AREA INTERACTIVA

www.areaint.com

CURSO DE

Visual Basic 6.0

Lección 1

En esta lección de introducción aprenderemos las principales características de un lenguaje de programación para crear aplicaciones bajo **Windows**.

Programación con Visual Basic

Con **Visual Basic** podemos crear cualquier tipo de aplicación para que funcione bajo **Windows**, utilizando así todos y cada uno de los elementos que forman parte de éeste. Si estamos familiarizados con **Windows** conoceremos de sobra estos elementos y quée es lo que solemos hacer con cada uno de ellos.

Si nosotros ejecutamos una aplicación como la **calculadora** podemos observar que es un programa que actúa de forma independiente, que tiene su tamaño delimitado y que lo que ocurra en su interior, en un principio, no afectará al resto de las aplicaciones que se estén ejecutando en este momento en **Windows**.

Podemos observar, que en un principio este programa, como la mayoría de programas que funcionan en **Windows** no realizan ningún tipo de acción a no ser que nosotros actuemos sobre él, dicho de otra manera, el programa espera a que nosotros le digamos que es lo que tiene que hacer.

. *Ejecuta la **calculadora**: **Inicio** – **Programas** – **Accesorios** – **Calculadora***.

Verás que aparece en el escritorio de **Windows** la siguiente ventana.

Observa que esta ventana está limitada con respecto al resto de **Windows** por el borde rectangular que la rodean. Observa también que la calculadora no realiza ningún tipo de acción ni operación, nos está esperando a que nosotros actuemos sobre ella.

Vamos a actuar sobre nuestra calculadora.

. *Pulsa sobre el **botón 6**, observa lo que pasa*.

Al pulsar sobre el **botón** con el número **6** hemos generado una actividad; hemos hecho que la calculadora realice una acción o evento. De esta forma podemos decir que un evento sería cualquier tipo de acción que se realiza sobre alguno de los objetos que

forman parte de una aplicación o programa. Un evento, por ejemplo, podría ser: hacer doble clic con el ratón sobre una casilla de texto, mover el ratón sobre la propia aplicación, pulsar una tecla, etc.

¿Cómo programaríamos en Visual Basic?

Como hemos podido ver, en **Visual Basic** las acciones que debe realizar un programa se realizan al generarse un evento. Así podemos decir que nuestras líneas de código estarán dentro de cada uno de los eventos de cada elemento que forman parte de nuestra aplicación.

Cuando pensamos en una aplicación para programarla en **Visual Basic** tenemos que pensar en cuales serán los eventos que realizarán las acciones y que condiciones deben cumplir los elementos que forman parte de la aplicación, para que estos actúen correctamente.

Vamos a ver la anterior explicación utilizando como ejemplo la calculadora.

. Práctica 1

*1.- Abre la **calculadora**.*

*2.- Pulsa sobre el **botón 6**.*

De esta forma podemos ver que el contenido del **botón** (el número 6) a pasado al cuadro de texto donde irán apareciendo las cantidades y resultados de nuestras operaciones, pero antes de esto se ha borrado el 0 que estaba en este recuadro de texto.

*3.- Vuelve a pulsar el **botón 6**.*

Fíjate en lo que ha pasado ahora. El nuevo 6 no ha sustituido (borrado) lo que había en el cuadro de texto, sino lo que ha hecho la calculadora es poner el segundo 6 seguido del primero con lo que tenemos el número 66.

Con esto podemos ver que el **botón 6** ha actuado de dos formas diferentes, aunque nosotros lo hallamos activado igual. ¿Por qué el **botón 6** ha actuado así? Pues por la simple razón que el botón antes de actuar ha mirado a su alrededor y según lo que ha visto ha reaccionado de una forma u otra. Al decir que mira a su alrededor queremos decir que mira que propiedades y características cumplen los otros elementos que forman parte de la aplicación.

Pues bien, nosotros como buenos programadores deberemos tener en cuenta que es lo que nos interesa que realice un objeto en cada momento determinado y como queremos que lo realice. Para que esto sea así nos debemos plantear cuando, como y porque el usuario realizará un evento y como debe actuar este.

