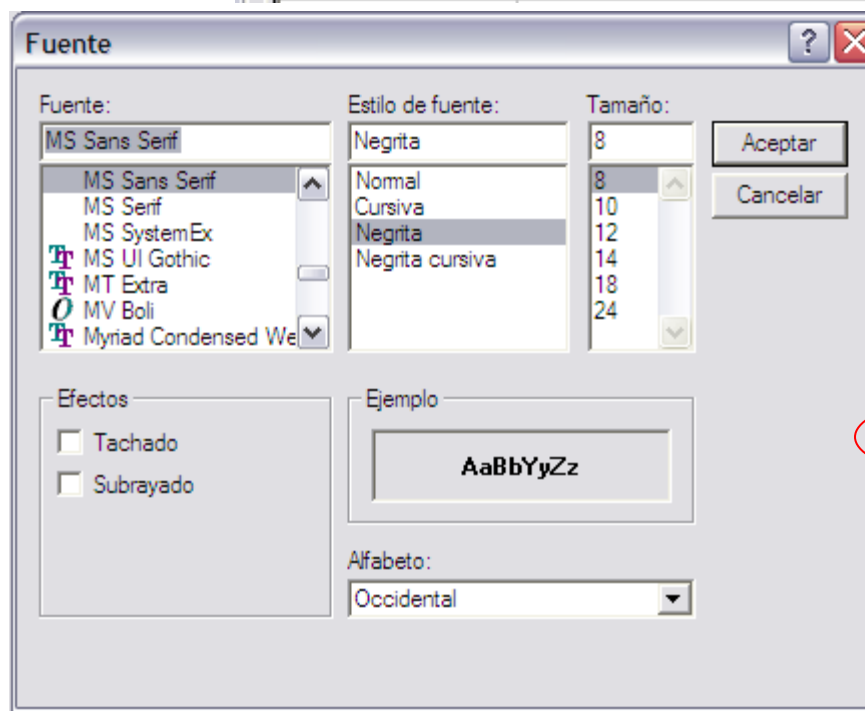
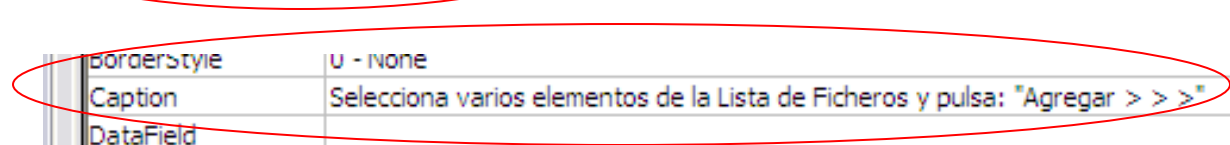
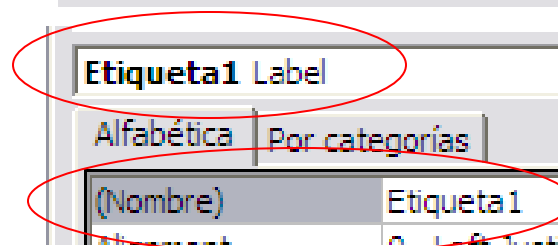
(Nombre)	B_BLista
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	Borrar-Lista

Visual Basic

Ejemplo 6.



Selecciona varios elementos de la Lista de Ficheros y pulsa: "Agregar > > >"



Lista de Ficheros:

EtiquetaLF Label

Alfabética Por categorías

(Nombre)	EtiquetaLF
Alignment	0 - Left Justify
Appearance	1 - 3D
AutoSize	True
BackColor	<input type="checkbox"/> &H8000000F&
BackStyle	1 - Opaque
BorderStyle	0 - None
Caption	Lista de Ficheros:

Font Bangle

ForeColor ☐ &H000000FF&

Height 300

Lista 2:

EtiquetaL2 Label

Alfabética Por categorías

(Nombre)	EtiquetaL2
Alignment	0 - Left Justify
Appearance	1 - 3D
AutoSize	True
BackColor	<input type="checkbox"/> &H8000000F&
BackStyle	1 - Opaque
BorderStyle	0 - None
Caption	Lista 2:

Font Bangle

ForeColor ☐ &H000000FF&

Height 300

AutoSize	True
BackColor	<input type="checkbox"/> &H8000000F&
BackStyle	1 - Opaque
BorderStyle	0 - None
Caption	Lista de Ficheros:
DataField	
DateFormat	
DataMember	
DataSource	
DragIcon	(Ninguno)
DragMode	0 - Manual
Enabled	True
Font	Bangle
ForeColor	<input type="checkbox"/> &H000000FF&
Height	300
Index	

Font
Devuelve un objeto Font.

Posición del formulario

Fuente

Fuente: Bangle

Estilo de fuente: Negrita

Tamaño: 12

Aceptar Cancelar

Efectos

☐ Tachado

☐ Subrayado

Ejemplo

AaBbYyZz

Alfabeto: Occidental

Número de Ficheros de la Lista 2:

EtiquetaNF Label

Alfabética | Por categorías

(Nombre)	EtiquetaNF
Alignment	0 - Left Justify
Appearance	1 - 3D
AutoSize	True
BackColor	<input type="checkbox"/> &H8000000F&
BackStyle	1 - Opaque
BorderStyle	0 - None
Caption	Número de Ficheros de la Lista 2:

Total_L2 TextBox

Alfabética | Por categorías

(Nombre)	Total_L2
Alignment	2 - Center
Appearance	1 - 3D

Id	
Text	
ToolTipText	

Lista 2:

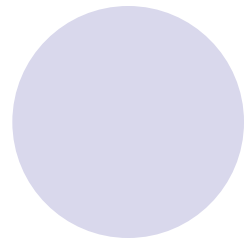
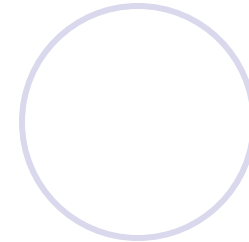
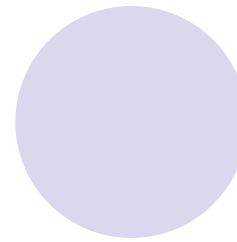
Lista_L2 ListBox

Alfabética | Por categorías

(Nombre)	Lista_L2
Appearance	1 - 3D

Lista de Ficheros:

--



Lista_Ficheros ListBox

Alfabética | Por categorías

(Nombre)	Lista_Ficheros
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H800000005&
CausesValidation	True
Columns	2
DataSource	
MousePointer	0 - Default
MultiSelect	2 - Extended
OLEDragMode	0 - Manual

Option Explicit

```
Private Sub Form_Load()  
    ' Con "AddItem" añadimos elementos a una lista.  
    Lista_Ficheros.AddItem "Command.COM"  
    Lista_Ficheros.AddItem "Config.SYS"  
    Lista_Ficheros.AddItem "Autoexec.BAT"  
    Lista_Ficheros.AddItem "Diskcopy.COM"  
    Lista_Ficheros.AddItem "Edit.COM"  
    Lista_Ficheros.AddItem "Xcopy.COM"  
    Lista_Ficheros.AddItem "Tree.COM"  
    Lista_Ficheros.AddItem "Diskcomp.COM"  
    Lista_Ficheros.AddItem "Format.COM"  
    Lista_Ficheros.AddItem "Scandisk.EXE"  
    Lista_Ficheros.AddItem "Help.COM"  
    Lista_Ficheros.AddItem "Scan.EXE"  
    Lista_Ficheros.AddItem "Clean.EXE"  
    Lista_Ficheros.AddItem "Himmem.SYS"  
    Lista_Ficheros.AddItem "Emm386.EXE"  
    Lista_Ficheros.AddItem "Keyb.COM"  
  
End Sub
```

```
Private Sub B_Agregar_Click()  
    Rem Se ha pulsado el botón "Agregar > > >".  
  
    Dim I As Integer  
  
    For I = 0 To Lista_Ficheros.ListCount - 1  
        Rem "ListCount" nos da el total de elementos de la lista  
        If Lista_Ficheros.Selected(I) Then  
            Rem Si el elemento de la lista esa "marcado" lo comprobamos  
            Rem viendo si "Selected" esta a True.  
            Lista_L2.AddItem Lista_Ficheros.List(I)  
            Rem Añadimos el elemento de la lista "Lista_Ficheros" a  
            Rem la lista "Lista_L2" mediante "List".  
        End If  
    Next I  
    Total_L2.Text = CStr(Lista_L2.ListCount)  
    Rem Introducimos el total de elementos de la lista "Lista_L2".  
End Sub
```

```
Private Sub B_BElemento_Click()  
    ' Si se ha seleccionado, anteriormente, algún elemento de la  
    ' lista "Lista_L2", al pulsar este botón, se borrará.  
    If Lista_L2.ListIndex <> -1 Then  
        ' Si "ListIndex" es igual a "-1" quiere decir que no hay  
        ' ningún elemento seleccionado en la lista "Lista_L2".  
        Lista_L2.RemoveItem Lista_L2.ListIndex  
        ' Si hay algún elemento seleccionado, la posición nos  
        ' la da "ListIndex" y lo borramos con "RemoveItem".  
        Total_L2.Text = CStr(Lista_L2.ListCount)  
    End If  
End Sub
```

```
Private Sub B_BLista_Click()  
    ' Al pulsar este botón se borrará la lista "Lista_L2" entera.  
    Lista_L2.Clear  
    Total_L2.Text = CStr(Lista_L2.ListCount)  
End Sub
```

```
Private Sub Lista_Ficheros_DblClick()  
    ' Cuando hagamos un Doble Click sobre alguno de los elementos  
    ' de "Lista_Ficheros", éste se añadadirá a la lista "Lista_L2".  
    Lista_L2.AddItem Lista_Ficheros.Text  
    Total_L2.Text = CStr(Lista_L2.ListCount)  
End Sub
```

```
Private Sub B_Salir_Click()  
    End  
End Sub
```

Visual Basic

Matrices.

- Las matrices nos permiten hacer referencia por el mismo nombre a una serie de variables y usar un número (índice) para distinguirlas. Las matrices tienen un límite superior y un límite inferior y los elementos de la matriz son contiguos dentro de esos límites. Puesto que Visual Basic asigna espacio para cada número de índice, debemos evitar declarar matrices más grandes de lo necesario.
- Todos los elementos de la matriz tienen el mismo tipo de datos.
- Formato:

Public|Private|Dim *Nombre_Matriz* (*Longitud_Matriz*) **As** *Tipo_Matriz*

Dim Contador (14) As Integer ‘ 15 elementos.

Dim Sumas (20) As Double ‘ 21 elementos.

Public Contador (14) As Integer ‘ 15 elementos.

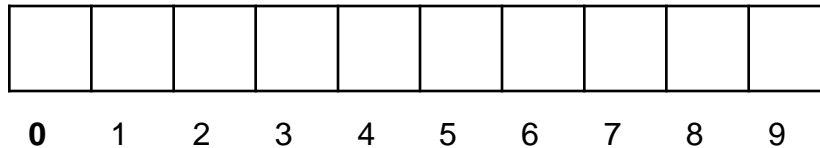
Public Sumas (20) As Double ‘ 21 elementos.

Visual Basic

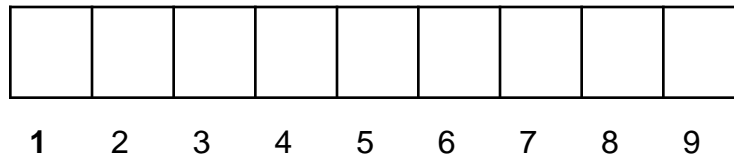
Matrices.

- Por defecto las matrices comienzan por cero.

Dim Valores (9) As Integer



- Con la sentencia **Option Base 1**, conseguimos que el vector comience desde la posición 1.



- En el caso de que no conozcamos los límites del vector, podemos obtener sus límites inferiores y superiores con las funciones:

LBound (Nombre_Vector) ‘ Devuelve el límite inferior del vector.

UBound (Nombre_Vector) ‘ Devuelve el límite superior del vector.

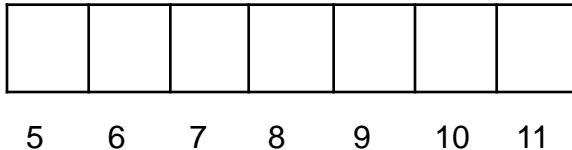
Visual Basic

Matrices.

- Podemos especificar, tanto el límite inferior como el límite superior. De esta forma no tenemos que preocuparnos por la sentencia *Option Base* ni por el límite inferior por defecto.

Public|Private|Dim *Nombre_Matriz* (*Posición_Inicial To Posición_Final*) **As** *Tipo_Matriz*

Dim Vector2 (5 To 11) As Integer



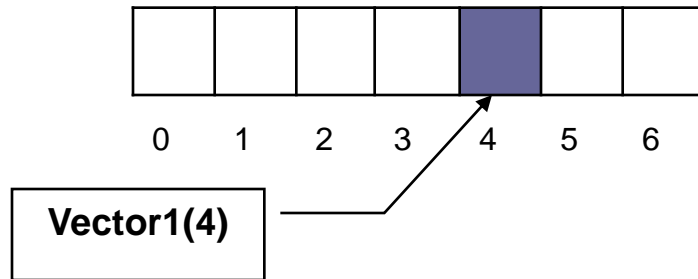
- Para acceder a un elemento del vector utilizamos el formato:

Nombre_Vector (*índice*)

Visual Basic

Matrices.

Dim Vector1 (6) As Integer



- La opción *Option Base 1* también funciona con matrices bidimensionales.
- Formato:

Public|Private|Dim Nombre_Matriz (Número_Columnas, Número_Filas) **As** Tipo_Matriz

Visual Basic

Matrices.

Ejemplo:

Dim Mat1 (4,9) As Integer

	0	1	2	3	4
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					

- También se puede acotar el rango de la matriz.

Visual Basic

Matrices.

Ejemplo:

Dim Mat3 (5 To 9 , 7 To 16) As Byte

	5	6	7	8	9
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

- Para acceder a un elemento de la matriz, se utiliza el formato:

Nombre_Matriz (índice_Columna, índice_Fila)

Visual Basic

Matrices.

Ejemplo:

Dim Mat4 (4 **To** 8 , 10 **To** 16) As String

	4	5	6	7	8
10					
11					
12					
13					
14					
15					
16					

Mat4(6, 12)

- Podemos crear matrices de más dimensiones:

Dim MultiD (3, 1 **To** 10, 1 **To** 15) as Double

- Esta declaración crea una matriz que tiene 3 dimensiones con tamaños de 4 por 10 y por 15. El número total de elementos de esta matriz sería de: $4 \times 10 \times 15 = 600$ elementos.

Visual Basic

Matrices.

- Para declarar una **Matriz Dinámica** utilizaremos las mismas sentencias que ya conocemos, Dim, Public o Private, pero dejaremos los paréntesis, que caracterizan la declaración de un matriz, totalmente **vacíos**, sin indicar límites.

Public|Private|Dim Nombre_Matriz () **As** Tipo_Matriz

- Una vez conozcamos el tamaño necesitado, generalmente a partir de parámetros o cálculos obtenidos en un procedimiento o función, usaremos la sentencia **ReDim** para indicar los límites de cada una de las dimensiones que existan.
- Usaremos la opción **Preserve**, que pondremos detrás de **ReDim** y antes del nombre de la matriz a redimensionar, para preservar los valores ya existentes de la matriz en el caso de que ésta crezca (al redimensionar una matriz utilizando **Preserve** no se puede alterar el número de dimensiones, ni el límite inferior).

ReDim Preserve MatrizNueva (UBound(MatrizNueva) + 2)

Visual BasicEjemplo 7.

- En este ejemplo tratamos los límites dinámicos de una matriz. Podemos ir cambiando los límites inferiores y superiores de la matriz y “redimensionarla” cada vez que pulsamos su botón (introduciendo valores aleatorios en la matriz para ver la comprobación).

The screenshot shows a Visual Basic form titled "Form1". On the left, there is a list of values with their corresponding indices:

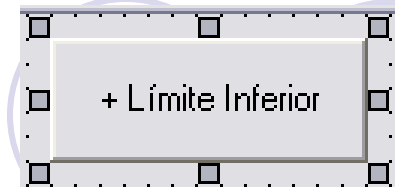
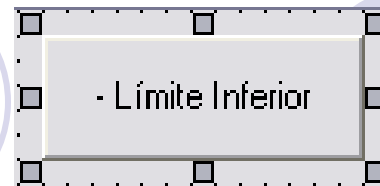
Índice	Valor
-5	137
-4	149
-3	75
-2	78
-1	199
0	5
1	195
2	209
3	182

On the right side of the form, there are several buttons for controlling the matrix limits and dimensions:

- Límite Inferior
- + Límite Inferior
- Límite Superior
- + Límite Superior
- Redimensionar
- Imprimir (highlighted with a dashed border)
- Salir

Visual Basic

Ejemplo 7.



Form1 Form

Alfabética	Por categorías
(Nombre)	Form1
Appearance	1 - 3D
AutoRedraw	True
WhatsThisHelp	False
Width	15360
WindowState	2 - Maximized

DeclimiteInf CommandButton

Alfabética	Por categorías
(Nombre)	DeclimiteInf
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	- Límite Inferior
CausesValidation	True

InclimiteInf CommandButton

Alfabética	Por categorías
(Nombre)	InclimiteInf
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	+ Límite Inferior
...	



DeclimiteSup CommandButton

Alfabética	Por categorías
(Nombre)	DeclimiteSup
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	- Límite Superior
...	

InclimiteSup CommandButton

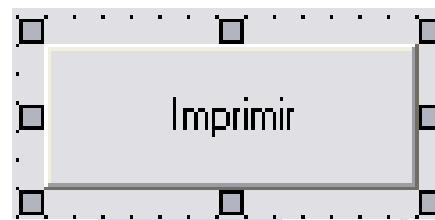
Alfabética	Por categorías
(Nombre)	InclimiteSup
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	+ Límite Superior
CausesValidation	True

Visual Basic

Ejemplo 7.



Redimensionar CommandButton	
Alfabética	Por categorías
(Nombre)	Redimensionar
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	Redimensionar
CausesValidation	True



Imprimir CommandButton	
Alfabética	Por categorías
(Nombre)	Imprimir
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	Imprimir
CausesValidation	True



Salir CommandButton	
Alfabética	Por categorías
(Nombre)	Salir
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Cancel	False
Caption	Salir
CausesValidation	True

```
Option Explicit
Dim Inferior As Integer
Dim Superior As Integer
Dim Numeros() As Byte
    ' Definimos "Numeros" como una matriz Dinámica.
```

```
Private Sub DecLimiteInf_Click()
    Inferior = Inferior - 1
    'Decrementamos el límite inferior de la matriz dinámica "Numeros".
End Sub
```

```
Private Sub DecLimiteSup_Click()
    Superior = Superior - 1
    'Decrementamos el límite superior de la matriz dinámica "Numeros".
End Sub
```

```
Private Sub IncLimiteInf_Click()
    Inferior = Inferior + 1
    'Aumentamos el límite inferior de la matriz dinámica "Numeros".
End Sub
```

```
Private Sub IncLimiteSup_Click()
    Superior = Superior + 1
    'Aumentamos el límite superior de la matriz dinámica "Numeros".
End Sub
```



```
Private Sub Redimensionar_Click()  
    ReDim Numeros(Inferior To Superior)  
    ' Cada vez que ejecutamos esta instrucción, establecemos los  
    ' los nuevos límites de la matriz dinámica "Numeros".  
   Cls  
    Print "El Límite INFERIOR es: "; LBound(Numeros)  
    Print "El Límite SUPERIOR es: "; UBound(Numeros)  
    ' Visualizamos en el formulario los nuevos límites de la  
    ' matriz "Numeros".  
    Dim I As Integer  
    For I = LBound(Numeros) To UBound(Numeros)  
        Numeros(I) = CByte(Rnd * 255 + 1)  
        ' Rellenamos con valores aleatorios la matriz "Numeros"  
        ' desde el nuevo límite inferior hasta el nuevo límite  
        ' superior.  
    Next I  
End Sub
```

```
Private Sub Imprimir_Click()  
    Dim I As Integer  
    For I = LBound(Numeros) To UBound(Numeros)  
        Print I; " --> "; Numeros(I)  
        'Imprimimos la matriz dinámica "Numeros".  
    Next I  
End Sub
```

```
Private Sub Salir_Click()  
    End  
End Sub
```


Visual Basic

Matrices de Controles.

- Una *matriz de controles* es un grupo de controles que comparten el mismo nombre y el mismo tipo. También comparten los mismos procedimientos de evento. Las *matrices de controles* tienen al menos un elemento y pueden crecer hasta contener tantos elementos como permitan los recursos y la memoria del sistema; su tamaño depende también de la memoria y de los recursos de Windows requeridos para cada control. El índice máximo que puede usar una matriz de controles es 32767. Los elementos de una misma matriz de controles tienen su propio valor de propiedades.
- Mediante el mecanismo de las matrices de controles, cada nuevo control hereda los procedimientos de evento comunes escritos para la matriz.

Visual Basic

Matrices de Controles.

- Hay 3 maneras de crear una matriz de controles en tiempo de diseño:
 - Asignar el mismo nombre a más de un control.
 - Copiar un control existente y después pegarlo en el formulario.
 - Establecer la propiedad **Index** del control a un valor distinto de **Null**.
- Operaciones para crear una matriz de controles.
 - Dibujar un control del cuadro de herramientas (controles) en el formulario.
 - Mientras el control “tiene el enfoque”, seleccionamos **Copiar** en el menú **Edición**.
 - En el menú **Edición**, seleccionamos **Pegar**. Visual Basic presenta un cuadro de diálogo que nos pide que confirmemos la creación de una matriz de controles. Hacemos clic en **Si** para confirmar la acción.
 - A este control se le asigna el índice 1. El primer control que dibujamos tiene el índice 0.

Visual Basic

Matrices de Controles.

- Por supuesto también podemos crear una matriz de controles a partir de una serie de controles que ya tengamos insertados en el formulario; lo único que debemos hacer es ir modificando la propiedad *Name*, asignándole le mismo nombre a todos.

Visual Basic

Ejemplo 8.

- Este programa tiene 2 finalidades: 1) Introducir un número en decimal (de 3 cifras y menor de 256) para pasarlo a binario ó 2) introducir un número en binario (de 8 bits) para pasarlo a decimal utilizando matrices de controles.

The screenshot shows a Visual Basic form titled "Form1". It contains a label "Número Decimal:" followed by a text box containing the number "83". To the right of the text box is the instruction "Entrar el número y pulsar ENTER.". Below this, there is a row of eight checkboxes labeled "Bits" on the left. The checkboxes are numbered 7, 6, 5, 4, 3, 2, 1, and 0 from left to right. Checkboxes 6, 4, 1, and 0 are checked, while 7, 5, 3, and 2 are unchecked. To the right of the checkboxes is a button labeled "Salir".

Bits	7	6	5	4	3	2	1	0
	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Visual Basic

Ejemplo 8.

- Las etiquetas (*Label*) “Número Decimal:”, “Entrar el número y pulsar ENTER” y “Bits” no tienen nada de especial. No las describimos. El botón “Salir” es como siempre. Las etiquetas (*Label*) “0”, “1”, “2”, ... y “7” las podéis definir como una matriz de controles pero no afecta a la programación del “programa”.

Visual Basic

Ejemplo 8.

Form1 Form

Alfabética Por categorías

WindowState	2 - Maximized
-------------	---------------

Terminal

pul

Numero TextBox

Alfabética Por categorías

(Nombre)	Numero
Alignment	2 - Center
Appearance	1 - 3D

Locked	raise
MaxLength	3
MouseIcon	(Ninguno)

Tag	
Text	
ToolTipText	

Visual Basic

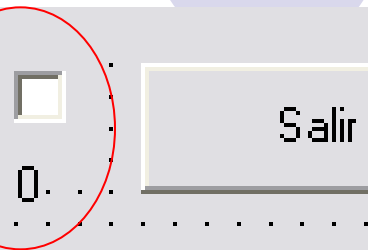
Ejemplo 8.

- Cuando creamos la Matriz de Controles (de Cuadros de Control, en este caso) debemos seleccionar uno, copiarlo al formulario, seleccionarlo, indicar que se quiere copiar mediante la opción “Copiar” del menú “Edición” y luego “Pegarlo”. En este momento parecerá un mensaje que nos dirá:

Ya existe un control llamado ‘Check1’. ¿Desea crear una matriz de controles?
- Le respondemos que “Si” y continuamos con el mismo proceso de copiar y pegar hasta obtener los 8 controles en matriz.
- En este caso es importante el orden en el que coloquemos los controles en el formulario. El de índice más bajo (0) es el que debe estar a tu derecha y el de índice más alto (7) a tu izquierda.

Visual Basic

Ejemplo 8.



Bit(0) CheckBox	
Alfabetica	Por categorías
(Nombre)	Bit
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Caption	
CausesValidation	True
HelpContextID	0
Index	0
Left	3240
UseMaskColor	False
Value	0 - Unchecked
Visible	True



Bit(7) CheckBox	
Alfabetica	Por categorías
(Nombre)	Bit
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H8000000F&
Caption	
CausesValidation	True
HelpContextID	0
Index	7
Left	770
UseMaskColor	False
Value	0 - Unchecked

Option Explicit

Private Sub Form_Load()

Numero.Text = "0"

'Inicializamos donde vamos a introducir el número a 0.

End Sub

Private Sub Bit_Click(Index As Integer)

' Cada vez que "marcamos" un cuadro de control pasamos por aquí.

' Aquí convertimos los cuadros de control "marcados" y "desmarcados"

' a decimal.

If Bit(Index) = 0 Then

Numero.Text = CStr(CByte(Numero.Text) And Not 2 ^ Index)

' Hemos pulsado un cuadro de control y lo hemos "desmarcado"

' entonces tenemos que ajustar el número decimal al nuevo

' valor binario.

Else

Numero = CStr(CByte(Numero.Text) Or 2 ^ Index)

' Hemos pulsado un cuadro de control y lo hemos "marcado"

' entonces tenemos que ajustar el número decimal al nuevo

' valor binario.

End If

End Sub

```
Private Sub Numero_KeyPress(KeyAscii As Integer)
    ' Cada vez que pulsamos una tecla en "Numero" pasamos por aquí.
    If KeyAscii <> 13 Then      ' Tecla 13 = ENTER.
        Exit Sub
        ' Hasta que no pulsemos ENTER no seguimos con la ejecución
        ' de estas instrucciones.
    End If
    If CInt(Numero.Text) > 255 Then
        MsgBox ("El número ha de estar en el rango - 255")
        Exit Sub
        ' Hemos pulsado ENTER y el número introducido es mayor que 255.
        ' Lo notificamos pero no continuamos con la ejecución de las
        ' siguientes instrucciones.
    End If
    Dim I As Byte
        ' Hemos pulsado ENTER y el número es menor que 255.
        ' Lo convertimos a binario.
        ' Recorremos los 8 cuadros de control para comprobar cual está
        ' marcado (1) y cual no (0).
    For I = 0 To 7
        If CByte(Numero.Text) And 2 ^ I Then
            Bit(I).Value = 1
        Else
            Bit(I) = 0
        End If
    Next I
End Sub
```

```
Private Sub Salir_Click()
    End
End Sub
```

Visual Basic

Pruebas.

The screenshot shows a Windows-style window titled "Form1" with a blue title bar. The main area is light gray. On the left, the text "Hola", "Buenos", and "Días" is displayed in red. On the right, there is a label "Escribe un texto:" followed by a text box containing the word "Días". Below the text box are two buttons: "Aceptar" (highlighted with a dotted border) and "Salir".

Segundo (112):

The screenshot shows a Windows-style window titled "Form1" with a blue title bar. The main area is light gray. On the left, the text "Estamos en Prácticas" is repeated ten times in different colors: red, blue, green, yellow, red, red, red, red, red, and blue. To the right of this list is a label "Escribe un texto:" followed by a text box containing the text "Estamos en Prácticas" in blue. Below the text box are two columns of buttons. The left column contains "Rojo", "Azul" (highlighted with a dotted border), and "Tamaño +". The right column contains "Verde", "Amarillo", "Tamaño -", and "Salir".

Pruebas.

Tercero (113):

Form1

Tienda Dolores

Artículos

- ☒ Libretas
- ☐ Platos
- ☐ Velas
- ☒ Jarrones
- ☒ Herramientas
- ☒ Relojes

Proveedores

- ☐ Proveedor 1
- ☐ Proveedor 2
- ☒ Proveedor 3

Las Libretas las compramos en el Proveedor 3.
Los Jarrones los compramos en el Proveedor 3.
Las Herramientas las compramos en el Proveedor 3.
Los Relojes los compramos en el Proveedor 3.

Pruebas.

Tercero (114):

Conversión::	
Número:	
0	Octal
1	Hexadecimal
2	Binario
3	
4	Salir
5	
6	
7	
8	
9	

Visual Basic

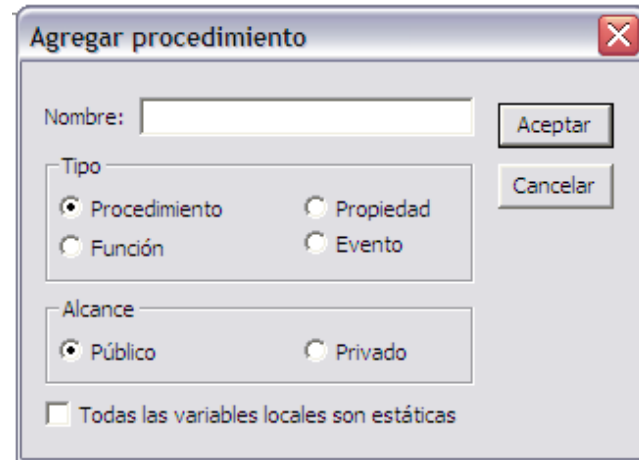
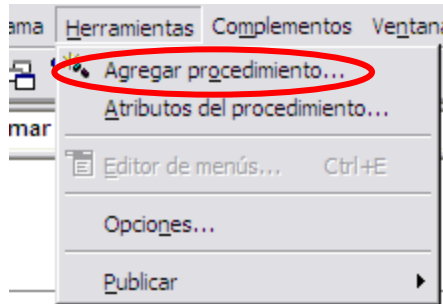
Procedimientos y Funciones.

- Cuando un programa tiene alguna finalidad más que la simple comprobación o prueba de un control o un operador, la cantidad de código necesaria crecerá, y en ocasiones necesitaremos el mismo código en distintos puntos del programa. Obviamente la solución no es introducir una y otra vez todo el código, tanto si se repite como si no. Para evitarlo podemos crear nuestros propios procedimientos y funciones, introduciendo en ellos el código que después podrá ser utilizado con una simple llamada.
- La única diferencia existente entre un **Procedimiento** y una **Función** es que el primero no puede devolver parámetro alguno, mientras que la segunda si.

Visual Basic

Procedimientos y Funciones.

- Para crear un nuevo **Procedimiento** lo más cómodo es abrir el menú “Herramientas” y elegir la opción “Agregar procedimiento”.



- Aparece una ventana en la que facilitaremos el nombre del Procedimiento, la opción para elegir entre Procedimiento, Función, Procedimiento de Propiedad y Procedimiento de Evento. También indicaremos si el procedimiento será público o privado. También podemos indicar si todas las variables a utilizar van a ser estáticas: “Todas las variables locales son estáticas”.

Visual Basic

Procedimientos y Funciones.

- Formato de un Procedimiento:

[Public|Private] [Static] **Sub** *nombre_procedimiento* (*parámetros*)

 [**Dim** *variables_locales*]

instrucciones...

End Sub

- Formato de una Función:

[Public|Private] [Static] **Function** *nombre_función* (*parámetros*) **As** *tipo_valor_retorno*

 [**Dim** *variables_locales*]

instrucciones...

End Function

Visual Basic

Procedimientos y Funciones.

- Podemos crear un procedimiento tanto en un Formulario como en un Módulo separado. Los procedimientos y funciones pueden ser, al igual que las variables, **Públicos** o **Privados**, según sean utilizados desde el módulo en que se han definido o desde cualquier otro módulo de la aplicación. Por defecto, cualquier Procedimiento o Función que se defina en el Formulario es Privado, mientras que cualquiera que se defina en un Módulo de Código es Público. Para modificar esto, podemos preceder a la palabra **Sub** o **Function** con **Public** o **Private**.
- Para no perder el valor de las variables locales cada vez que salgamos de un procedimiento, se definirán con Static (o al principio):

Static *nombre_variable* **As** *tipo*

Visual Basic

Procedimientos y Funciones.

- Si se pasa como parámetro un **Vector**, en la declaración del parámetro, NO PONDREMOS EL NÚMERO DE ELEMENTOS, lo que nos permitirá recibir cualquier número de elementos sin conocerlos de antemano.

```
Dim Numeros (5 To 10) As String
```

```
.....
```

```
Numeros (5) = "Uno"
```

```
Numeros (6) = "Dos"
```

```
Numeros (7) = "Tres"
```

```
Numeros (8) = "Cuatro"
```

```
Numeros (9) = "Cinco"
```

```
Numeros (10) = "Seis"
```

```
.....
```

```
MuestraElementos Numeros            ' O Call MuestraElementos (Numeros)
```

```
.....
```

```
Private Sub MuestraElementos (Nombre_Numeros ( ) As String)
```

```
    Dim I As Integer
```

```
    For I = LBound(Nombre_Numeros) To UBound(Nombre_Numeros)
```

```
        Print Nombre_Numeros (I)
```

```
    Next I
```

```
End Sub
```

Visual Basic

Procedimientos y Funciones.

Dim **Resultado** As Long

.....

Resultado = **SumaCuadrados** (3,5)

‘ Los Paréntesis son obligatorios en las
‘ llamadas a funciones.

.....

Private Function **SumaCuadrados** (N1 As Integer, N2 As Integer) As Long

SumaCuadrados = N1 * N1 + N2 * N2

End Function

Visual Basic

Procedimientos y Funciones.

- Parámetros por Referencia.
- Cuando llamamos a un procedimiento, y pasamos como parámetro una variable (no una constante), ésta se puede recibir por **Referencia** o por **Valor**. En el primer caso (**Referencia**) el procedimiento lo que recibe es una referencia a la misma variable que se pasa como parámetro, o lo que es lo mismo, *recibe la propia variable*. Esto significa que *si el procedimiento realiza cualquier modificación, la variable original se verá afectada*.
- **Por defecto el parámetro es recibido por referencia.**
- Para indicar que un parámetro se pasa por referencia y no por valor, debemos utilizar **ByRef**.

```
Private Sub nombre_procedimiento (ByRef parámetros)
    Dim variables_locales
    .....
End Sub
```

Visual Basic

Procedimientos y Funciones.

Dim Numero As Long

.....

Numero = 625

Print RaizCuarta (Numero), Numero

.....

Public Function RaizCuarta (Operando As Long) As Long

Operando = Sqr (Operando)

RaizCuarta = Sqr (Operando)

End Function

- La función **RaizCuarta** utiliza el propio parámetro que recibe (*Operando*) para realizar una operación intermedia ($\text{Operando} = \text{Sqr}(\text{Operando})$), y dado que **por defecto el parámetro es recibido por referencia**, la variable **Numero** original se habrá visto afectada, por lo que al imprimirla no aparecerá: 5, 625, sino: 5, **25**.

Visual Basic

Procedimientos y Funciones.

- Parámetros por Valor.
- Cuando una variable se recibe por **Valor**, lo que el procedimiento obtiene es **una copia** del contenido de la variable, pero NO la variable original, por lo que *cualquier modificación que se efectúe, no alterará su valor*. Para indicar que un procedimiento se pase por valor, debemos utilizar **ByVal**.

```
Private Sub nombre_procedimiento (ByVal parámetros)
    Dim variables_locales
    ....
End Sub
```

Visual Basic

Procedimientos y Funciones.

Dim Numero As Long

.....

Numero = 625

Print RaizCuarta (Numero), Numero

.....

Public Function RaizCuarta (**ByVal** Operando As Long) As Long

Operando = Sqr (Operando)

RaizCuarta = Sqr (Operando)

End Function

- La función **RaizCuarta** utiliza el propio parámetro que recibe (*Operando*) para realizar una operación intermedia (Operando = Sqr(Operando)), y dado que esta vez el parámetro es pasado por valor (**ByVal**), la variable **Numero** original NO se habrá visto afectada, por lo que al imprimirla aparecerá: 5, **625**.

Visual Basic

Exit Sub y Exit Function.

- Hasta ahora nuestros procedimientos se han ejecutado desde principio a fin terminando al llegar a la línea **End Sub**, que provocaba que el control vuelva a la sentencia siguiente a la que provocó la llamada. Disponemos de unas sentencias que nos permitirán finalizar la ejecución de un procedimiento de forma anticipada.
- Para provocar la salida de un procedimiento utilizaremos **Exit Sub**, mientras que para realizar la misma operación en una función usaremos **Exit Function**.
- Generalmente estas sentencias están asociadas a alguna estructura decisiva, ya que si las disponemos directamente como una sentencia ejecutable más, provocará la salida del procedimiento siempre en el mismo punto, impidiendo así la ejecución del resto del código que exista.

Visual Basic

Ejemplo 9.

- El siguiente programa realiza las operaciones de Suma, Resta y Producto de los 2 operandos introducidos. Lo vamos a realizar mediante funciones.

The screenshot shows a Visual Basic form titled "Form1". It contains the following elements:

- Two input fields for operands: "Operando 1:" with the value "789" and "Operando 2:" with the value "123".
- A label "Resultado:" followed by an output field displaying the value "666".
- A group box labeled "Operar" containing three radio buttons: "Sumar", "Restar" (which is selected), and "Multiplicar".
- A "Salir" button.

o 1:

2:

Numero1 TextBox

Alfabética | Por categorías

(Nombre)	Numero1
Alignment	2 - Center
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H80000005&

Numero2 TextBox

Alfabética | Por categorías

(Nombre)	Numero2
Alignment	2 - Center
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H80000005&

Resultado:

Resultado TextBox

Alfabética | Por categorías

(Nombre)	Resultado
Alignment	2 - Center
Appearance	1 - 3D
BackColor	<input type="checkbox"/> &H80000005&

☐ Sumar

☐ Restar

OpciónSuma **OptionButton**

Alfabética

Por categorías

(Nombre)

OpciónSuma

Alignment

0 - Left Justify

OpciónResta **OptionButton**

Alfabética

Por categorías

(Nombre)

OpciónResta

Alignment

0 - Left Justify

☐ Multiplicar

OpciónMultiplicar **OptionButton**

Alfabética

Por categorías

(Nombre)

OpciónMultiplicar

Alignment

0 - Left Justify

Option Explicit

```
Private Sub Form_Load()  
    Numero1.Text = "0"  
    Numero2.Text = "0"  
    Resultado.Text = "0"  
    Resultado.Enabled = False  
End Sub
```

```
Private Sub OpciónSuma_Click()  
    Resultado.Text = Suma(Numero1.Text, Numero2.Text)  
End Sub
```

```
Private Sub OpciónResta_Click()  
    Resultado.Text = Resta(Numero1.Text, Numero2.Text)  
End Sub
```

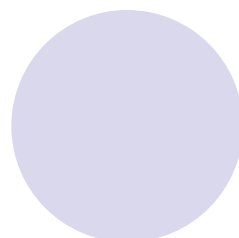
```
Private Sub OpciónMultiplicar_Click()  
    Resultado.Text = Producto(Numero1.Text, Numero2.Text)  
End Sub
```

```
Private Sub BotónSalir_Click()  
    End  
End Sub
```

```
Private Function Suma(Suma1 As String, Suma2 As String) As String  
    Suma = CStr(CInt(Suma1) + CInt(Suma2))  
End Function
```

```
Private Function Resta(Resta1 As String, Resta2 As String) As String  
    Resta = CStr(CInt(Resta1) - CInt(Resta2))  
End Function
```

```
Private Function Producto(Producto1 As String, Producto2 As String) As String  
    Producto = CStr(CInt(Producto1) * CInt(Producto2))  
End Function
```



Visual Basic

Ejemplo 1144.

- Realiza un programa similar al realizado en el ejemplo 114, pero **UTILIZANDO FUNCIONES**.

Número:

Conversión::

0 1 2

3 4 5

6 7 8

9

Octal Hexadecimal Binario

Salir

Ejemplo 91.

The screenshot shows a Visual Basic application window titled "Form1". The window has a light gray background. At the top, there are four input fields arranged in two rows. The first row has "Introduce Libro:" followed by a text box containing "El Nombre de la Rosa" in red text. The second row has "Longitud Libro:" followed by a spin box containing the number "21". The third row has "Introduce Autor:" followed by a text box containing "Umberto Eco" in blue text. The fourth row has "Longitud Autor:" followed by a spin box containing the number "12" and a button labeled "Introducir". Below these fields is a list box titled "Lista Libros y Autor". The list box contains two items: "Libro: El Nombre De La Rosa" and "Autor: Umberto Eco". To the right of the list box are two buttons: "Buscar Libro" and "Salir".

El siguiente programa lo hemos dividido en 2 partes. Por un lado vamos a introducir “Títulos de Libros” y sus “Autores” en una lista y vamos a “medir” el tamaño de cada una de sus cadenas. Cuando pulsemos “Introducir” pasarán a la lista “Lista Libros Autor”. Si pulsamos “Salir” salimos de la aplicación. Y si pulsamos “Buscar Libro” pasamos a la otra parte del programa que hasta ahora ha estado “invisible”.

Ejemplo 91.

Cuando pulsamos “Buscar Libro” aparece la otra parte de la pantalla y desaparecen varios campos de la anterior. Aquí vamos a buscar títulos de libros que comiencen por, por lo menos, los 3 caracteres iniciales que introduzcamos en “Empieza por”.

También anotaremos el total de libros encontrados en “Total”.