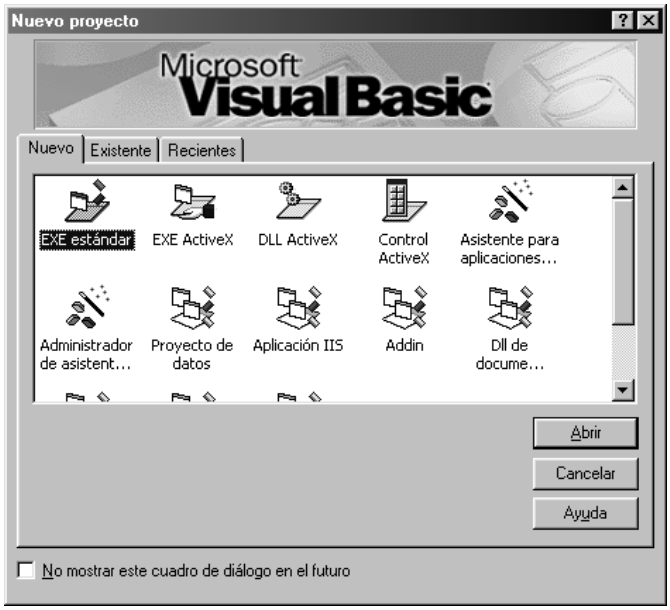
Debemos pensar que este punto, junto con la comunicación con el usuario (ya hablaremos más adelante), son dos de los puntos más importantes dentro de la programación al estilo de **Visual Basic**.

Empecemos a trabajar

Antes de nada vamos a familiarizarnos un poco con el entorno de trabajo de **Visual Basic** mirando partes, nombres y funciones más características del entorno de trabajo para así poder empezar a crear nuestras aplicaciones. Este primer acercamiento será superficial ya que solo echaremos un vistazo. Conforme avancemos en el curso iremos adentrándonos más en sus características y funciones. Es importante que aprendas los nombres de las diferentes partes de **Visual Basic** ya que en las próximas lecciones nos referiremos a ellas por su nombre.

. Práctica 2

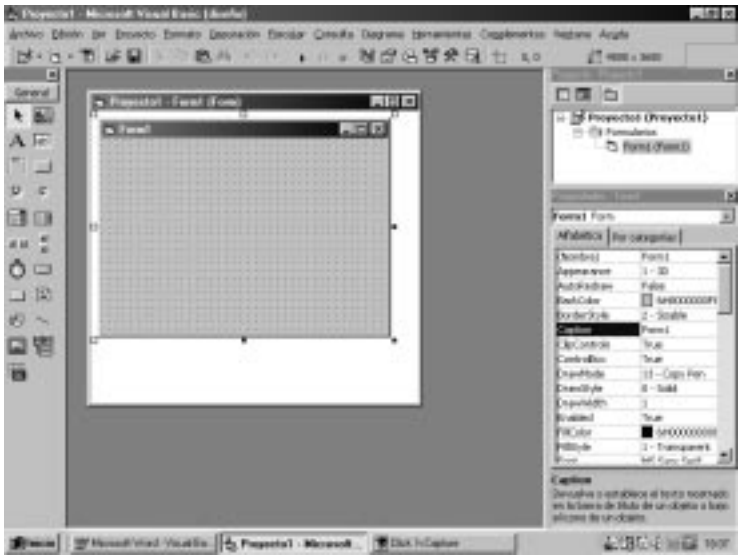
1.- Inicia Visual Basic: ***Inicio*** – ***Programas*** – ***Microsoft Visual Studio 6.0*** – ***Microsoft Visual Basic 6****.*



Al iniciar **Visual Basic** te aparecerá en primer termino una pantalla como esta:

2.- Haz un clic en Aceptar para iniciar un nuevo proyecto.

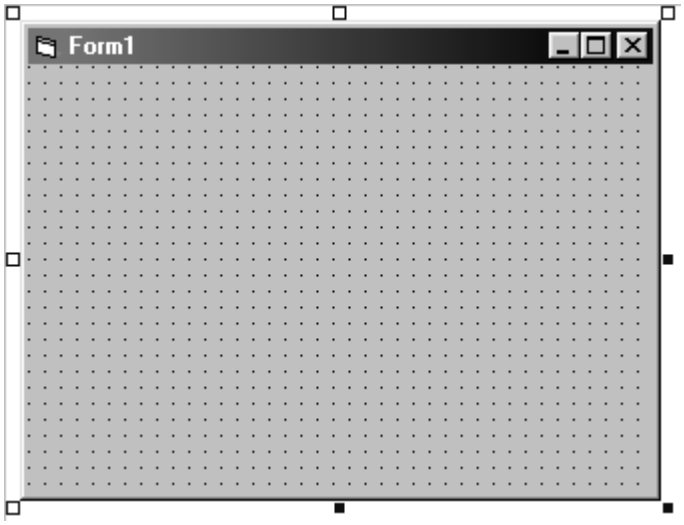
Observa la siguiente pantalla e identifica las partes que iremos nombrando a continuación.



Barra de menús

En las barras de menús tenemos todas las opciones, utilidades y comandos de los que podemos disponer en **Visual Basic**. (Archivo, Edición, Ver, etc.)

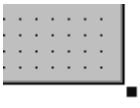
Formulario



Esta es una de las partes más importantes, ya que aquí es donde diseñaremos la pantalla o pantallas que formarán parte de nuestro programa. A estas pantallas le llamaremos formularios. Aquí iremos "pegando" y modificando los diferentes elementos de nuestra aplicación, como puedan ser botones, cuadros de texto, etc. Si no viéramos la pantalla del formulario podríamos activarla desde **Ver** – **Objeto** o pulsar **Mayúsculas** + **F7**.

El diseño de una pantalla es tan simple como arrastrar los objetos que deseamos, desde el **cuadro de herramientas** hasta el **formulario**. Para modificar el tamaño de cualquier objeto, incluso del **formulario** solo es necesario situarse en cualquiera de las esquinas del objeto o en el centro de uno de sus lados marcados con un cuadrado, esperar que el ratón se convierta en una fecha de desplazamiento, pulsar el botón izquierdo del ratón y mientras se mantiene pulsado movernos hasta que el objeto tome un nuevo tamaño. Si cambiamos el tamaño desde uno de los vértices podremos modificar tanto el alto como el ancho, mientras que si arrastramos desde uno de los lados solo podremos modificar el alto o el ancho dependiendo del lado en el que nos encontremos.

. Práctica 3



*1.- Sitúate sobre la esquina inferior derecha del **formulario**, sobre el cuadrado pequeño inferior.*

*2.- Espera hasta que el ratón se convierta en una doble flecha, pulsa y arrastra hasta que veas como el **formulario** cambia de tamaño.*

Así de fácil.

Cuadro de herramientas

En este cuadro encontramos las herramientas que podemos utilizar para diseñar nuestro proyecto. El **cuadro de herramientas** que presentamos a continuación es el estándar, el cual contiene los elementos básicos. Más adelante veremos como podemos agregar elementos a este **cuadro de herramientas**.

A continuación vamos a nombrar las herramientas básicas, para así poder empezar a crear una pequeña aplicación. En futuras lecciones iremos explicando el resto de herramientas.

Puntero. Utilizaremos este control para poder mover, cambiar el tamaño o seleccionar los diferentes elementos que insertemos en el formulario.

Label. Utilizaremos este control para escribir etiquetas donde aparecerá texto que el usuario no podrá cambiar.

TextBox. Son cuadros de texto que el usuario podrá cambiar.

CommandButton. Utilizaremos este control para crear botones sobre los cuales podrá actuar el usuario.

CheckBox. Casilla que el usuario podrá utilizar para marcar dos posibles opciones. Verdadero o falso, sí o no, activado, desactivado... El usuario podrá marcar la cantidad de casillas de verificación que desee dentro de una aplicación.

OptionButton. Muy parecida al control anterior, pero el usuario solo podrá marcar una de las opciones. Si tenemos dos controles de este tipo, en el momento de seleccionar uno automáticamente se quitará la selección el otro.

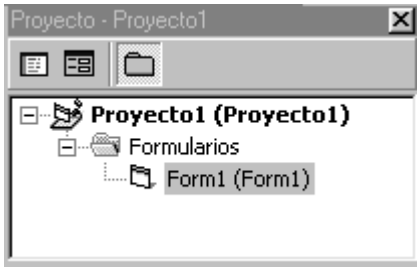
Para visualizar el **cuadro de herramientas** podremos ir a la opción **Cuadro de herramientas** dentro de la opción **Ver** o hacer un clic en este botón: en la barra de herramientas (definida a continuación).

Barra de herramientas

Desde las **barras de herramientas** podemos acceder a todas aquellas instrucciones o comandos que son usados cuando estamos editando y programando nuestra aplicación (**Grabar**, **abrir**, **ejecutar**, mostrar diferentes elementos de **Visual Basic**, etc.). Al iniciar **Visual Basic** aparece una **barra de herramientas estándar**. Nosotros podemos ocultar o mostrar otras barras de herramientas, las cuales ya veremos.

Para visualizar la **Barra de herramientas estándar** debemos ir a la opción **Barra de herramientas** dentro de la opción **Ver**. Allí podremos encontrar diferentes **Barras de herramientas** para que se active una de ellas solo deberás hacer un clic sobre el nombre deseado. En este caso haríamos un clic sobre **Estándar**.