The screenshot shows a Visual Basic application window with a light gray background. At the top right, there is a label "Empieza por:" followed by a text box containing the text "el". Below this, on the right side, is a list box containing three items: "Libro: El Nombre De La Rosa", "Libro: El Sabor Ácido", and "Libro: El Ciruelo". In the bottom right corner, there is a label "Total:" followed by a text box containing the number "3". In the bottom center, there are two buttons: "Buscar Libro" and "Salir". On the left side, there is a large text box containing a list of book titles and authors: "Libro: El Nombre De La Rosa", "Autor: Umberto Eco", "Libro: Camaleón", "Autor: R. S. Hams", "Libro: Cristal Y Cristal", "Autor: María P. M.", "Libro: El Sabor Ácido", "Autor: Lola M. Sanz", "Libro: El Ciruelo", and "Autor: Rosa Villas".

Introduce Libro:

Empieza por:

Longitud Libro:

Introduce Autor:

Longitud Autor:

Introducir

Lista Libros y Autor

Buscar Libro

Salir

Total:

OBJETO	COMENTARIO
Introduce Libro (et.)	Nombre = Label1
Longitud Libro (et.)	Nombre = Label2
Introduce Autor (et.)	Nombre = Label3
Longitud Autor (et.)	Nombre = Label4
Lista Libros y Autor (et.)	Nombre = Label5
Empieza por (et.)	Nombre = Label6
Total (et.)	Nombre = Label7
Introducir (bot.)	Nombre = BotonIntroducir
Buscar Libro (bot.)	Nombre = BotonBuscar
Salir (bot.)	Nombre = BotonSalir
Introduce Libro (campt.)	Nombre = IntroLibro , ForeColor = Rojo
Longitud Libro (campt.)	Nombre = LongLibro
Introduce Autor (campt.)	Nombre = IntroAutor , ForeColor = Azul
Longitud Autor (campt.)	Nombre = LongiAutor
Lista Libros Autor (lista)	Nombre = ListaLibroAutor
Empieza por (campt.)	Nombre = EmpiezaPor , MaxLength = 3
Empieza por (lista)	Nombre = ListaSeleccion
Total (campt.)	Nombre = Total
Formulario	WindowState = 2 - Maximized
Fuente	Font = MS Sans Serif, Negrita, 10

```
Dim Resultado As String
Dim I As Integer
    ' "I" es el contador que vamos a utilizar para la lista
    ' "ListaLibroAutor".
Dim J As Integer
    ' "J" es el contador que utilizamos para contar los libros que
    ' coinciden con la cadena a buscar.
Dim Z As Integer
    ' "Z" es el contador que utilizamos para contar el número de
    ' caracteres que va a tener la cadena de búsqueda.

Private Sub BotonBuscar_Click()
    Label1.Visible = False
    Label2.Visible = False
    Label3.Visible = False
    Label4.Visible = False
    Label5.Visible = False
    IntroLibro.Visible = False
    IntroAutor.Visible = False
    LongiLibro.Visible = False
    LongiAutor.Visible = False
    BotonIntroducir.Visible = False
    ' Cuando pulsamos el botón "Buscar Libro" estos campos desaparecen del
    ' formulario.
    Label6.Visible = True
    Label7.Visible = True
    EmpiezaPor.Visible = True
    ListaSeleccion.Visible = True
    Total.Visible = True
    ' Y estos campos que hasta ahora estaban invisibles pasan a visibles.
End Sub
```

```

Private Sub BotonIntroducir_Click()
    'Cuando pulsamos el "BotonIntroducir" el "Libro" y el "Autor"
    'introducidos pasan a la lista "ListaLibroAutor".
    Resultado = StrConv("Libro: " & IntroLibro.Text, vbProperCase)
    'Pasamos a la variable "Resultado" el nombre del libro introducido en
    '"IntroLibro". Lo hacemos de la siguiente forma:"Libro: " más
    'el libro introducido. Con "vbPropercase" lo que hacemos es que
    'cada caracter de inicio del cadena completa aparece en mayúsculas.
    ListaLibroAutor.AddItem Resultado
    'Pasamos el valor de "Resultado" a "ListaLibroAutor".
    Resultado = StrConv("Autor: " & IntroAutor.Text, vbProperCase)
    ListaLibroAutor.AddItem Resultado
    'Lo mismo pero con "Autor".
    LongiLibro.Text = CStr(Len(IntroLibro.Text))
    LongiAutor.Text = CStr(Len(IntroAutor.Text))
    'Ponemos el total de caracteres de la cadena de caracteres del
    'libro introducido y del autor introducido en "LongiLibro" y en
    '"LongiAutor" respectivamente.

```

End Sub

```

Private Sub BotonSalir_Click()

```

```

    End

```

```

End Sub

```

```

Private Sub EmpiezaPor_KeyPress(KeyAscii As Integer)

```

```

    If KeyAscii <> 13 Then

```

```

        'Hasta que no pulsemos "ENTER" no seguimos ejecutando este

```

```

        'procedimiento (vamos introduciendo caracteres en "EmpiezaPor"

```

```

        'hasta que pulsemos ENTER).

```

```

        Z = Z + 1

```

```

        'Vamos contando los caracteres que vamos metiendo en "EmpiezaPor".

```

```

        Exit Sub

```

```

    End If

```

```

    BuscarLibro

```

```

    'Llamamos al procedimiento: BuscarLibro.

```

```

    Total.Text = CStr(J)

```

```

    'Colocamos en "Total" el número de Libros que coinciden con la cadena

```

```

    'a buscar.

```

```

End Sub

```

```
Private Sub Form_Load()  
LongiLibro.Enabled = False  
LongiAutor.Enabled = False  
    'Desactivamos estos campos para que no puedan ser manipulados.  
Label6.Visible = False  
Label7.Visible = False  
EmpiezaPor.Visible = False  
ListaSeleccion.Visible = False  
Total.Visible = False  
    'Ponemos a INVISIBLE estos campos para que no se vean al principio.  
J = 0  
Z = 0  
    'Inicializamos contadores.  
End Sub
```

```
Private Sub IntroLibro_GotFocus()  
    IntroLibro.Text = " "  
    IntroAutor.Text = " "  
    LongiLibro.Text = " "  
    LongiAutor.Text = " "  
    'Cada vez que posicionamos el cursor en el campo: "IntroLibro"  
    'los campos: "IntroLibro", "IntroAutor", LongiLibro" y "LongiAutor"  
    'se "limpian".  
End Sub
```

```
Public Sub BuscarLibro()  
    'Procedimiento "BuscarLibro".  
    For I = 0 To ListaLibroAutor.ListCount - 1 Step 2  
        ' "Liscount" nos da el total de elementos del fichero.  
        ' Como solo queremos el Libro NO el Autor realizamos un "Step 2".  
        If UCase(Mid(ListaLibroAutor.List(I), 8, 2)) = UCase(EmpiezaPor.Text) Then  
            'Hay que saltar los 8 caracteres de:"Libro: ", por lo tanto  
            'seleccionamos el trozo de cadena a buscar desde el caracter  
            '8 de "ListaLibroAutor". A partir de ahí tomamos los caracteres que  
            'indique "Z".  
            '"EmpiezaPor" puede tomar como máximo 3 caracteres.  
            ListaSeleccion.AddItem ListaLibroAutor.List(I)  
            'Colocamos el libro que coincide en la lista "ListaSeleccion".  
            J = J + 1  
            'En "J" tenemos el número de cadenas que coinciden para  
            'luego sacarla en "Texto".  
        End If  
    Next I  
End Sub
```

Visual Basic

Función MsgBox.

- Visualiza una ventana, en cuyo interior aparece un mensaje que se pasa como parámetro, espera la pulsación de un botón y devuelve un valor según el botón pulsado.

MsgBox (*mensaje* [, *botones*] [, *título*])

- La sintaxis de la función MsgBox tiene estas partes de Argumentos con Nombre:
- **Argumentos con Nombre:** Es un argumento que tiene un nombre predefinido en la biblioteca de objetos. En lugar de proporcionar valores a los argumentos en el orden esperado por la sintaxis, es posible utilizar Argumentos con Nombre para asignar valores en cualquier orden.
- **Mensaje:** Expresión de cadena que representa el “**Mensaje**” en el Cuadro de Dialogo. La longitud máxima de un mensaje es, aproximadamente, de 1024 caracteres, según el ancho de los caracteres utilizados. Si el “**Mensaje**” se compone de más de una línea, nos debemos asegurar un Retorno de Carro (Chr(13)), o una Continuación de Línea (Chr(13) y Chr(10)) entre cada línea y la siguiente.

Visual Basic

Función MsgBox.

- **Botones:** Expresión numérica que corresponde a la suma de los valores que especifican el *Número* y *Tipo* de botones que se pretenden mostrar, el *Estilo* de icono que se va a utilizar, la *Identidad* del botón predeterminado y la *Modalidad*. Si se omite este argumento se utiliza el valor predeterminado, que es 0.
- **Título:** Expresión de Cadena que se muestra en la “Barra de Título” del cuadro de dialogo. Si se omite “Título”, no se pone nada en la Barra de Título.

Configuración del Argumento con Nombre (BOTONES).

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 SI, NO y CANCELAR.
VbYesNo	4	Muestra los botones SI y NO.
VbRetryCancel	5	Muestra los botones REINTENTAR y CANCELAR.
VBCritical	16	Muestra el icono de “Mensaje Crítico”.
VbQuestion	32	Muestra el icono de “Interrogación”.
VbExclamation	48	Muestra el icono de “Exclamación”.
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 redeterminado.
VbApplicationModal	0	Aplicación Modal: El usuario debe responder al cuadro de mensaje antes de poder continuar trabajando en la aplicación en la que se encuentra.
vbSystemModal	4096	Sistema Modal: Se suspenden todas las aplicaciones hasta que el usuario responda el cuadro de mensaje.
vbMsgBoxHelpButton	16384	Agrega el botón Ayuda al cuadro de mensaje.
vbMsgBoxRight	524288	El texto se alinea a la derecha.
vbMsgBoxRtlReading	1048576	Especifica que el texto debe aparecer para ser leído de derecha a izquierda en sistemas hebreo y árabe.
vbMsgBoxSetForeground	65536	Especifica la ventana del cuadro de mensaje como la ventana de primer plano.

Visual Basic

Función MsgBox.

- El primer grupo de valores (0 a 5) describe el *Número* y el *Tipo* de los botones mostrados en el cuadro de diálogo; el segundo grupo (16, 32, 48 y 64) describe el *Estilo* del icono, el tercer grupo (0, 256, 512) determina el botón *Predeterminado* y el cuarto grupo (0 y 4096) determina la *Modalidad* del cuadro de mensajes. Cuando se suman números para obtener el valor final del argumento BOTONES, se utiliza solamente un número de cada grupo.
- La respuesta que el operador da a la caja de diálogo, se almacena en una variable tipo **integer** que contiene alguno de los números que se incluyen a continuación. También en este caso, el programador puede consultar sobre las constantes y no sobre los valores. Puede también averiguar la respuesta del operador sin colocarla en una variable.

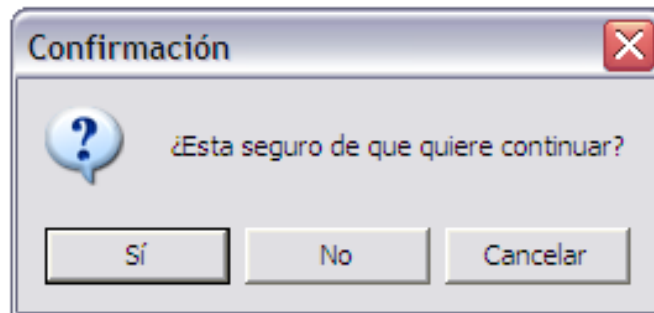
Constante	Valor	Botón Seleccionado
vbOK	1	Aceptar.
vbCancel	2	Cancelar.
vbAbort	3	Anular.
vbRetry	4	Reintentar.
vbIgnore	5	Ignorar.
vbYes	6	Si.
vbNo	7	No.

- Si deseamos especificar solamente el primer Argumento con Nombre, debemos usar MsgBox en una expresión. Si deseamos omitir algún argumento de posición, debemos incluir el delimitador de coma correspondiente.

Visual Basic

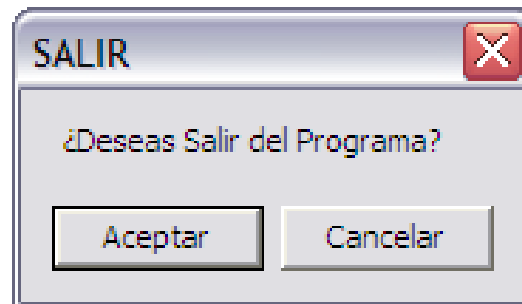
Función MsgBox.

```
Private Sub Form_Load()  
    MsgBox "¿Esta seguro de que quiere continuar?", vbYesNoCancel + vbQuestion, "Confirmación"  
End Sub
```



Option Explicit

```
Private Sub Form_Load()  
    Dim Respuesta As Integer  
    Respuesta = MsgBox("¿Deseas Salir del Programa?", 1, "SALIR")  
    ' Respuesta = MsgBox("¿Deseas Salir del Programa?", vbOKCancel, "SALIR")  
    If Respuesta = 1 Then  
        Print "ACEPTAR"  
    End If  
    If Respuesta = 2 Then  
        Print "CANCELAR"  
    End If  
End Sub
```



Visual Basic

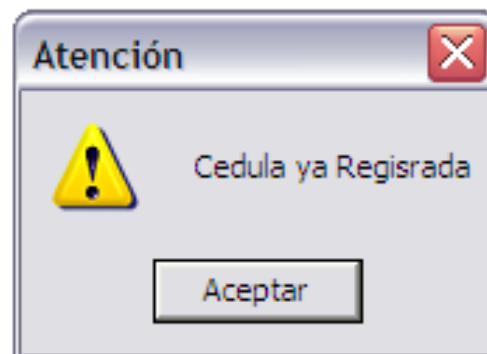
Función MsgBox.

```
Private Sub Form_Load()  
    MsgBox "Mi primer programa"  
End Sub
```



Option Explicit

```
Private Sub Form_Load()  
    Dim Respuesta As Integer  
    Respuesta = MsgBox("Cedula ya Registrada", vbExclamation + vbOKOnly, "Atención")  
    ' Respuesta = MsgBox("¿Deseas Salir del Programa?", vbOKCancel, "SALIR")  
    If Respuesta = 1 Then  
        Print "ACEPTAR"  
    End If  
End Sub
```



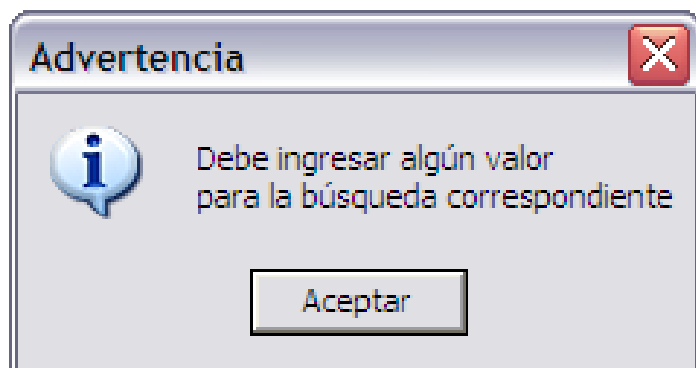
Visual Basic

Función MsgBox.

```
Private Sub Form_Load()  
    MsgBox "¿Quieres Salir?", 5 + 64, "Advertencia"  
End Sub
```

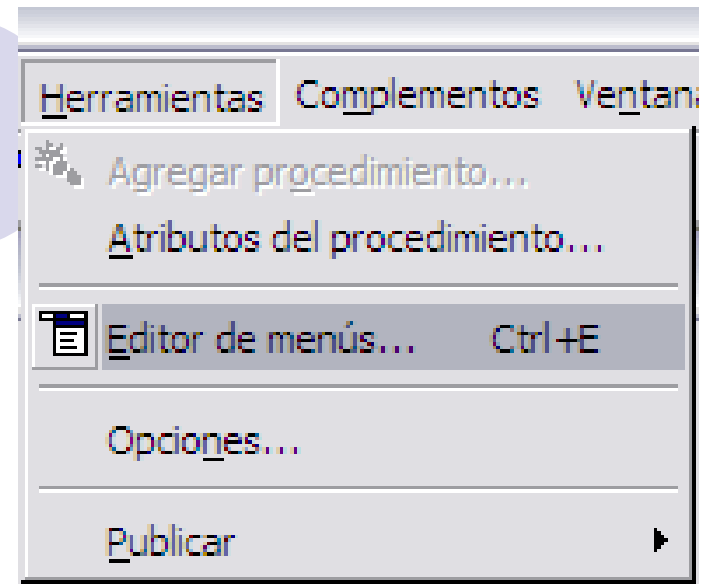
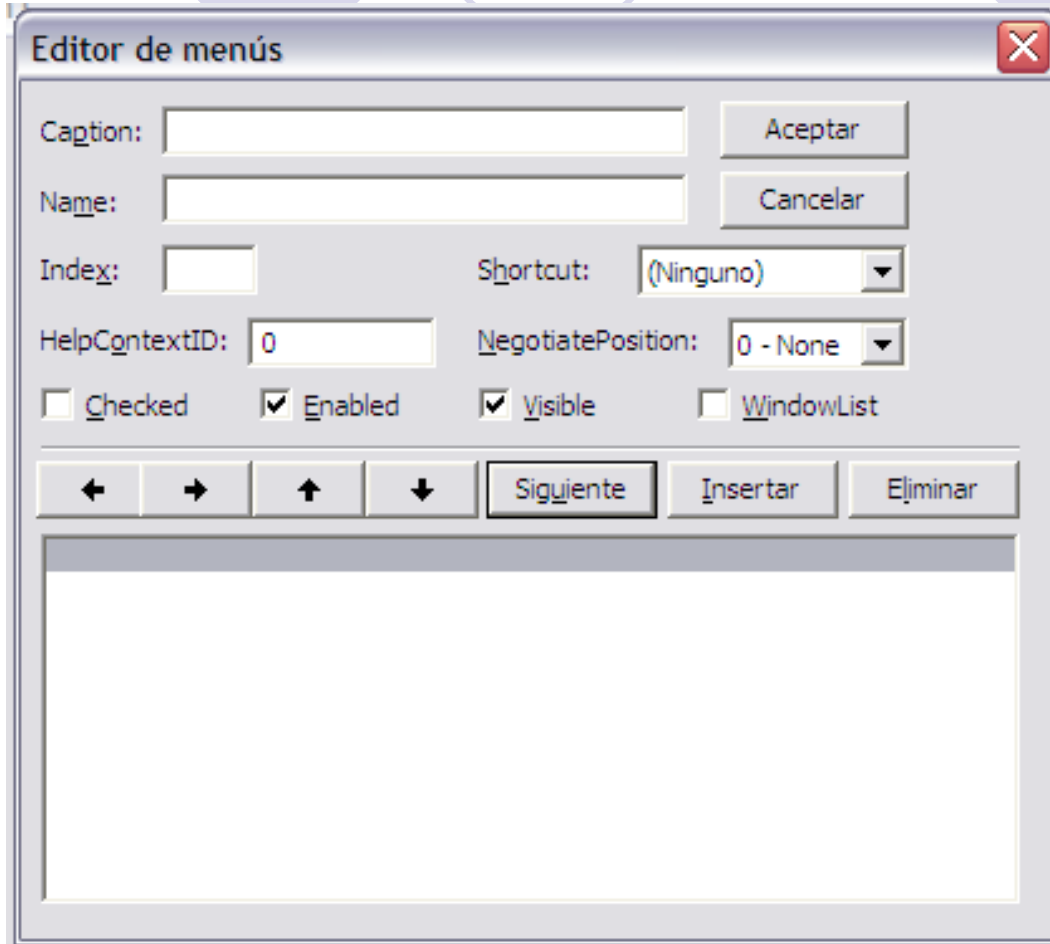


```
Private Sub Form_Load()  
    MsgBox "Debe ingresar algún valor" + Chr(10) + Chr(13) + _  
        "para la búsqueda correspondiente", vbOKOnly + vbInformation, "Advertencia"  
End Sub
```



Visual Basic

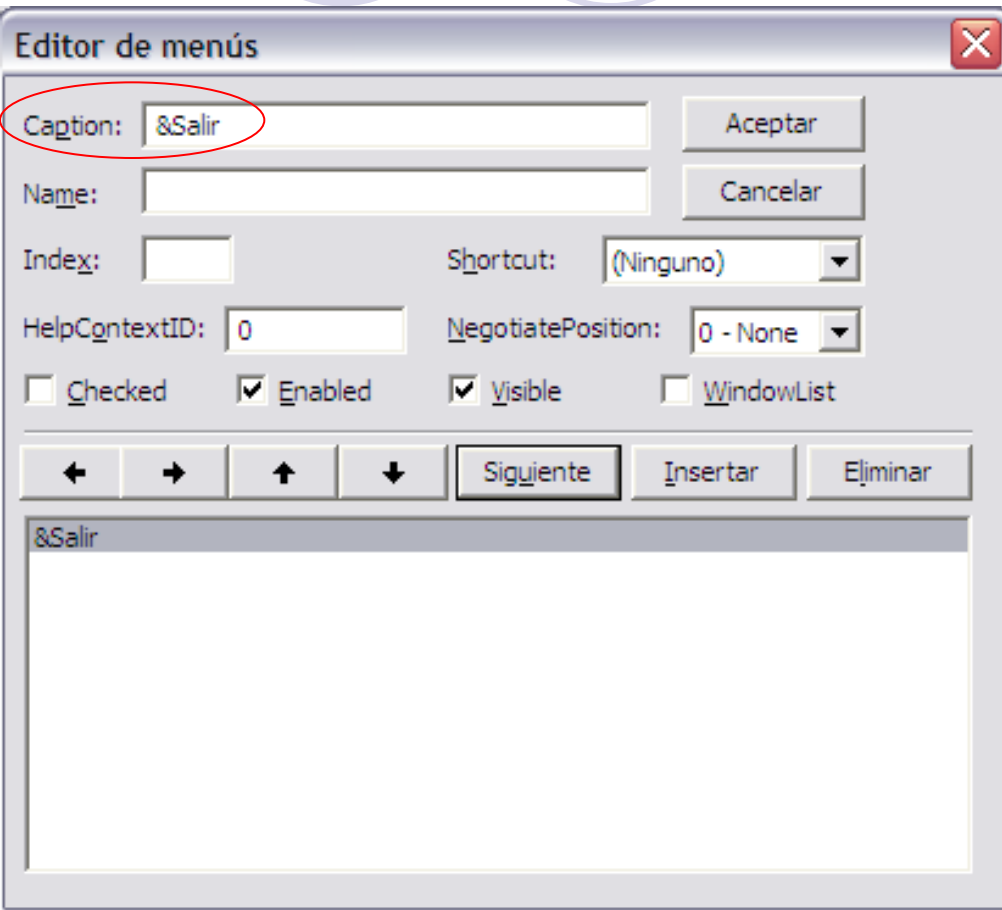
Uso del Menú de Opciones.



- Se puede acceder al **Editor de Menús** de 2 formas: o bien pulsando la combinación de tecla **Ctrl + E**, o bien abriendo el menú **Herramientas** y seleccionando **Editor de menús...**

Visual Basic

Uso del Menú de Opciones.



- A las opciones de menú es posible asignarles una tecla de *Selección Rápida* simplemente precediendo uno de los caracteres del Título del Menú con el signo “&”. Esto causará que dicho carácter aparezca subrayado y se pueda elegir de forma rápida con la combinación **Alt + carácter**.
- Dentro del Editor de Menús el primer dato que tendremos que introducir será el Título del Menú. Esto se hace a través de la propiedad **Caption**.

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Salir

Name: FicheroSalir

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiete Insertar Eliminar

&Salir

- El nombre que utilizaremos para el menú dentro de la codificación, se lo daremos a través de su propiedad **Name**. Esta opción junto con la de **Caption** **son obligatorias**.

Visual Basic

Uso del Menú de Opciones.

- En el caso de que vayamos a crear una Matriz de Opciones, éstas tendrán todas el mismo nombre, y utilizaremos el campo correspondiente a la propiedad **Index** para identificar qué orden de la matriz fue seleccionada.
- En el caso de que deseemos asignar una Tecla de Acceso Rápido a una opción determinada, no tendremos más que desplegar la lista **ShortCut** y elegir una de las que aparecen. Normalmente, NO se puede asignar una Tecla de Acceso Rápido a un nivel superior.
- Los apartados *HelpContextID* (Para crear ayudas para nuestro programa) y *NegotiatePosition* (Para el uso de objetos en nuestro formulario; sobre todo objetos OLE) no los veremos.
- La propiedades **Checked** (*marcado*), **Enabled** (*activado*), **Visible** (*visible*) y *WindowList* (*que no veremos*) pueden tomar valores **True** o **False** según marquemos o no cada opción, estando todos los controles del Menú activados por defecto.

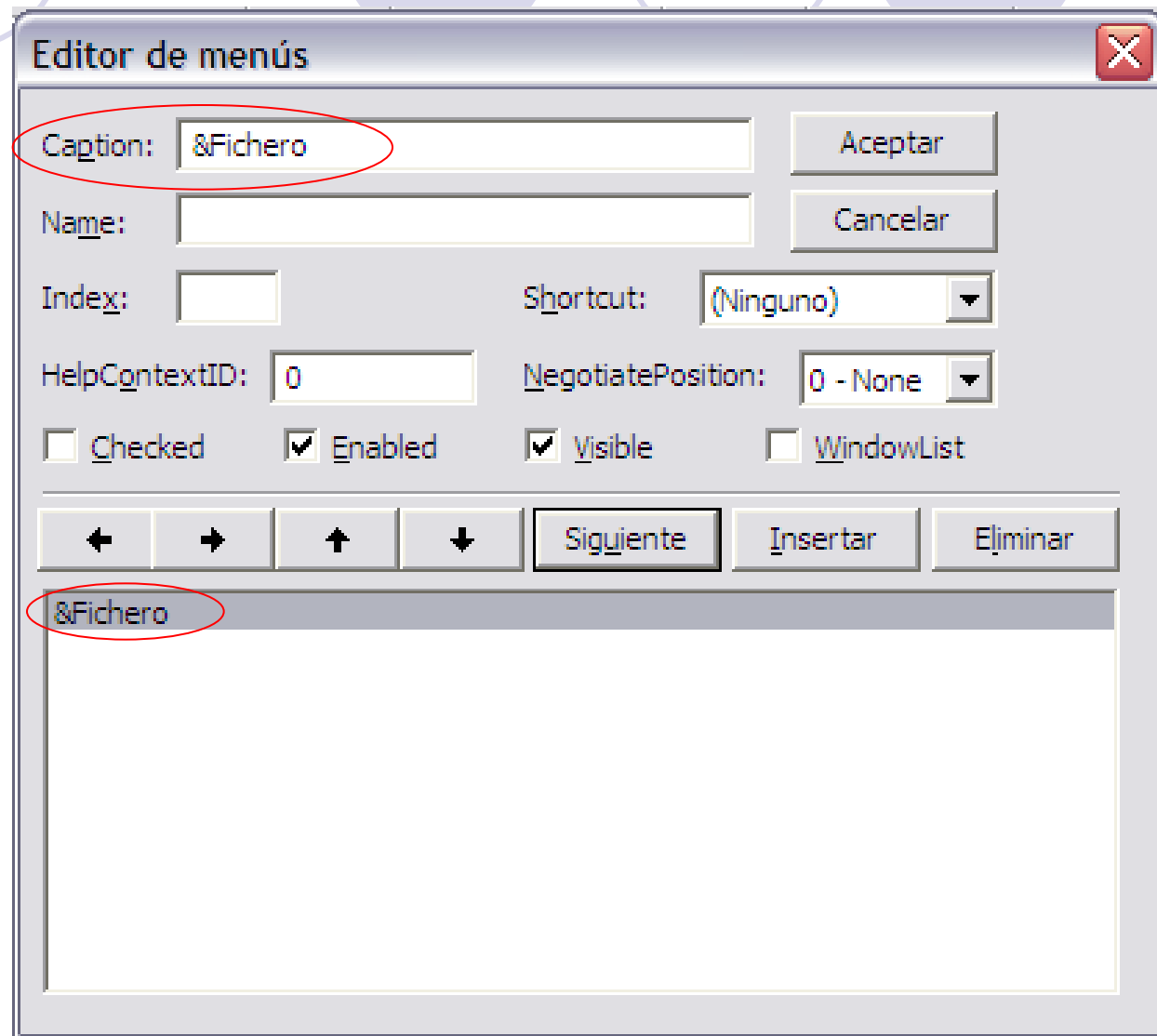
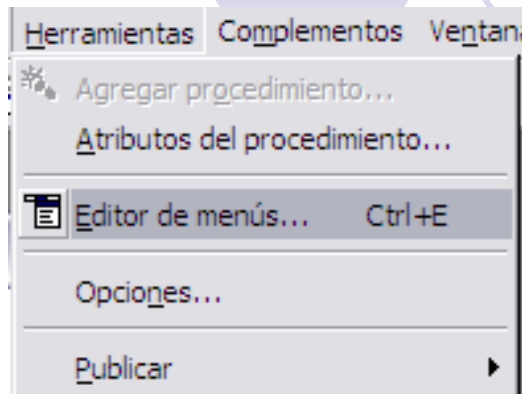
Visual Basic

Uso del Menú de Opciones.

- Cada vez que insertemos los datos de una opción y pulsemos **ENTER** o el botón **Siguiente**, en la parte inferior aparecerá la descripción de la opción.
- El botón **Siguiente** también nos permitirá desplazarnos por la lista de opciones, seleccionando la siguiente opción que exista.
- Los botones ◀ y ▶ nos servirán para seleccionar el nivel de las opciones.
- Los botones ▲ y ▼ nos permitirán desplazar la opción seleccionada a través de toda la lista de opciones.
- El botón **Insertar** nos permitirá abrir un espacio delante de la opción que tengamos seleccionada en ese momento, para poder realizar cualquier otra inserción.
- El botón **Eliminar** elimina la opción seleccionada en ese momento en la lista.
- Para insertar **Líneas de Separación** lo que hay que hacer es colocar en la propiedad **Caption** de la opción de menú seleccionada, un guión ("-") además de un valor en la propiedad **Name**.
- Cuando se termine de diseñar el menú, hay que pulsar la tecla **Aceptar** para poder empezar a trabajar con éste.

Visual Basic

Uso del Menú de Opciones.



Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Fichero

Acceptar

Name: OpciónFichero

Cancelar

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiete Insertar Eliminar

&Fichero

③

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption:

Name:

Index:

Shortcut:

HelpContextID:

NegotiatePosition:

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ **Siguiente** Insertar Eliminar

&Fichero

4

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Abrir

Name: OpciónAbrir

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiete Insertar Eliminar

&Fichero
&Abrir

5

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Abrir Aceptar

Name: OpciónAbrir Cancelar

Index: Shortcut: (Ninguno)

HelpContextID: 0 NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiete Insertar Eliminar

&Fichero
...&Abrir

6

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption:

Name:

Index:

Shortcut: (Ninguno) ▼

HelpContextID:

NegotiatePosition: ▼

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ **Siguiente** Insertar Eliminar

&Fichero
.....&Abrir
.....

7

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Guardar

Name: OpciónGuardar

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓

Siguiete Insertar Eliminar

- &Fichero
 - &Abrir
 - &Guardar

8

Visual Basic

Uso del Menú de Opciones.

Editor de menú

Caption:

Name:

Index:

Shortcut:

HelpContextID:

NegotiatePosition:

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ **Siguiente** Insertar Eliminar

&Fichero
...&Abrir
...&Guardar
...

9

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Mismo Nombre

Name: OpciónMN

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiente Insertar Eliminar

- &Fichero
 -&Abrir
 -&Guardar
 -&Mismo Nombre

10

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Mismo Nombre

Name: OpciónMN

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiente Insertar Eliminar

&Fichero
...&Abrir
...&Guardar
.....&Mismo Nombre

① ①

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: Aceptar

Name: Cancelar

Index: Shortcut: (Ninguno) ▼

HelpContextID: NegotiatePosition: 0 - None ▼

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ **Siguiente** Insertar Eliminar

&Fichero
.....&Abrir
.....&Guardar
.....&Mismo Nombre
.....

① ②

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Otro Nombre

Name: OpciónON

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiente Insertar Eliminar

- &Fichero
 - &Abrir
 - &Guardar
 - &Mismo Nombre
 - &Otro Nombre

① ③

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Otro Nombre

Acceptar

Name: OpciónON

Cancelar

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓

Siguiete Insertar Eliminar

- &Fichero
 - &Abrir
 - &Guardar
 - &Mismo Nombre
 - &Otro Nombre

1

4

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption:

Aceptar

Name:

Cancelar

Index:

Shortcut: (Ninguno) ▼

HelpContextID:

NegotiatePosition: ▼

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ **Siguiente** Insertar Eliminar

&Fichero
.....&Abrir
.....&Guardar
.....&Mismo Nombre
.....&Otro Nombre
.....

①

⑤

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: -

Name: Guión

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓

Siguiente Insertar Eliminar

&Fichero
.....&Abrir
.....&Guardar
.....&Mismo Nombre
.....&Otro Nombre
....._

①

⑥

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: - Aceptar

Name: Guión Cancelar

Index: Shortcut: (Ninguno)

HelpContextID: 0 NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguierte Insertar Eliminar

&Fichero
.....&Abrir
.....&Guardar
.....&Mismo Nombre
.....&Otro Nombre
.....

① ⑦

Visual Basic

Uso del Menú de Opciones.

Editor de menú

Caption:

Name:

Index:

Shortcut:

HelpContextID:

NegotiatePosition:

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ **Siguiente** Insertar Eliminar

&Fichero
....&Abrir
....&Guardar
.....&Mismo Nombre
.....&Otro Nombre
...._
....

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Salir

Name: OpciónSalir

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiete Insertar Eliminar

- &Fichero
 - &Abrir
 - &Guardar
 - &Mismo Nombre
 - &Otro Nombre
 - _
 - &Salir

①

⑨

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Salir

Name: OpciónSalir

Index:

Shortcut: (Ninguno)

HelpContextID: 0

☐ Checked ☒ Enabled ☒ Visible

← → ↑ ↓ Siguiente

&Fichero
...&Abrir
...&Guardar
...&Mismo Nombre
...&Otro Nombre
...
...&Salir

Ctrl+P
Ctrl+Q
Ctrl+R
Ctrl+S
Ctrl+T
Ctrl+U
Ctrl+V
Ctrl+W
Ctrl+X
Ctrl+Y
Ctrl+Z
F1

2 0

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption:

Name:

Index:

Shortcut:

HelpContextID:

NegotiatePosition:

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ **Siguiente** Insertar Eliminar

&Fichero
...&Abrir
...&Guardar
...&Mismo Nombre
...&Otro Nombre
...
...&Salir Ctrl+S
...

② ①

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption:

Aceptar

Name:

Cancelar

Index:

Shortcut: (Ninguno) ▼

HelpContextID:

NegotiatePosition: 0 - None ▼

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

◀ ▶ ↕ ⬇

Siguiente Insertar Eliminar

&Fichero

.....&Abrir

.....&Guardar

.....&Mismo Nombre

.....&Otro Nombre

....._

.....&Salir Ctrl+S

② ②

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Edición

Name: OpciónEdición

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Sigüiente Insertar Eliminar

- &Fichero
 -&Abrir
 -&Guardar
 -&Mismo Nombre
 -&Otro Nombre
 -_
 -&Salir Ctrl+S
- &Edición

② ③

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption:

Name:

Index: Shortcut:

HelpContextID: NegotiatePosition:

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ **Siguiente** Insertar Eliminar

&Fichero
 &Abrir
 &Guardar
 &Mismo Nombre
 &Otro Nombre
 .
 &Salir
&Edición

Ctrl+S

2 4

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Cortar

Name: OpciónCortar

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiente Insertar Eliminar

&Fichero
 &Abrir
 &Guardar
 &Mismo Nombre
 &Otro Nombre
 .
 &Salir
&Edición
&Cortar

Ctrl+S

2

5

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Cortar

Name: OpciónCortar

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiente Insertar Eliminar

&Fichero
...&Abrir
...&Guardar
...&Mismo Nombre
...&Otro Nombre
...
...&Salir Ctrl+S
&Edición
...&Cortar

2

6

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption:

Name:

Index:

Shortcut: (Ninguno) ▼

HelpContextID:

NegotiatePosition: ▼

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ **Siguiente** Insertar Eliminar

&Fichero
 &Abrir
 &Guardar
 &Mismo Nombre
 &Otro Nombre
 .
 &Salir
&Edición
 &Cortar
 .

Ctrl+S

② ⑦

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: Co&piar

Name: OpciónCopiar

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiente Insertar Eliminar

&Fichero
 &Abrir
 &Guardar
 &Mismo Nombre
 &Otro Nombre
 _
 &Salir Ctrl+S
&Edición
 &Cortar
 Co&piar

2

8

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption:

Aceptar

Name:

Cancelar

Index:

Shortcut: (Ninguno) ▼

HelpContextID:

NegotiatePosition: ▼

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ **Siguiente** Insertar Eliminar

&Fichero
.....&Abrir
.....&Guardar
.....&Mismo Nombre
.....&Otro Nombre
.....
.....&Salir
Ctrl+S
&Edición
.....&Cortar
.....Co&piar
.....

② ⑨

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Pegar

Name: OpciónPegar

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☐ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiete Insertar Eliminar

&Fichero
 &Abrir
 &Guardar
 &Mismo Nombre
 &Otro Nombre

 &Salir
&Edición
 &Cortar
 Co&piar
 &Pegar

Ctrl+S

③ ①

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Pegar

Name: OpciónPegar

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☒ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiente Insertar Eliminar

&Fichero
...&Abrir
...&Guardar
...&Mismo Nombre
...&Otro Nombre
...
...&Salir
&Edición
...&Cortar
...Co&piar
...&Pegar

Ctrl+S

③ ①

Visual Basic

Uso del Menú de Opciones.

Editor de menús

Caption: &Pegar

Name: OpciónPegar

Index:

Shortcut: (Ninguno)

HelpContextID: 0

NegotiatePosition: 0 - None

☒ Checked ☒ Enabled ☒ Visible ☐ WindowList

← → ↑ ↓ Siguiente Insertar Eliminar

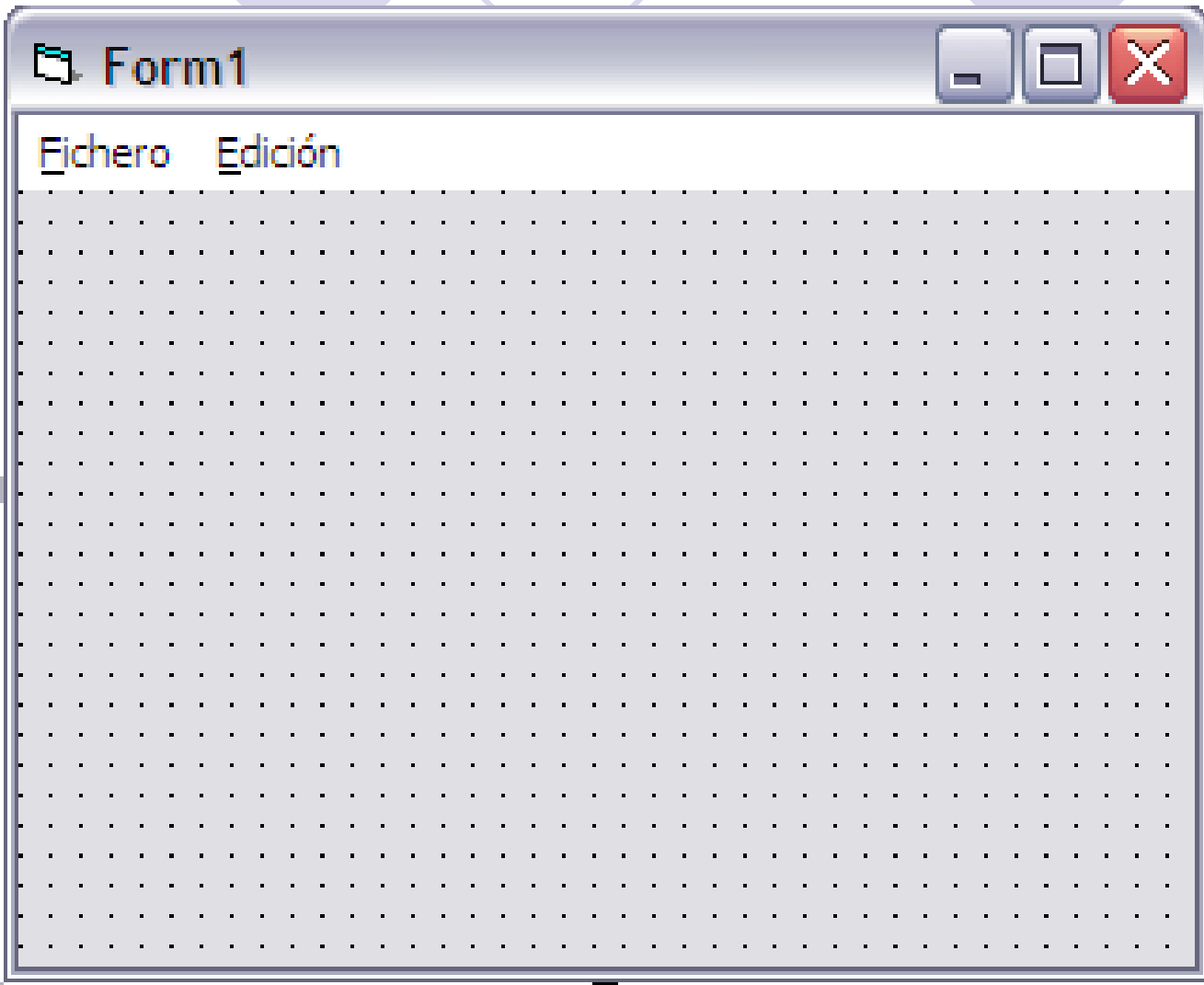
&Fichero
....&Abrir
....&Guardar
.....&Mismo Nombre
.....&Otro Nombre
....
....&Salir
&Edición
....&Cortar
....Co&piar
....&Pegar