Explorador de proyectos



Desde el **explorador de proyectos** podemos ver todas "las pantallas", formularios, que componen nuestra aplicación.

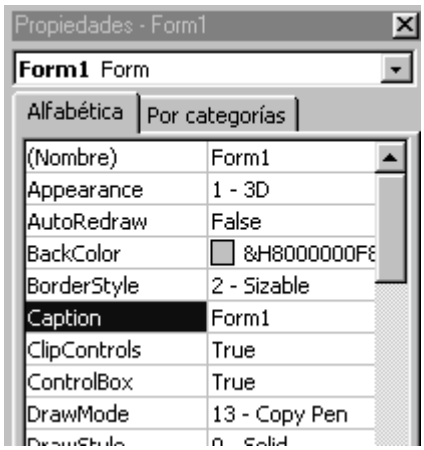
Para poder visualizar el **explorador de proyectos** deberás ir a **Ver – Explorador de proyectos**, pulsar la combinación de teclas **Ctrl + R** o pulsar sobre este botón: en la barra de herramientas.

Ventana de propiedades

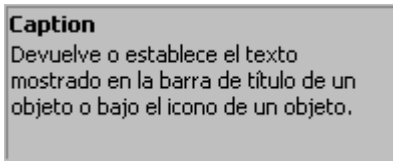
En esta pantalla vemos las **propiedades** de los objetos que tenemos seleccionados. (Las **propiedades** las veremos con más detenimiento en futuras lecciones). Las **propiedades** son las características que puede tener cada uno de los elementos como puede ser su tamaño, su posición, su contenido, su color, su forma, su tipo de letra, etc. Todas estas propiedades se pueden cambiar cuando nos encontramos en forma **diseño**, creando el programa, o en forma **ejecución**, cuando estamos ejecutando la aplicación.

Para cambiar una propiedad de un objeto cuando estamos en modo diseño, solo tenemos que seleccionar el **objeto** ir a la **ventana de propiedades** y cambiar la propiedad que nos interese. Más adelante realizaremos unas cuantas prácticas donde veremos como hacerlo.

Si no nos aparece la ventana de propiedades podemos pulsar **F4**, o ir a la opción de la barra de menús **Ver** – **Ventana propiedades** o como última opción utilizar el botón de la **barra de herramientas**:



Observa que en la parte inferior de la **ventana de propiedades** aparece un pequeño cuadrado en el que tienes una pequeña ayuda sobre la propiedad seleccionada.



Los demás elementos que aparecen en tu pantalla los iremos comentado en siguientes lecciones.

Primera aplicación

Vamos a realizar una pequeña aplicación donde podremos empezar a utilizar todo lo que hemos visto hasta el momento. Si alguna de las cosas que explicamos no te queda del todo clara, no te preocupes, ya lo irás entendiendo a medida que avances en el curso. Lo importante de esta práctica es crear una primera aplicación donde veas el funcionamiento de diferentes objetos y las propiedades de estos. Así que sin más demora, adelante y sin miedo.

. Práctica 4

*1. Inicia **Visual Basic 6.0**.*

*2. De la pantalla **Nuevo proyecto** escoge la opción **EXE estándar** y pulsa **Aceptar**.*

Después de unos segundos tendrás en pantalla un nuevo **formulario**, donde crearemos nuestra primera aplicación.

Tamaño del formulario

*3. Pulsa un clic sobre el **formulario**, observa como en el **cuadro de las propiedades** aparece el nombre del **formulario**, que por defecto es **Form1**.*

*4. Busca la propiedad **Height** (Las propiedades están ordenadas alfabéticamente).*

*5. Haz doble clic sobre esta propiedad y escribe **3100**. Pulsa **Intro**.*

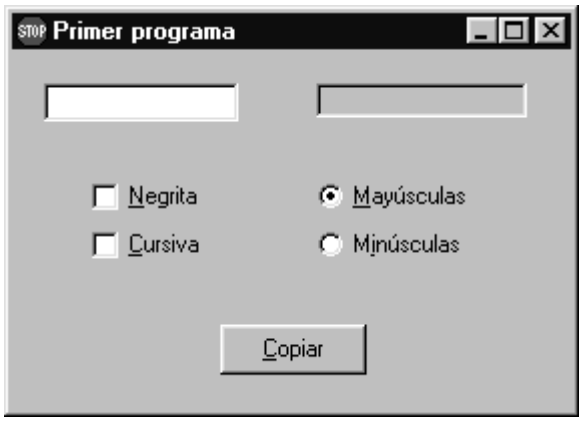
Observa como el ancho de nuestro formulario ha cambiado.