Ctrl+S

3 2

Visual Basic

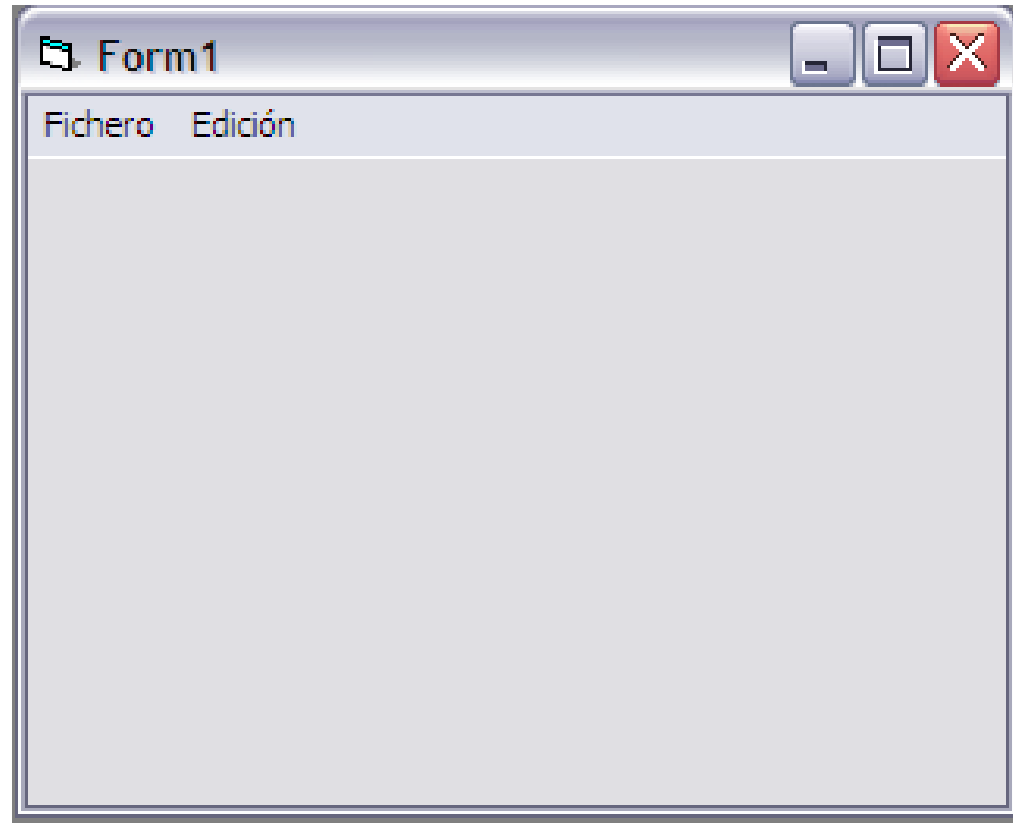
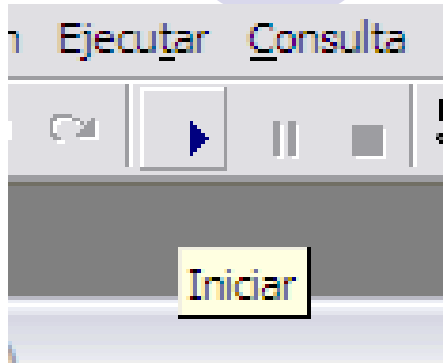
Uso del Menú de Opciones.



③ ③

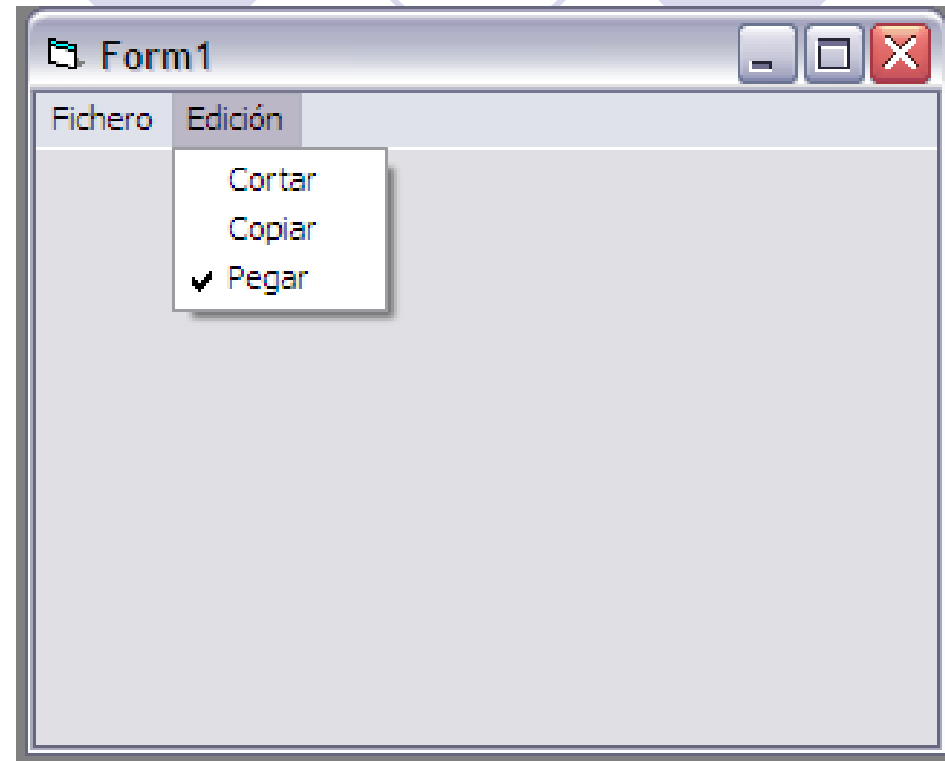
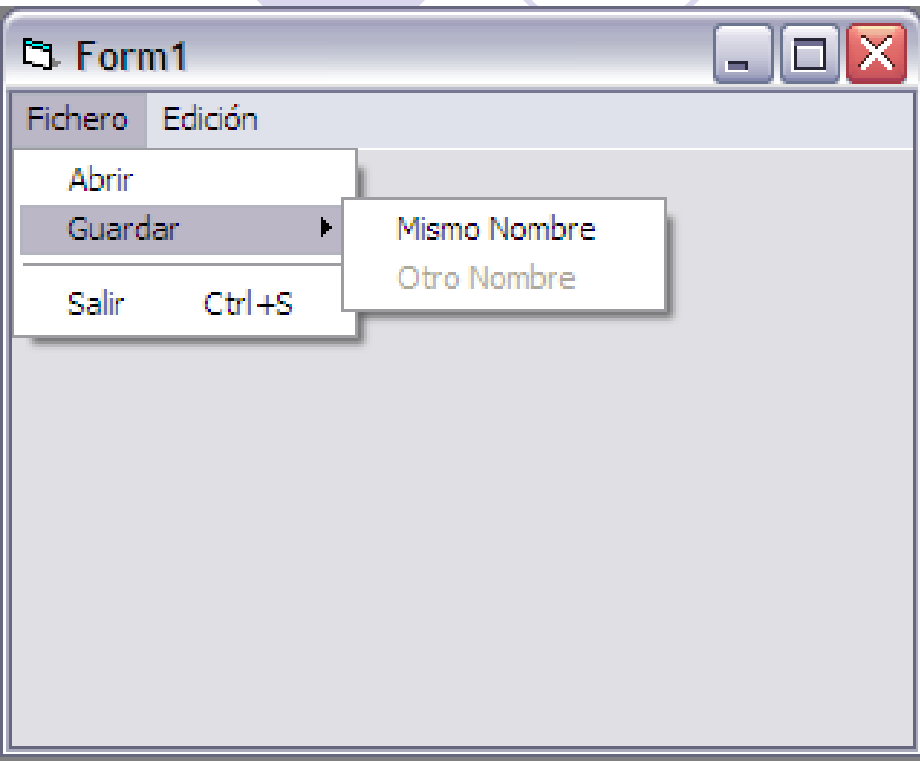
Visual Basic

Uso del Menú de Opciones.



Visual Basic

Uso del Menú de Opciones.



Visual Basic

Uso del Menú de Opciones.

- **Definición de Tipos:**
- Podemos definir tipos compuestos por varios campos, como en Pascal:

```
Private Type Fecha
    Dias As Byte
    Mes As Byte
    Año As Byte
End Type
```

```
Private Type Tipo_Empleado
    Nombre As String * 30
    Direccion As String * 40
    CodPostal As Integer
    FechaNaci As Fecha
    Sueldo As Currency
End Type
```

Visual Basic

Uso del Menú de Opciones.

- **Definición de Tipos:**

```
Dim EmpleadoA As Tipo_Empleado  
Dim EmpleadoB As Tipo_Empleado  
Dim Vector_Empleados(10) As Tipo_Empleado
```

```
EmpleadoA.Nombre = "Alberto Ruiz"  
EmpleadoB.FechaNaci.Dias = 10  
EmpleadoB.FechaNaci.Mes = 3  
EmpleadoB.FechaNaci.Año = 71  
EmpleadoB.Nombre = "Ana López"
```

```
Vector_Empleados(3).Nombre = "Marisa García"
```

Visual Basic

Uso del Menú de Opciones.

- **Definición de Tipos:**

```
Private Type Vector_Enterros  
    Vector(100) As Integer  
End Type
```

```
Dim V1 As Vector_Enterros, V2 As Vector_Enterros
```

Visual Basic

Uso del Menú de Opciones.

- **Definición de Tipos:**
- La definición de un **Tipo** siempre se ha de encontrar a nivel de Módulo; no podemos crear un nuevo **Tipo** en el interior de un procedimiento o función. Por defecto, en los formularios y códigos, los **Tipos** que definamos serán públicos, es decir, aunque los hayamos definido en un formulario, el **Tipo** ya será conocido para el resto. Si deseamos que un **Tipo** se conozca solamente en el interior del módulo donde se ha creado, antepondremos a su definición, como ya hemos visto, la palabra **Private**.
- Podemos utilizar la notación **With**, igual que en Pascal, precediendo los campos de un punto. **With** nos permitirá especificar un **Tipo** de inicio, tras la palabra **With**, de tal forma que a continuación, podremos hacer referencia a todas sus propiedades, simplemente poniendo un punto (.) y el nombre de la propiedad, sin tener que escribir más el nombre del Tipo.

```
With EmpleadoA
    .Nombre = "Josefa López"
    .Direccion = "c/ Francisco Pizarro, 15"
    .CodPostal = 6800
End With
```


Visual Basic

Ejemplo 10.

Menú de Opciones

Estilo Tamaño Tipo Color Salir

Código:

Nombre:

Apellido 1:

Apellido 2:

Dirección: c/ Amoharin Nº 45

Teléfono: 924 313333

C.P.: 06800

Ciudad: Don Benito

Provincia: Badajoz

Visualizar

1	Álvaro	López	López		
	c/ Arquita Nº 23	924-311111	06800	Villafranca	Badajoz
2	Pablo	García	García		

Salir

Visual Basic

Uso del Menú de Opciones.

Caption	Name	Index	Checked	Enable	Visible	Shortcut
&Estilo	MenuEstilo			Si	Si	
....&Negrita	OpcEstilo	0		Si	Si	Ctrl + N
....&Cursiva	OpcEstilo	1		Si	Si	Ctrl + C
....&Subrayada	OpcEstilo	2		Si	Si	Ctrl + S
....&Tachada	OpcEstilo	3		Si	Si	Ctrl + T
&Tamaño	MenuTamaño			Si	Si	
....&Nuevo . . .	OpcNuevo			Si	Si	
....-	OpcTamaño	0		Si	Si	
....10	OpcTamaño	1		Si	Si	
....12	OpcTamaño	2		Si	Si	
T&ipo	MenuTipo			Si	Si	
....&Algerian	OpcTipo	0		Si	Si	
....A&rial	OpcTipo	1		Si	Si	
....&Braggadocio	OpcTipo	2		Si	Si	
....&Courier	OpcTipo	3		Si	Si	
....&Desdemona	OpcTipo	4		Si	Si	
....&Modern	OpcTipo	5		Si	Si	
....M&S Sans Serif	OpcTipo	6		Si	Si	
....&Times New Roman	OpcTipo	7		Si	Si	

Visual Basic

Uso del Menú de Opciones.

Caption	Name	Index	Checked	Enable	Visible	Shortcut
&Color	MenuColor			Si	Si	
....&Tinta	MenuTinta			Si	Si	
.....&Negro	OpcTinta	0		Si	Si	
.....&Azul	OpcTinta	1		Si	Si	
.....Azul &Celeste	OpcTinta	2		Si	Si	
.....&Verde	OpcTinta	3		Si	Si	
.....V&ioleta	OpcTinta	4		Si	Si	
.....&Rojo	OpcTinta	5		Si	Si	
.....&Blanco	OpcTinta	6		Si	Si	
.....A&marillo	OpcTinta	7		Si	Si	
....&Fondo	MenuFondo			Si	Si	
.....&Negro	OpcFondo	0		Si	Si	
.....&Blanco	OpcFondo	1		Si	Si	
&Salir	Salir			Si	Si	

Visual Basic

Uso del Menú de Opciones.

The screenshot shows a Visual Basic application window titled "Menú de Opciones". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Below the title bar is a menu bar with the following items: **E**stilo, **T**amaño, **T**ipo, **C**olor, and **S**alir. The main area of the window is a grid with a dotted background, containing several form controls:

- Form Fields:** On the left side, there are six text boxes labeled "Código:", "Nombre:", "Apellido 1:", "Apellido 2:", "Dirección:", and "Teléfono:". Below these are three more text boxes labeled "C.P.:", "Ciudad:", and "Provincia:". The "Teléfono:" field is split into two parts. To the right of these fields are two large, empty rectangular boxes.
- Elegir Section:** A section titled "Elegir" contains two radio buttons: "Introducir Inf." (which is selected) and "Visualizar Inf.". Below these is a button labeled "Aceptar".
- Opción a Visualizar Section:** A section titled "Opción a Visualizar" contains two checkboxes: "Nombre y Apell." and "Dirección". Below these is a button labeled "Aceptar".
- Visu Section:** A section titled "Visu" contains two radio buttons: "Uno" and "Fin". Below these are two buttons labeled "Aceptar" and "Salir".
- Introduce Valor Section:** A section titled "Introduce Valor" contains a single text box and a button labeled "Salir".
- Visualizar Section:** At the bottom of the window is a section titled "Visualizar" containing a large text area with scrollbars and a button labeled "Salir".

Visual Basic

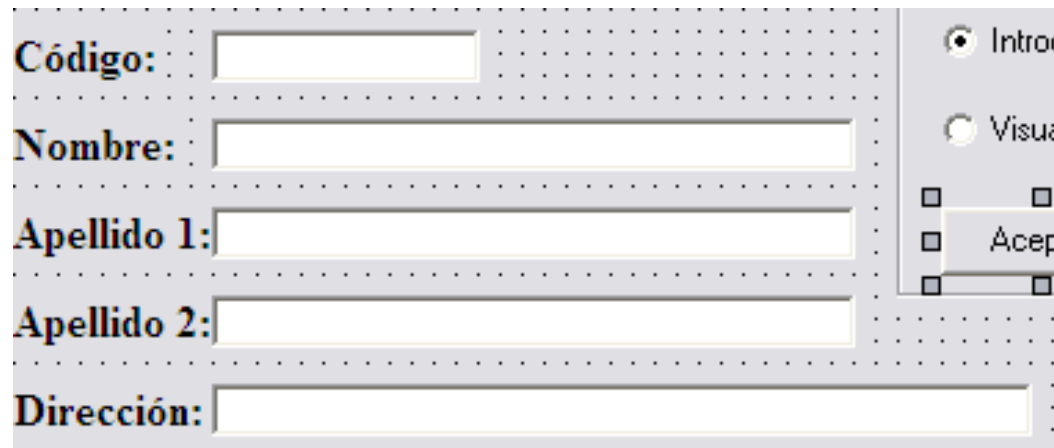
Uso del Menú de Opciones.

Código: .
Nombre: .
Apellido 1: .
Apellido 2: .
Dirección: .
Teléfono: .
C.P.: .
Ciudad: .
Provincia: .

Etiqueta(Label)	Name	Caption	Autosize
Código	LitCódigo	Código:	True
Nombre	LitNombre	Nombre:	True
Apellido 1	LitApellido1	Apellido 1:	True
Apellido 2	LitApellido2	Apellido 2:	True
Dirección	LitDireccion	Dirección:	True
Teléfono	LitTelefono	Teléfono:	True
C.P.	LitCP	C.P.:	True
Ciudad	LitCiudad	Ciudad:	True
Provincia	LitProvincia	Provincia:	True

Visual Basic

Uso del Menú de Opciones.



The image shows a Visual Basic form with a dotted background. It contains five text boxes with the following labels: "Código:", "Nombre:", "Apellido 1:", "Apellido 2:", and "Dirección:". To the right of the form is a portion of the Visual Basic Properties window. It shows the "Intro" radio button selected, the "Visua" radio button, and a "Aceptar" button.

Cuadro de Texto (TextBox)	Name	Alignment	MaxLength	Text
	TexCodigo	2 - Center	5	
	TexNombre	0 – Left Justify	30	
	TexApellido1	0 – Left Justify	30	
	TexApellido2	0 – Left Justify	30	
	TexDireccion	0 – Left Justify	40	

Visual Basic

Uso del Menú de Opciones.

Formulario de entrada de datos:

- Teléfono:
- C.P.:
- Ciudad:
- Provincia:
- Visualizar

Introduce Valor

Cuadro de Texto (TextBox)	Name	Alignment	MaxLength	Text	MultiLine
	TexPrefijo	2 - Center	3		
	TexNumTelefono	2 - Center	6		
	TexCP	2 - Center	5		
	TexCiudad	0 – Left Justify	20		
	TexProvincia	0 – Left Justify	20		
	TexVisualizar	0 – Left Justify	0		True
	TexIValor	0 – Left Justify	2		

Visual Basic

Uso del Menú de Opciones.

Visu

☐ Uno

☐ Fin

Aceptar

Salir

Elegir

☒ Introducir Inf.

☐ Visualizar Inf.

Aceptar

Opción a Visualizar

☐ Nombre y Apell.

☐ Dirección

Aceptar

Introduce Valor

Salir

Visualizar

Salir

Marco (Frame)	Name	Caption
	MarElegir	Elegir
	MarOpVisualizar	Opción a Visualizar
	MarIValor	Introduce Valor
	Visu	Visu
	MarVisualizar	Visualizar

Visual Basic

Uso del Menú de Opciones.

Elegir

☒ Introducir Inf.

☐ Visualizar Inf.

Aceptar

Visu

☒ Uno

☐ Fin

Aceptar

Salir

Opción a Visualizar

☐ Nombre y Apell.

☐ Dirección

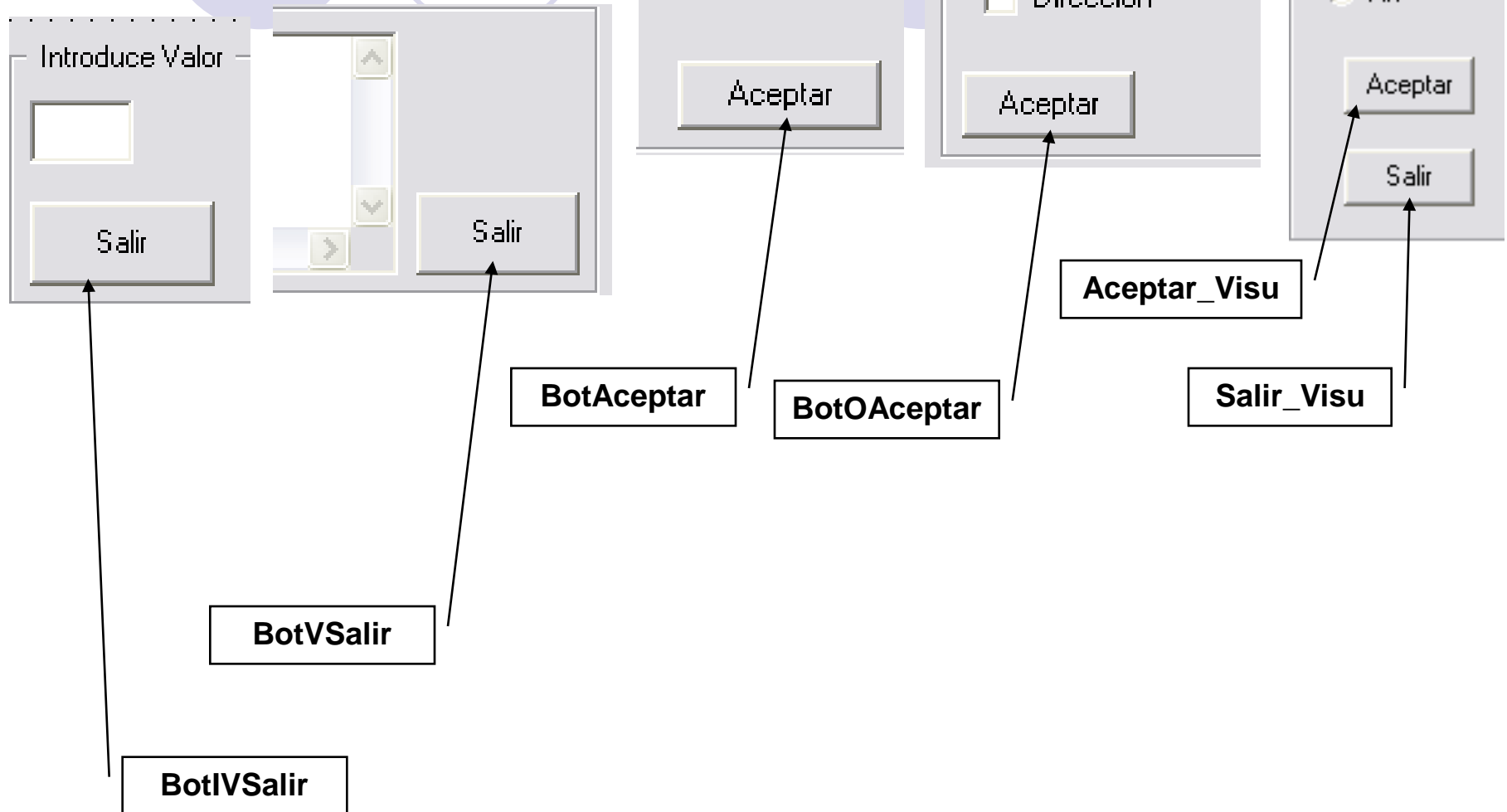
Aceptar

Cuadro de Control (ChekBox)	Name	Caption
	OpNombreApe	Nombre y Apell.
	OpDireccion	Dirección

Botón de Opciones (OptionButton)	Name	Caption	Value
	Uno_Visu	Uno	True
	Fin_Visu	Fin	
	OpIntroducir	Introducir Inf.	True
	OpVisualizar	Visualizar Inf.	

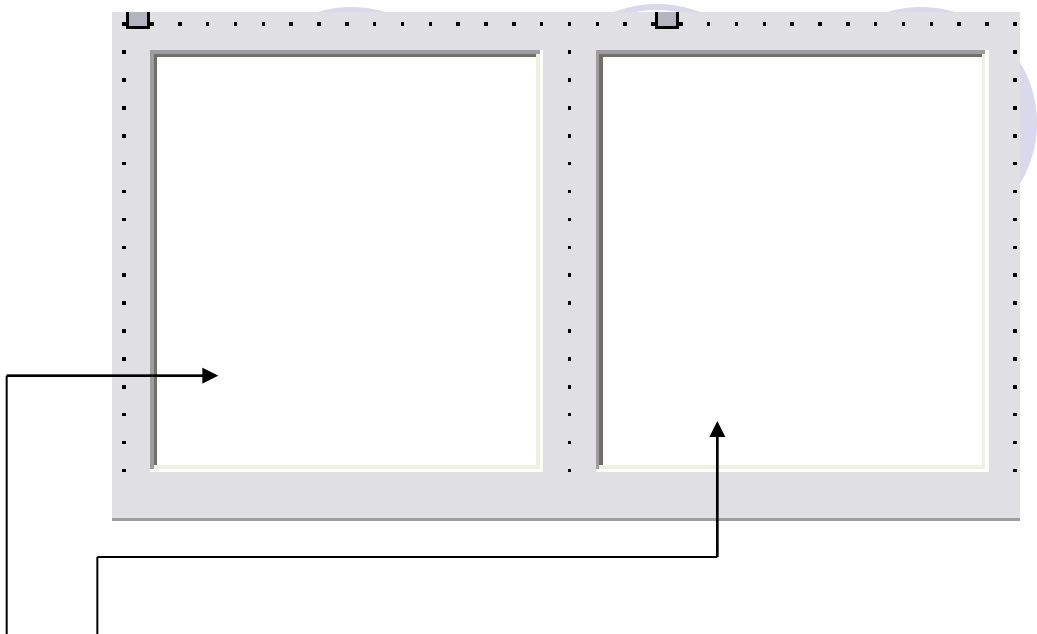
Visual Basic

Uso del Menú de Opciones.



Visual Basic

Uso del Menú de Opciones.



Cuadro de Lista (ListBox)	Name		Columns	List	MultiSelect	Sorted
	LisCiudad		0		0 – None	True
	LisProvincia		0		0 - None	True

```
Private Type NumTelefono
    Prefijo As Integer
    Telefono As Long
End Type
' Definimos una estructura TYPE "NumTelefono" que utilizaremos
' en la estructura TYPE "Datos" de abajo.
Private Type Datos
    Codigo As Long
    Nombre As String * 30
    Apellido1 As String * 30
    Apellido2 As String * 30
    Direccion As String * 40
    NTelefono As NumTelefono      ' "NTelefono" es de la estructura
                                ' TYPE "NumTelefono" de arriba.
    CP As String * 5
    Ciudad As String * 20
    Provincia As String * 20
End Type
Dim V(30) As Datos      ' Definimos un vector de 30 posiciones
                        ' de la estructura TYPE "Datos".

Dim I As Integer

                        ' "I" es el índice que vamos a utilizar en el
                        ' vector "V" para ir introduciendo los elementos (30)
                        ' en dicho vector para, posteriormente, visualizarlos.

Dim Res As Integer

                        ' "Res" es una variable que vamos a utilizar con la
                        ' instrucción "MsgBox".
```

```
Private Sub Form_Load()
```

```
    LisCiudad.AddItem "Zafra"
```

```
    LisCiudad.AddItem "Llerena"
```

```
    LisCiudad.AddItem "Azuaga"
```

```
    LisCiudad.AddItem "Mérida"
```

```
    LisCiudad.AddItem "Don Benito"
```

```
    LisCiudad.AddItem "Almendralejo"
```

```
    LisCiudad.AddItem "Castuera"
```

```
    LisCiudad.AddItem "Montijo"
```

```
    LisCiudad.AddItem "Plasencia"
```

```
    LisCiudad.AddItem "Trujillo"
```

```
    LisCiudad.AddItem "Navalmoral de la Mata"
```

```
    LisCiudad.AddItem "Guadalupe"
```

```
    ' Cargamos la Lista de Ciudades con varias ciudades.
```

```
    LisProvincia.AddItem "Avila"
```

```
    LisProvincia.AddItem "Badajoz"
```

```
    LisProvincia.AddItem "Cáceres"
```

```
    LisProvincia.AddItem "Cordoba"
```

```
    LisProvincia.AddItem "Salamanca"
```

```
    LisProvincia.AddItem "Toledo"
```

```
    LisProvincia.AddItem "Sevilla"
```

```
    LisProvincia.AddItem "Madrid"
```

```
    LisProvincia.AddItem "Barcelona"
```

```
    LisProvincia.AddItem "Zaragoza"
```

```
    LisProvincia.AddItem "Málaga"
```

```
    LisProvincia.AddItem "Cadiz"
```

```
    LisProvincia.AddItem "Granada"
```

```
    ' Cargamos la Lista de Provincias con varias provincias.
```

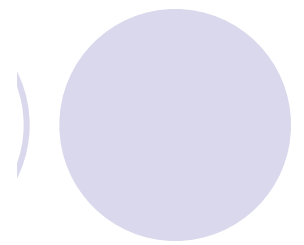
```
LitNombre.Visible = False
LitApellido1.Visible = False
LitApellido2.Visible = False
LitDireccion.Visible = False
LitTelefono.Visible = False
LitCP.Visible = False
LitCiudad.Visible = False
LitProvincia.Visible = False
MarOpVisualizar.Visible = False
LisCiudad.Visible = False
LisProvincia.Visible = False
MarVisualizar.Visible = False
MarElegir.Visible = False
MarIValor.Visible = False
TexNombre.Visible = False
TexApellido1.Visible = False
TexApellido2.Visible = False
TexDireccion.Visible = False
TexPrefijo.Visible = False
TexNumTelefono.Visible = False
TexCP.Visible = False
TexCiudad.Visible = False
TexProvincia.Visible = False
Visu.Visible = False
```

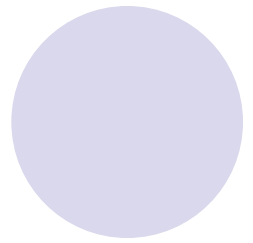
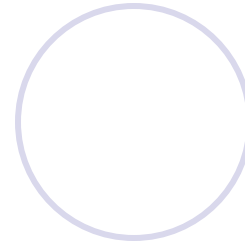
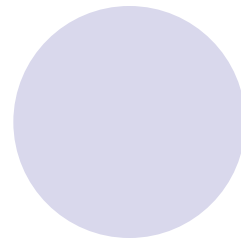
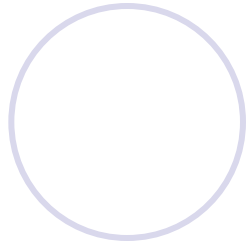
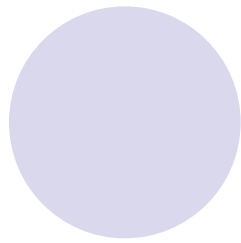
```
    ' Ocultamos todos los campos del formulario menos el de
    ' entrada de Código "TexCodigo".
```

```
I = 0
```

```
    ' Inicializamos el índice "I" del vector "V" a cero.
```

```
End Sub
```





```
' El programa no empezará a funcionar hasta que no entremos en el campo  
' "Código" el valor 99999.  
' Si no lo introducimos aparece un mensaje indicándolo.  
' Si el valor es correcto visualizamos los campos personales.
```

```
Private Sub TexCodigo_KeyPress(KeyAscii As Integer)
```

```
Private Sub TexCodigo_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then
        ' Hemos pulsado ENTER después de introducir la clave.
        If TexCodigo.Text <> "999999" Then
            ' La Clave no coincide con "999999".
            Res = MsgBox("La Clave NO coincide", 16, "¡ERROR!")
            TexCodigo.Text = ""
            TexCodigo.SetFocus
            ' Sacamos el mensaje de error (MsgBox), limpiamos el campo
            ' de la clave y colocamos sobre él el cursor.
        Else
            ' Si hemos acertado con la clave.
            LitNombre.Visible = True
            LitApellido1.Visible = True
            LitApellido2.Visible = True
            LitDireccion.Visible = True
            LitTelefono.Visible = True
            LitCP.Visible = True
            LitCiudad.Visible = True
            LitProvincia.Visible = True
            ' Visualizamos los Literales.
            MarElegir.Visible = True
            ' Visualizamos el Marco "Elegir".
            TexNombre.Visible = True
            TexApellido1.Visible = True
            TexApellido2.Visible = True
            TexDireccion.Visible = True
            TexPrefijo.Visible = True
            TexNumTelefono.Visible = True
            TexCP.Visible = True
            TexCiudad.Visible = True
            TexProvincia.Visible = True
            TexCodigo.Text = ""
            TexCodigo.SetFocus
        End If
    End If
End Sub
```



```

Private Sub BotAceptar_Click()
    ' Comprobamos si el Botón de Control de "Introducir Inf." está marcado.
    If OpIntroducir.Value = True Then
        ' Si está marcado.
        If I <= 30 Then
            ' Comprobamos ahora que el número de elementos del vector "V"
            ' sea menor que 30.
            ' Si es menor o igual a 30 seguimos por aquí.
            If Not (IsNumeric(TexCodigo.Text)) Then
                MsgBox "El Código debe ser numérico"
                TexCodigo.Text = ""
                TexCodigo.SetFocus
                ' Comprobamos que el "Código" sea numérico.
            Else
                V(I).Codigo = CLng(TexCodigo.Text)
                V(I).Nombre = TexNombre.Text
                V(I).Apellido1 = TexApellido1.Text
                V(I).Apellido2 = TexApellido2.Text
                V(I).Direccion = TexDireccion.Text

                If Not (IsNumeric(TexPrefijo.Text)) Then
                    MsgBox "El Prefijo debe ser numérico"
                    TexPrefijo.Text = ""
                    TexPrefijo.SetFocus
                    ' Comprobamos que el "Prefijo" sea numérico.
                Else
                    V(I).NTelefono.Prefijo = CInt(TexPrefijo.Text)

                    If Not (IsNumeric(TexNumTelefono.Text)) Then
                        MsgBox "El Número de Teléfono debe ser numérico"
                        TexNumTelefono.Text = ""
                        TexNumTelefono.SetFocus
                        ' Comprobamos que el "Numero de Teléfono" sea numérico.
                    End If
                End If
            End If
        End If
    End If
End Sub

```

```

If Not (IsNumeric(TexNumTelefono.Text)) Then
    MsgBox "El Número de Teléfono debe ser numérico"
    TexNumTelefono.Text = ""
    TexNumTelefono.SetFocus
    ' Comprobamos que el "Numero de Teléfono" sea numérico.
Else
    V(I).NTelefono.Telefono = CLng(TexNumTelefono.Text)
    V(I).CP = TexCP.Text
    V(I).Ciudad = TexCiudad.Text
    V(I).Provincia = TexProvincia.Text
    I = I + 1
    ' Hemos metido elementos en el vector "V" e incrementamos
    ' el puntero "I".
    TexCodigo.Text = ""
    TexNombre.Text = ""
    TexApellido1.Text = ""
    TexApellido2.Text = ""
    TexDireccion.Text = ""
    TexPrefijo.Text = ""
    TexNumTelefono.Text = ""
    TexCP.Text = ""
    TexCiudad.Text = ""
    TexProvincia.Text = ""
    TexCodigo.SetFocus
    ' Limpiamos todos los campos y nos colocamos de nuevo
    ' en el campo de "Codigo" para introducir el
    ' siguiente elemento.
End If
End If
End If
Else
    Res = MsgBox("Índice superior a 30", 16, "¡ERROR!")

```

```

        End If
    End If
End If
Else
    Res = MsgBox("Índice superior a 30", 16, "¡ERROR!")
    TexCodigo.Text = ""
    TexCodigo.SetFocus
        ' Ya hemos introducido los 30 elementos que caben en el
        ' vector "V".

```

```

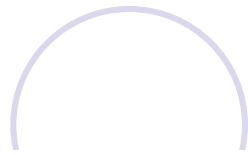
    End If
Else
    TexCodigo.Enabled = False
    TexNombre.Enabled = False
    TexApellido1.Enabled = False
    TexApellido2.Enabled = False
    TexDireccion.Enabled = False
    TexPrefijo.Enabled = False
    TexNumTelefono.Enabled = False
    TexCP.Enabled = False
    TexCiudad.Enabled = False
    TexProvincia.Enabled = False
    TexCodigo.Enabled = False
    MarElegir.Visible = False
    MarOpVisualizar.Visible = True
        ' Como el Botón de Control de "Introducir Inf." NO está marcado, quiere
        ' decir que el Botón de control "Visualizar Inf." ESTÁ marcado.
        ' Por lo tanto, deshabilitamos los campos de entrada de
        ' de información: Código, Nombre, ..., ocultamos el marco "Elegir"
        ' y hacemos visible el marco "Opción a Visualizar".

```

```

    End If
End Sub

```



```
Private Sub BotOAceptar_Click()  
    Dim J As Integer  
        ' Va a ser el indice que vamos a utilizar para visualizar  
        ' la información que hay en el evector "V".  
    Dim Cadena As String  
        ' Variable donde vamos a almacenar temporalmente la cadena a  
        ' visualizar en el Cuadro de Texto "TexVisualizar".  
    Dim Retorno As String  
  
    MarOpVisualizar.Visible = False  
        ' Ocultamos el Marco "Opción a Visualizar".  
    Retorno = Chr(13) & Chr(10)  
        ' Cargamos en "Retorno" en ENTER.  
    MarVisualizar.Visible = True  
        ' Hacemos visible el Marco "Visualizar".
```

cargamos en retorno en pantalla.

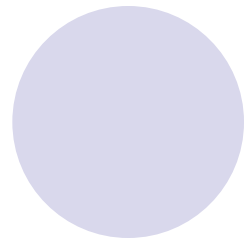
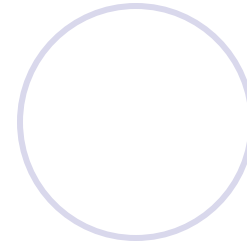
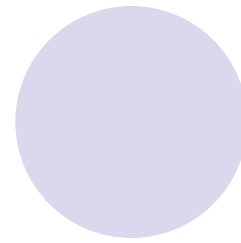
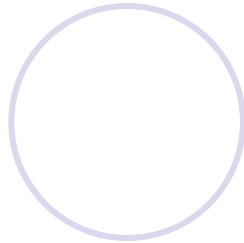
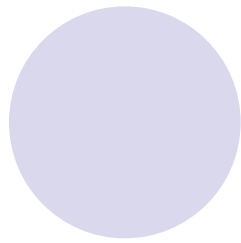
```
MarVisualizar.Visible = True
    ' Hacemos visible el Marco "Visualizar".
For J = 0 To I - 1
    If OpNombreApe.Value = 1 Then
        ' Si la opción "Nombre y Apell." esta marcada, visualizamos los
        ' nombres y los apellidos.
        Cadena = Cadena & V(J).Codigo & " " & V(J).Nombre & _
        " " & V(J).Apellido1 & " " & V(J).Apellido2 & Retorno
    End If
    If OpDireccion.Value = 1 Then
        ' Si la opción "Dirección" est marcada, visualizamos
        ' los nombres y los apellidos (SI están marcados y/o el resto
        ' de campos.
        Cadena = Cadena & V(J).Direccion & " " & V(J).NTelefono.Prefijo & _
        "-" & V(J).NTelefono.Telefono & " " & V(J).CP & " " & _
        V(J).Ciudad & " " & V(J).Provincia & Retorno
    End If
Next J
TexVisualizar.Text = Cadena
    ' Visualizamos la cadena con todos los campos por el Cuadro de
    ' Texto "TexVisualizar".
OpNombreApe.Value = 0
OpDireccion.Value = 0
    ' Desactivamos las 2 opciones del
End Sub
```

```
Private Sub TexCiudad_KeyPress(KeyAscii As Integer)
    ' Si cuando estamos introduciendo el nombre de la "Ciudad", se
    ' pulsa ENTER, se visualza una Lista con todas las Ciudades
    ' que hemos introducido anteriormente con la posibilidad de poder
    ' seleccionar una.
    If KeyAscii = 13 Then
        LisCiudad.Visible = True
    End If
End Sub
```

```
Private Sub LisCiudad_Click()
    ' Seleccionamos una ciudad que pasa al campo "TexCiudad" y luego
    ' hacemos la lista invisible.
    TexCiudad.Text = LisCiudad.Text
    LisCiudad.Visible = False
End Sub
```

```
Private Sub TexProvincia_KeyPress(KeyAscii As Integer)
    ' Si cuando estamos introduciendo el nombre de la "Provinvia", se
    ' pulsa ENTER, se visualza una Lista con todas las Provincias
    ' que hemos introducido anteriormente con la posibilidad de poder
    ' seleccionar una.
    If KeyAscii = 13 Then
        LisProvincia.Visible = True
    End If
End Sub
```

```
Private Sub LisProvincia_Click()
    ' Seleccionamos una provincia que pasa al campo "TexProvincia" y luego
    ' hacemos la lista invisible.
    TexProvincia.Text = LisProvincia.Text
    LisProvincia.Visible = False
End Sub
```



```
Private Sub TexNombre_KeyPress(KeyAscii As Integer)
    ' Si estando situados en el campo de texto del "Nombre",
    ' pulsamos ENTER, se visualiza el marco "Visu".
    If KeyAscii = 13 Then
        Visu.Visible = True
    End If
End Sub
```

```
Private Sub Aceptar_Visu_Click()
```

```
    Dim Z As Integer
```

```
    TexCodigo.Text = ""
```

```
    TexNombre.Text = ""
```

```
    TexApellido1.Text = ""
```

```
    TexApellido2.Text = ""
```

```
    TexDireccion.Text = ""
```

```
    TexPrefijo.Text = ""
```

```
    TexNumTelefono.Text = ""
```

```
    TexCP.Text = ""
```

```
    TexCiudad.Text = ""
```

```
    TexProvincia.Text = ""
```

```
    ' Limpiamos todos los campos indicados arriba.
```

```
    If Uno_Visu.Value = True Then
```

```
        ' Si pulsamos la opción "Uno" se visualizan los campos
```

```
        ' correspondiente al primer valor del vector "V".
```

```
        Z = 0
```

```
    Else
```

```
        ' Si hemos pulsado la otra opción, "Fin", se visualizan los campos
```

```
        ' correspondientes al último valor del vector "V".
```

```
        Z = I - 1
```

```
    End If
```

```
    TexCodigo.Text = V(Z).Codigo
```

```
    TexNombre.Text = V(Z).Nombre
```

```
    TexApellido1.Text = V(Z).Apellido1
```

```
    TexApellido2.Text = V(Z).Apellido2
```

```
    TexDireccion.Text = V(Z).Direccion
```

```
    TexPrefijo.Text = V(Z).NTelefono.Prefijo
```

```
    TexNumTelefono.Text = V(Z).NTelefono.Telefono
```


```
    TexCP.Text = V(Z).CP
```

```
    TexCiudad.Text = V(Z).Ciudad
```


```
    TexProvincia.Text = V(Z).Provincia
```

```
End Sub
```

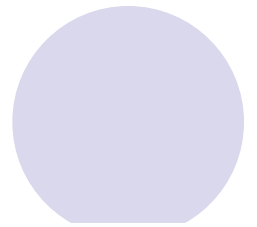
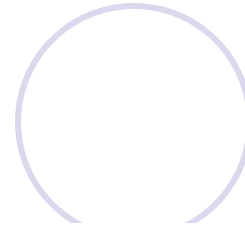
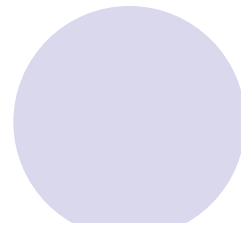
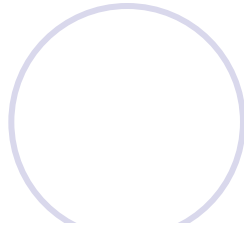
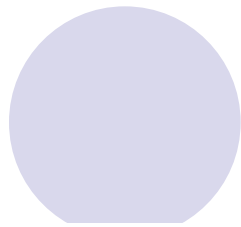




```
Private Sub Salir_Visu_Click()  
    Visu.Visible = False  
        ' Hacemos el marco "Visu" invisible.  
    TexCodigo.Text = ""  
    TexNombre.Text = ""  
    TexApellido1.Text = ""  
    TexApellido2.Text = ""  
    TexDireccion.Text = ""  
    TexPrefijo.Text = ""  
    TexNumTelefono.Text = ""  
    TexCP.Text = ""  
    TexCiudad.Text = ""  
    TexProvincia.Text = ""  
        ' Inicializamos todos los campos de entrada.  
    TexCodigo.SetFocus  
        ' Colocamos el cursor en el campo de "Código".  
End Sub
```



```
Private Sub OpcEstilo_Click(Index As Integer)
    ' Hemos seleccionado la opción "Estilo" del MENÚ.
    ' Podemos seleccionar el "Estilo" (Negrita, Cursiva, Subrayada o Tachada)
    ' a través de su valor "Index" (Name = OpcEstilo).
    ' Una vez que hemos seleccionado una opción, el texto que hay en
    ' "Visualizar" tomará las características de la opción seleccionada.
    Select Case Index
        Case 0
            TexVisualizar.FontBold = Not TexVisualizar.FontBold
            ' El texto que hay en "Visualizar" pasa a Negrita.
        Case 1
            TexVisualizar.FontItalic = Not TexVisualizar.FontItalic
            ' El texto que hay en "Visualizar" pasa a Italic.
        Case 2
            TexVisualizar.FontUnderline = Not TexVisualizar.FontUnderline
            ' El texto que hay en "Visualizar" se subraya.
        Case 3
            TexVisualizar.FontStrikethru = Not TexVisualizar.FontStrikethru
            ' El texto que hay en "Visualizar" se tacha.
    End Select
    OpcEstilo(Index).Checked = Not OpcEstilo(Index).Checked
    ' La opción del MENÚ "Estilo" que esté marcada se "desmarca", y la
    ' que esté "desmarcada" queda "marcada".
End Sub
```

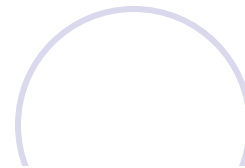
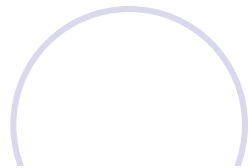


```
Private Sub OpcTamaño_Click(Index As Integer)
    ' Hemos seleccionado la opción "Tamaño" del MENÚ.
    TexVisualizar.FontSize = OpcTamaño(Index).Caption
    ' El texto que hay en "Visualizar" toma los tamaños: 10 o 12, que
    ' estan indicados en la propiedad "Caption" de cada uno de estos valores.
    ' Cada valor se obtiene a partir de su valor "Index"
    ' Para coger otro tamaño distinto, lo podemos hacer a traves de su
    ' opción "Nuevo . . .".
End Sub
```

```
Private Sub OpcNuevo_Click()
    MarIValor.Visible = True
    ' Visualizamos el Marco "MarIValor".
    TexIValor.SetFocus
    ' Nos situamos en el campo de texto "TexIValor".
End Sub
```

```
Private Sub BotIVSalir_Click()  
    Static NumTamaño As Byte  
  
    MarIValor.Visible = False  
        ' Ocultamos el Marco "MarIValor".  
    TexVisualizar.FontSize = CInt(TexIValor.Text)  
        ' El texto que hay en el Marco "MarVisualizar" - "TexVisuaizar"  
        ' toma el tamaño NUEVO que hemos indicado en "TexIValor".  
    Load OpcTamaño(NumTamaño + 3)  
        ' Siempre que tengamos una matriz de controles (OpcTamaño) en  
        ' nuestro formulario, durante la ejecución de nuestro programa  
        ' podemos crear "elementos" adicionales. Esto se hace con la  
        ' sentencia "Load" seguida del nombre del control (OpcTamaño)  
        ' y entre paréntesis el índice del nuevo elemento, que como es  
        ' lógico, no puede ya estar asignado a otro control de la misma  
        ' matriz.  
    OpcTamaño(NumTamaño + 3).Caption = TexIValor.Text  
        ' La nueva opción que aparecerá en el MENÚ "Tamaño" tendrá  
        ' como "presencia" el valor pasado a su propiedad "Caption".  
    OpcTamaño(NumTamaño + 3).Visible = True  
        ' Hacemos "Visible" la nueva opción de "Tamaño".  
    NumTamaño = NumTamaño + 1  
        ' Incremenetamos el valor del "Índice" para futuras nuevas opciones.  
    TexIValor.Text = ""  
        ' Limpiamos el campo de texto de "TexIValor".  
  
End Sub
```

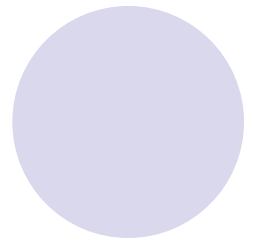
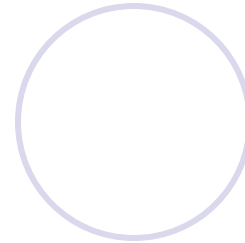
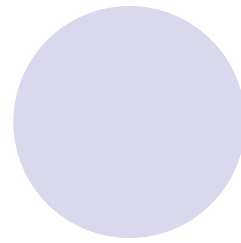
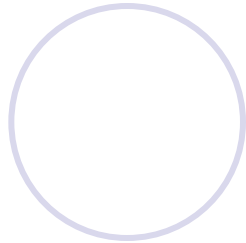
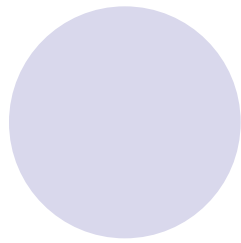
```
Private Sub OpcTipo_Click(Index As Integer)
    ' Hemos seleccionado la opción "Tipo" del MENÚ.
    Select Case Index
        ' Con el tipo de letra seleccionado (Index) cambiamos
        ' el tipo de letra de "TexVisualizar".
    Case 0
        TexVisualizar.FontName = "Arial"
    Case 1
        TexVisualizar.FontName = "Arial Black"
    Case 2
        TexVisualizar.FontName = "Century"
    Case 3
        TexVisualizar.FontName = "MS Sans Serif"
    Case 4
        TexVisualizar.FontName = "Tahoma"
    Case 5
        TexVisualizar.FontName = "MS Sans Serif"
    Case 6
        TexVisualizar.FontName = "Times New Roman"
    Case 7
        TexVisualizar.FontName = "Verdana"
    End Select
End Sub
```



```
Private Sub OpcTinta_Click(Index As Integer)
    ' De la opción de "Color" del MENÚ hemos seleccionado la opción "Tinta".
    Select Case Index
        ' Con el color de letra seleccionado (Index) cambiamos
        ' el color de letra de "TexVisualizar".
    Case 0
        TexVisualizar.ForeColor = vbBlack
    Case 1
        TexVisualizar.ForeColor = vbBlue
    Case 2
        TexVisualizar.ForeColor = vbCyan
    Case 3
        TexVisualizar.ForeColor = vbGreen
    Case 4
        TexVisualizar.ForeColor = vbMagenta
    Case 5
        TexVisualizar.ForeColor = vbRed
    Case 6
        TexVisualizar.ForeColor = vbWhite
    Case 7
        TexVisualizar.ForeColor = vbYellow
    End Select
End Sub
```

```
Private Sub OpcFondo_Click(Index As Integer)
    ' De la opción de "Color" del MENÚ hemos seleccionado la opción "Fondo".
    If Index Then
        ' Hemos seleccionado el fondo "Blanco". El "Negro" se desactiva.
        OpcFondo(0).Checked = False
        OpcFondo(1).Checked = True
        TexVisualizar.BackColor = vbWhite
    Else
        ' Hemos seleccionado el fondo "Negro". El "Blanco" se desactiva.
        OpcFondo(0).Checked = True
        OpcFondo(1).Checked = False
        TexVisualizar.BackColor = vbBlack
    End If
End Sub
```

```
Private Sub BotVSalir_Click()
    ' Hemos seleccionado el botón "Salir" del Marco "MarVisualizar"
    MarVisualizar.Visible = False
    MarIValor.Visible = False
    MarElegir.Visible = True
    TexCodigo.Enabled = True
    TexNombre.Enabled = True
    TexApellido1.Enabled = True
    TexApellido2.Enabled = True
    TexDireccion.Enabled = True
    TexPrefijo.Enabled = True
    TexNumTelefono.Enabled = True
    TexCP.Enabled = True
    TexCiudad.Enabled = True
    TexProvincia.Enabled = True
    TexCodigo.Enabled = True
    OpIntroducir.Value = True
    TexCodigo.SetFocus
End Sub
```

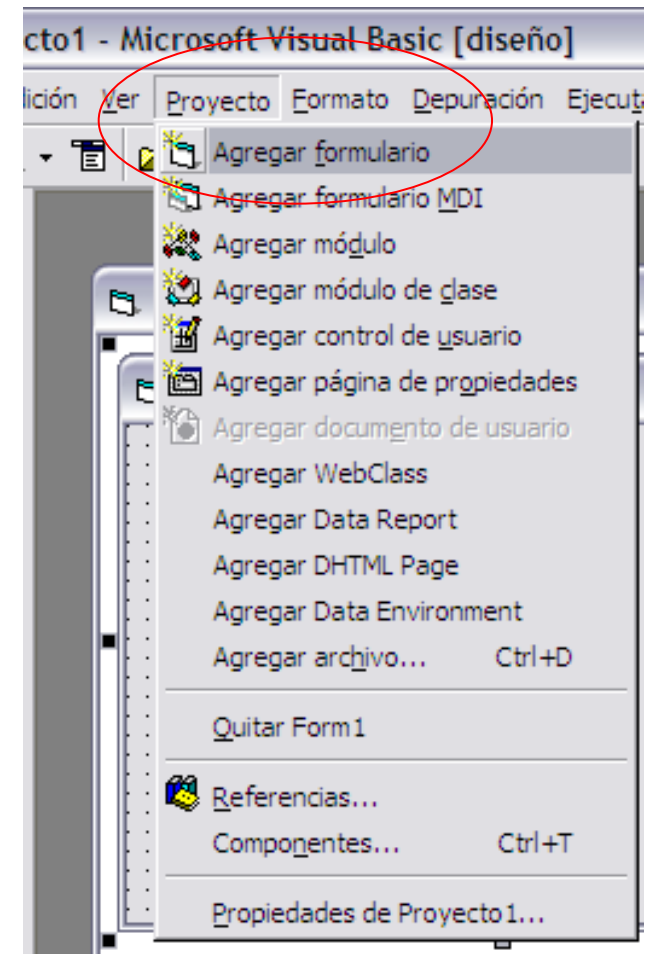
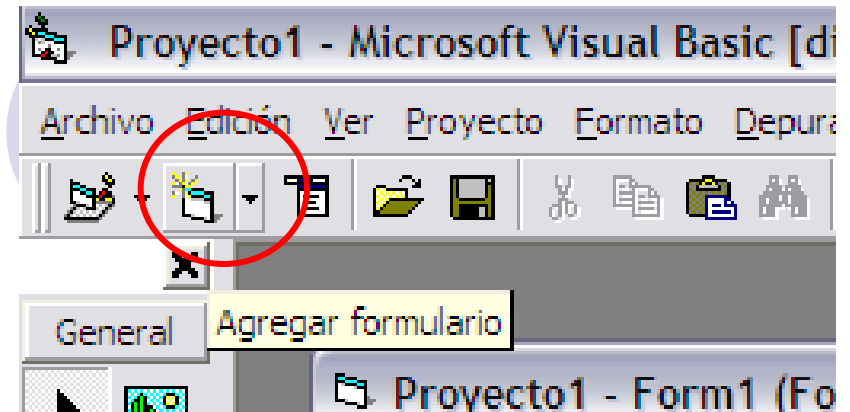


```
Private Sub Salir_Click()  
    ' Hemos seleccionado la opción "Salir" del MENÚ.  
End  
End Sub
```


Visual Basic

Múltiple Formularios.

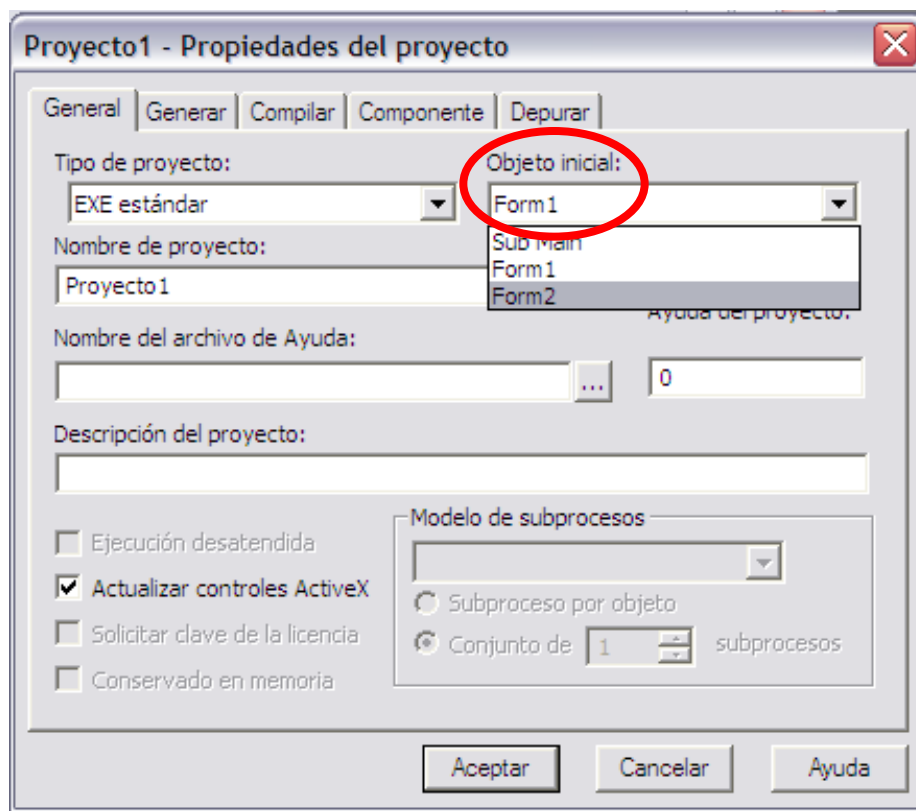
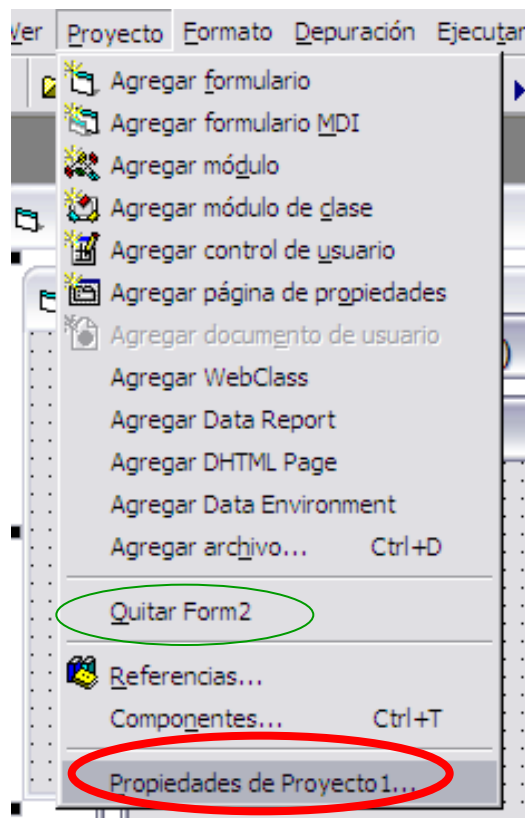
- Para añadir un nuevo formulario a nuestra aplicación bastará con pulsar el botón **Formulario**:
- También lo podemos hacer abriendo el menú **Proyecto** y seleccionando la opción "**Agregar Formulario**":



Visual Basic

Múltiple Formularios.

- Si después de añadir varios formularios al formulario inicial, ejecutamos la aplicación, aparecerá el formulario principal. Esto es porque el formulario inicial fue el primero que insertamos en nuestro proyecto. En el caso de que deseemos que al ejecutar el programa el formulario que aparezca en primer lugar sea otro, lo único que tenemos que hacer es seleccionar la opción “**General**” de la opción “**Propiedades de proyecto**” del menú *Proyecto*.



Visual Basic

Múltiple Formularios.

- En la opción **General** de la opción Propiedades de **proyecto** hay un apartado llamado **Objeto inicial** con una lista en la que podemos seleccionar el formulario inicial con el que queremos que se empiece a ejecutar la aplicación.
- Desde el formulario o procedimiento de inicio es posible cargar resto de formularios (“ventanas”) de la aplicación.
- Para cargar un formulario y visualizarlo de forma inmediata, lo podemos hacer utilizando el método **Show**.

Nombre_Formulario.**Show**

- El formulario, en este caso, se mostraría como **NO modal**; por lo que permitiría seccionar cualquier otra ventana de la aplicación. Si deseamos que el formulario se presente como ventana **modal**, impidiendo el cambio a otras ventanas en tanto ésta no se cierre, tan sólo tenemos que facilitar el parámetro **1** a **Show**.

Nombre_Formulario.**Show 1**

- En el caso de que antes de visualizar el formulario deseemos realizar alguna operación previa con éste –como el establecimiento de propiedades- debemos cargarlo por medio de la sentencia Load.

Nombre_Formulario.**Load**

Visual Basic

Múltiple Formularios.

- A partir de este momento, podemos acceder a todas sus propiedades. Cuando deseemos visualizarlo podemos utilizar el método **Show**, citado anteriormente, o simplemente dar el valor **True** a la propiedad **Visible**.
- Indistintamente del método que hayamos utilizado para cargar o visualizar un formulario, cuando éste ya no nos sea útil, podemos abandonarlo liberando la memoria que ocupa. Para ello utilizaremos la sentencia **Unload**.

Unload Nombre_Formulario

- En el caso de que esta operación se realice desde un procedimiento del mismo formulario, podemos utilizar la palabra **Me** en lugar del nombre del formulario.

Unload Me

- **Me** representa al objeto desde el que se ejecuta el código.
- Si en un formulario hemos insertado una serie de controles y queremos acceder a uno de ellos, utilizamos su nombre seguido de un punto y de la propiedad. Esto lo podemos hacer desde un procedimiento que pertenezca al mismo formulario que contiene los controles, pero si desde este procedimiento lo que queremos es acceder a los controles de otro formulario distinto, tendremos que facilitar además, el nombre del formulario.
- Para ello se utiliza el operador **!**, cuya finalidad es referenciar a un objeto dentro de otro (¡OBJETO!, no propiedad).

Nombre_Formulario!Nombre_Objeto.Propiedad

Visual Basic

Ejemplo 11.

A Visual Basic form titled "TITULO" with a standard Windows-style title bar (minimize, maximize, close buttons). The form contains a vertical stack of five buttons on the left side: "Autor", "Editorial", "Título", "ISBN", and "Salir". The "Autor" button is highlighted with a dotted border. To the right of these buttons is a 2x2 grid of text boxes. The top-left text box is labeled "AUTOR" below it. The top-right text box is labeled "EDITORIAL" below it. The bottom-left text box is labeled "TITULO" below it. The bottom-right text box is labeled "ISBN" below it. All text boxes are currently empty.

Visual BasicEjemplo 11.

Objeto	Name	Caption	Index
Formulario (Form)	VentanaPrincipal	TITULO	
Botón de Comando (CommandButton)	Busqueda (Busqueda(0))	&Autor	0
Botón de Comando (CommandButton)	Busqueda (Busqueda(1))	&Editorial	1
Botón de Comando (CommandButton)	Busqueda (Busqueda(2))	&Título	2
Botón de Comando (CommandButton)	Busqueda (Busqueda(3))	&ISBN	3
Botón de Comando (CommandButton)	Busqueda (Busqueda(4))	&Salir	4
Cuadro de Lista (ListBox)	AutorLista		
Cuadro de Lista (ListBox)	EditorialLista		
Cuadro de Lista (ListBox)	TituloLista		
Cuadro de Lista (ListBox)	ISBNLista		
Etiqueta (Label)	Label1	AUTOR	
Etiqueta (Label)	Label2	EDITORIAL	
Etiqueta (Label)	Label3	TITULO	
Etiqueta (Label)	Label4	ISBN	

Visual Basic

Ejemplo 11.

The screenshot shows a Visual Basic form titled "TITULO". The form is designed with a grid layout. On the left side, there is a vertical column of five buttons: "Autor", "Editorial", "Ítulo", "ISBN", and "Salir". To the right of these buttons, the form is divided into four large rectangular areas arranged in a 2x2 grid. The top-left area is labeled "AUTOR", the top-right area is labeled "EDITORIAL", the bottom-left area is labeled "TITULO", and the bottom-right area is labeled "ISBN". Each of these four areas is currently empty, suggesting they are intended for user input or display of data.

The screenshot displays the Visual Basic IDE interface. On the right side, the "Proyecto - Project1" window shows the project structure. The file "VentanaPrincipal (Vp.frm)" is selected and highlighted with a red circle. Below this, the "Propiedades - VentanaPrincipal" window shows the properties for the selected form. The "Caption" property is set to "TITULO".

(Nombre)	VentanaPrincipal
Appearance	1 - 3D
AutoRedraw	False
BackColor	<input type="checkbox"/> &H8000000F&
BorderStyle	1 - Fixed Single
Caption	TITULO
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid
DrawWidth	1
Enabled	True
FillColor	<input checked="" type="checkbox"/> &H00000000&
FillStyle	1 - Transparent

```
Private Sub Busqueda_Click(Index As Integer)
    ' Comprobamos cual ha sido el BOTON que hemos pulsado
    ' de la matriz de botones que hemos creado.
    ' Nuestra matriz de botones se llama Búsqueda".
    Select Case Index
        Case 0
            PorAutor.Show 1
            ' Visualizamos el Formulario de "Autor" de forma "modal".
        Case 1
            PorEditorial.Show 1
            ' Visualizamos el Formulario de "Editorial" de forma "modal".
        Case 2
            PorTitulo.Show 1
            ' Visualizamos el Formulario de "Titulo" de forma "modal".
        Case 3
            PorISBN.Show 1
            ' Visualizamos el Formulario de "ISBN" de forma "modal".
        Case 4
            Unload Me
            ' "Descargamos" el formulario principal "VentanaPrincipal".
        End
        ' Salimos.
    End Select
End Sub
```


Visual Basic

Ejemplo 11.

TITULO

Autor

Editorial

Titulo

ISBN

Salir

AUTOR

EDITORIAL

TITULO

ISBN

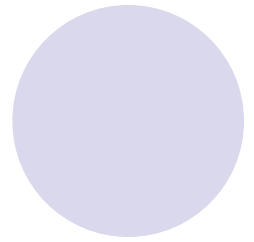
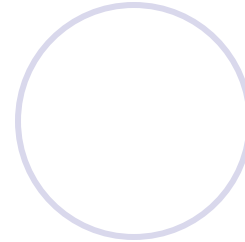
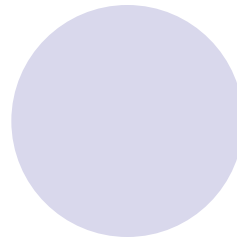
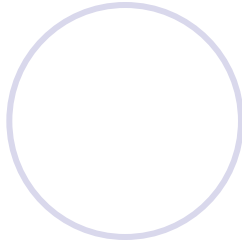
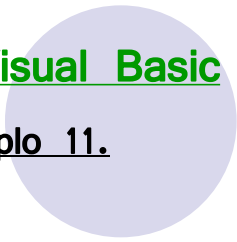
Búsqueda por Autor

Introduzca el Nombre del Autor a buscar:

Salir

Visual Basic

Ejemplo 11.



Objeto	Name	Caption	BorderStyle
Formulario (Form)	PorAutor	Búsqueda por Autor	3
Botón de Comando (CommandButton)	SalirAutor	&Salir	
Cuadro de Texto (TextBox)	NombreAutor		
Etiqueta (Label)	Label1	Introduzca el Nombre del Autor a buscar:	



Option Explicit

```
Private Sub NombreAutor_KeyPress(KeyAscii As Integer)
    ' Vamos introduciendo los caracteres del Nombre del Autor
    ' hasta que pulsemos ENTER. En ese momento, el autor introducido
    ' pasa a la lista de autores del formulario principal (VentanaPrincipal).
    If KeyAscii = 13 Then
        Load VentanaPrincipal
        VentanaPrincipal!AutorLista.AddItem NombreAutor.Text
        NombreAutor.Text = " "
    End If
End Sub
```

```
Private Sub SalirAutor_Click()
    ' Cuando pulsamos el botón "Salir" se "descarga" este formulario
    ' y se "carga" el de la ventana principal (VentanaPrincipal).
    Unload Me
    VentanaPrincipal.Show
End Sub
```

Visual Basic

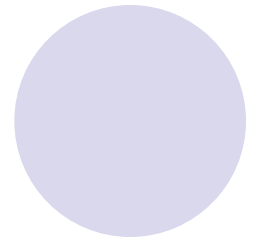
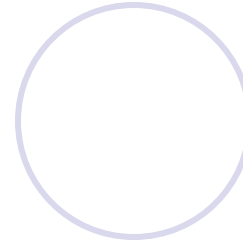
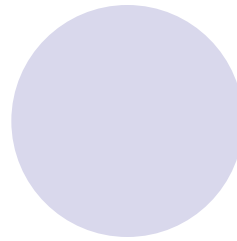
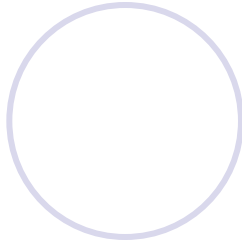
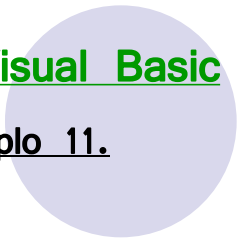
Ejemplo 11.

The screenshot shows a Visual Basic form titled "TITULO". On the left side, there are five buttons: "Autor", "Editorial", "Título", "ISBN", and "Salir". The "Editorial" button is circled in red. To the right of these buttons are four text boxes arranged in a 2x2 grid. The top-left box is labeled "AUTOR", the top-right box is labeled "EDITORIAL", the bottom-left box is labeled "TITULO", and the bottom-right box is labeled "ISBN".


The screenshot shows a Visual Basic form titled "Búsqueda por Editorial". It features a single text box with the label "Entrar la Editorial a buscar:" above it. At the bottom right of the form is a button labeled "Salir". A red arrow points from the "Editorial" button in the first screenshot to this text box.

Visual Basic

Ejemplo 11.



Objeto	Name	Caption	BorderStyle
Formulario (Form)	PorEditorial	Búsqueda por Editorial	3
Botón de Comando (CommandButton)	SalirEditorial	&Salir	
Cuadro de Texto (TextBox)	Editorial		
Etiqueta (Label)	Label1	Entrar la Editorial a buscar:	



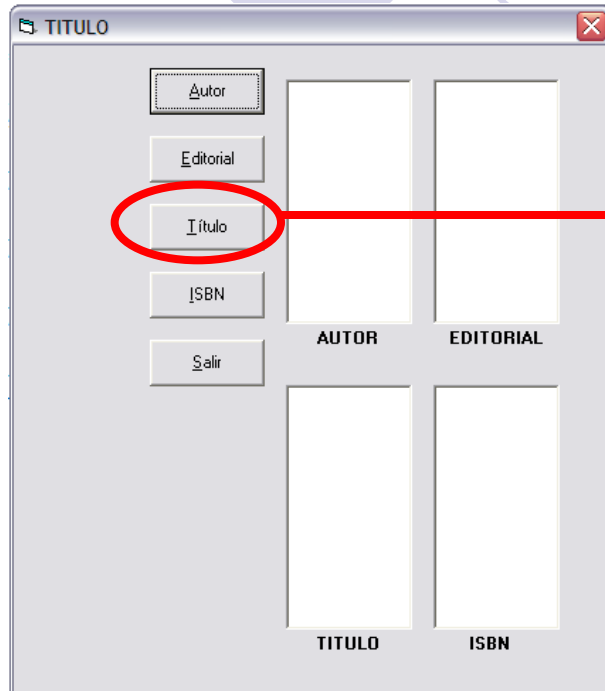
Option Explicit

```
Private Sub Editorial_KeyPress(KeyAscii As Integer)
    ' Vamos introduciendo los caracteres de la Editorial
    ' hasta que pulsemos ENTER. En ese momento, la editorial introducida
    ' pasa a la lista de editoriales del formulario principal (VentanaPrincipal).
    If KeyAscii = 13 Then
        Load VentanaPrincipal
        VentanaPrincipal!EditorialLista.AddItem Editorial.Text
        Editorial.Text = " "
    End If
End Sub
```

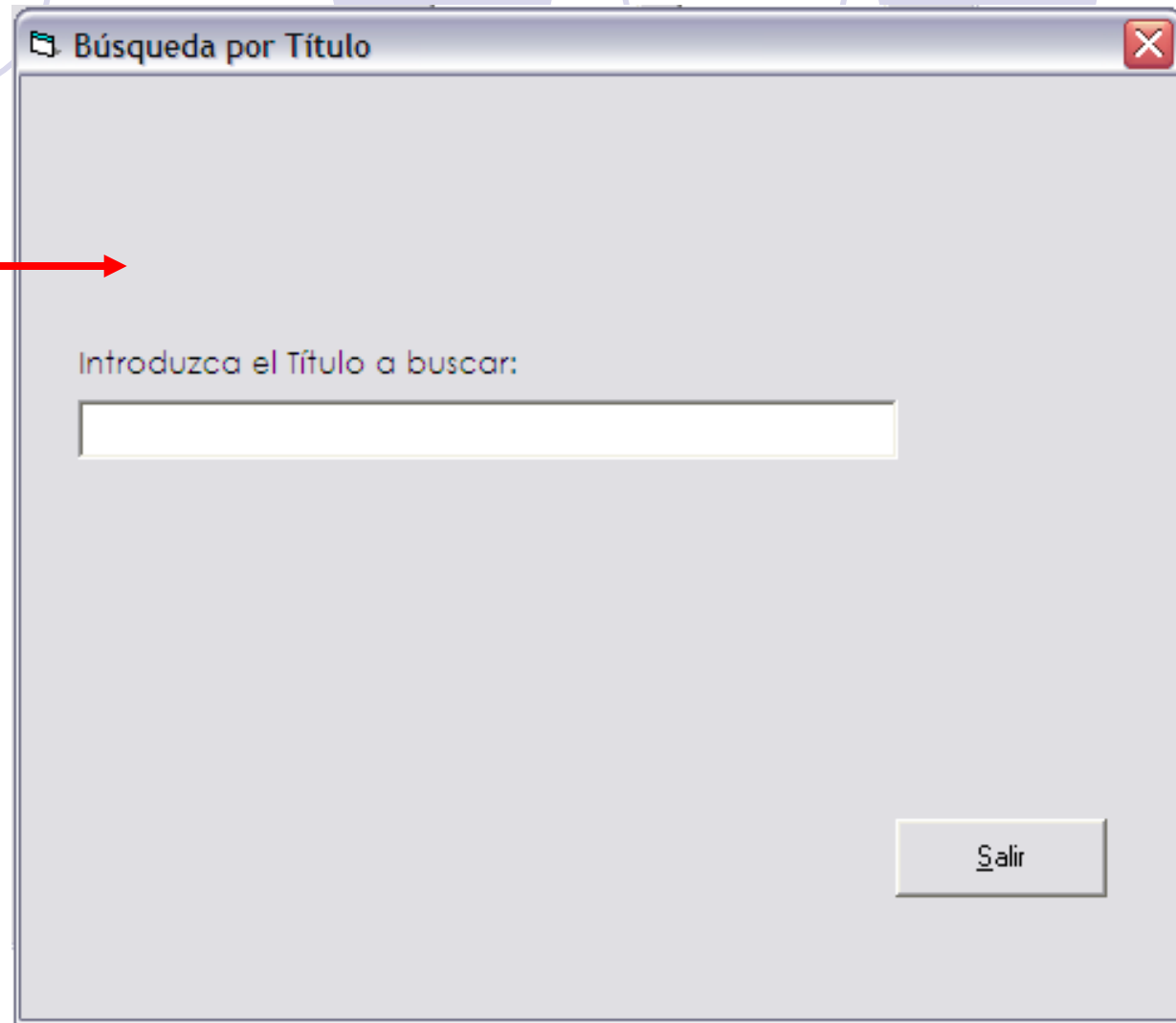
```
Private Sub SalirEditorial_Click()
    ' Cuando pulsamos el botón "Salir" se "descarga" este formulario
    ' y se "carga" el de la ventana principal (VentanaPrincipal).
    Unload Me
    VentanaPrincipal.Show
End Sub
```

Visual Basic

Ejemplo 11.



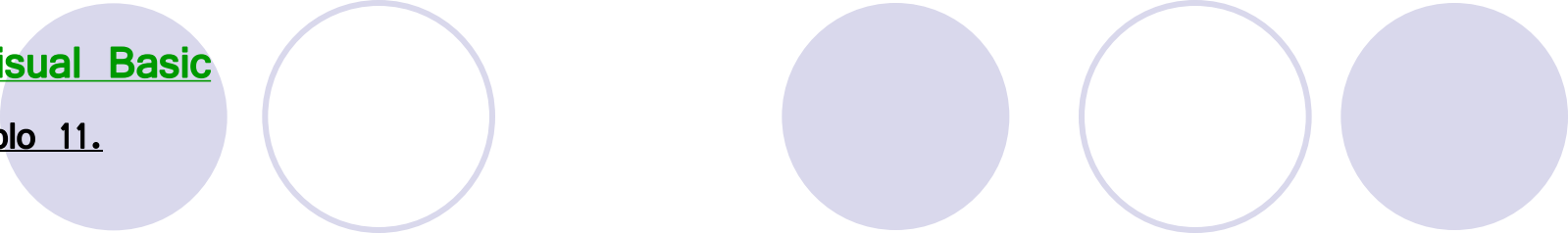
The screenshot shows a Visual Basic form titled "TITULO". On the left side, there is a vertical stack of five buttons: "Autor", "Editorial", "Título", "ISBN", and "Salir". The "Título" button is circled in red. A red arrow points from this button to the "Búsqueda por Título" form on the right. To the right of the buttons are four empty text boxes arranged in a 2x2 grid. Below the top-left box is the label "AUTOR", below the top-right box is "EDITORIAL", below the bottom-left box is "TITULO", and below the bottom-right box is "ISBN".



The screenshot shows a Visual Basic form titled "Búsqueda por Título". It features a single text input field with the placeholder text "Introduzca el Título a buscar:". At the bottom right of the form is a button labeled "Salir".

Visual Basic

Ejemplo 11.



Objeto	Name	Caption	BorderStyle
Formulario (Form)	PorTitulo	Búsqueda por Título	3
Botón de Comando (CommandButton)	SalirTitulo	&Salir	
Cuadro de Texto (TextBox)	Titulo		
Etiqueta (Label)	Label1	Introduzca el Título a buscar:	



Option Explicit

```
Private Sub Titulo_KeyPress(KeyAscii As Integer)
    ' Vamos introduciendo los caracteres del Título
    ' hasta que pulsemos ENTER. En ese momento, el título introducido
    ' pasa a la lista de títulos del formulario principal (VentanaPrincipal).
    If KeyAscii = 13 Then
        Load VentanaPrincipal
        VentanaPrincipal!TituloLista.AddItem Titulo.Text
        Titulo.Text = " "
    End If
End Sub
```

```
Private Sub SalirTitulo_Click()
    ' Cuando pulsamos el botón "Salir" se "descarga" este formulario
    ' y se "carga" el de la ventana principal (VentanaPrincipal).
    Unload Me
    VentanaPrincipal.Show
End Sub
```

Visual Basic

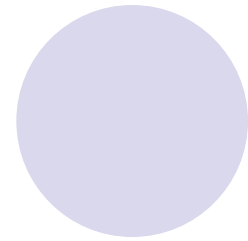
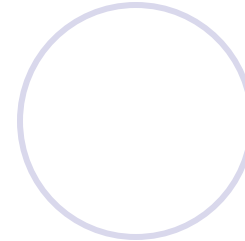
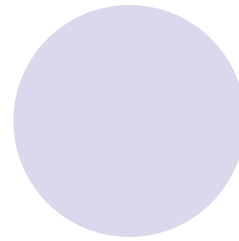
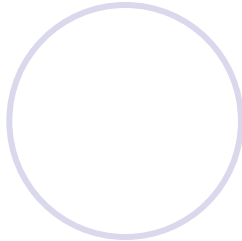
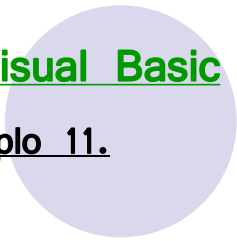
Ejemplo 11.

The 'TITULO' window contains a vertical stack of buttons on the left: 'Autor', 'Editorial', 'Titulo', 'ISBN', and 'Salir'. The 'ISBN' button is circled in red. To the right of these buttons are four empty rectangular text boxes arranged in a 2x2 grid. Below the top-left box is the label 'AUTOR', below the top-right box is 'EDITORIAL', below the bottom-left box is 'TITULO', and below the bottom-right box is 'ISBN'. A red arrow points from the circled 'ISBN' button to the 'Búsqueda por ISBN' window.

The 'Búsqueda por ISBN' window has a title bar with a close button. The main area contains the text 'Introduzca el ISBN a buscar:' followed by a single-line text input field. At the bottom right, there is a 'Salir' button.

Visual Basic

Ejemplo 11.



Objeto	Name	Caption	BorderStyle
Formulario (Form)	PorISBN	Búsqueda por ISBN	3
Botón de Comando (CommandButton)	SalirISBN	&Salir	
Cuadro de Texto (TextBox)	ISBN		
Etiqueta (Label)	Label1	Introduzca el ISBN a buscar:	

Option Explicit

```
Private Sub ISBN_KeyPress(KeyAscii As Integer)
    ' Vamos introduciendo los números de ISBN
    ' hasta que pulsemos ENTER. En ese momento, el ISBN introducido
    ' pasa a la lista de ISBN's del formulario principal (VentanaPrincipal).
    If KeyAscii = 13 Then
        Load VentanaPrincipal
        VentanaPrincipal!ISBNLista.AddItem ISBN.Text
        ISBN.Text = " "
    End If
End Sub
```

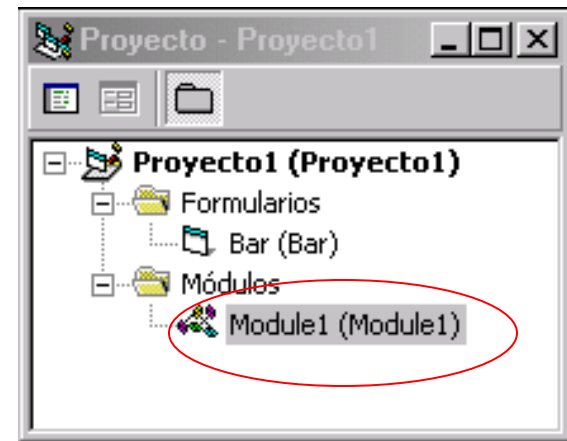
```
Private Sub SalirISBN_Click()
    ' Cuando pulsamos el botón "Salir" se "descarga" este formulario
    ' y se "carga" el de la ventana principal (VentanaPrincipal).
    Unload Me
    VentanaPrincipal.Show
End Sub
```

Visual Basic

Múltiple Formularios.

- **Observaciones:**

- Para el caso en el cual tengamos que definir variables a utilizar en distintos formularios, lo primero que hay que hacer es insertar un módulo:



- En el modulo hay que definir dichas variables
- como **PUBLIC**.
- Por ejemplo:
 - **Public** Var1 as String.
- A la hora de llamarlo desde otro formulario se realizará invocando, en primer lugar, el objeto (formulario) primero, después un punto (".") y por último el nombre de la variable a utilizar:
 - Texto2.Text = **Var1**