*6. Busca la propiedad **Width**.*

*7. Haz doble clic sobre dicha propiedad y escribe **4300**.*

Ahora podemos observar como la altura de nuestro formulario ha cambiado.

Vamos ha empezar a colocar los elementos necesarios para que funcione nuestra aplicación. De tal forma que queden como en la siguiente imagen. (Sigue los pasos que te indicamos, no te avances)

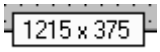


*8. Colócate con el puntero del ratón en el **cuadro de herramientas** sobre del control **CommandButton**.*

*9. Pulsa un doble clic sobre este control, verás como ha aparecido un **botón** en el centro de nuestro **formulario**.*

Cambio del tamaño del botón

10. Sitúate sobre la esquina inferior derecha de dicho elemento.



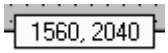
11. Mantén pulsado el ratón hasta que aparezca el siguiente recuadro:

(Puede ser que los valores de tu recuadro no sean iguales que los que aparecen en esta imagen). Este recuadro nos informa del ancho y alto del objeto.

*12. Muévete, arrastrando hasta que dentro del recuadro aparezca **1215 x 375**. Cuando consigas estos valores suelta el botón del ratón.*

Fíjate en las siguientes propiedades y sus valores dentro del cuadro de propiedades **Height** = **375** y **Width** = **1215**.

Cambio de posición de un objeto.



*13. Haz **un clic** sobre el botón que acabamos de crear. Después de unos instantes te aparecerá un recuadro con dos números separados con una coma.*

Este recuadro nos indica la posición que se encuentra el elemento con respecto

a la esquina izquierda superior de nuestro **formulario**.

*14. Mantén pulsado el botón del ratón y muévete hasta la posición **1560, 2040** aproximadamente.*

Ahora observa los valores de las propiedades **Top** = **2040** y **Left** = **1560**.

Es importante que recuerdes para que se utilizan las propiedades: **Height**, **Width** y **Top**, **Left**.

Cambio del nombre del botón

La propiedad **(Nombre)**, nos servirá para referirnos a este objeto en el momento que estemos programando.

*15. Selecciona el **botón** haciendo un clic sobre él. Pulsa **F4**. Este punto es solo necesario en caso de no tener el botón seleccionado.*

*16. Haz un doble clic en la propiedad **(Nombre)**, (está situada en la primera posición).*

*17. Escribe **Copiar**. Pulsa **Intro**.*

A partir de este momento siempre que queramos hacer referencia al botón de nuestro formulario utilizaremos el nombre **Copiar**.

Cambio del texto del botón.

Ahora, para que el usuario de nuestra aplicación tenga un poco de idea que hace este botón vamos a cambiar su texto.

*18. Vuelve a pulsar **F4**.*

*19. Haz un doble clic sobre **Caption** y escribe **C&opia***

El signo **&** delante de la **o** nos marcará la combinación de teclas que podremos utilizar para que se active nuestro botón. En este caso sería **Alt+o**. Observa como dentro del botón aparece escrito **Co<u>p</u>ia**.

Vamos a colocar los demás elementos que forman parte de nuestra aplicación.

Creación de un TextBox

*20. Pulsa doble clic sobre el **TextBox**.*

*21. Colócalo utilizando el método que quieras dentro del **formulario** en la posición **240, 240** con un tamaño de **1455 x 285**.*

*22. Cambia la propiedad **(Nombre)** por **Texto**.*

*23. Sitúate sobre la propiedad **Text** y borra el contenido.*

De esta forma haremos que cuando iniciemos el programa no aparezca ningún texto en el interior de este objeto.

Creación de un Label

*24. Coloca un **Label** en la posición **2280, 240** con un tamaño de **1575 x 255**.*

*25. Cambia su nombre por **Etiqueta**.*

*26. Sitúate sobre la propiedad **Caption** y borra el contenido.*

De esta manera haremos que cuando ejecutemos la aplicación no exista ningún texto dentro de este objeto.

Fíjate que para cambiar el contenido del objeto **TextBox** utilizamos la propiedad **Text**, mientras que en el objeto **Label** utilizamos **Caption**.

*27. Sitúate sobre la propiedad **BorderStyle** del **Label**. Abre la lista desplegable de la misma propiedad y escoge la opción **1-Fixed Single**.*

Con esta opción lo que conseguimos es que el **Label** tenga un borde, con el que podemos ver el límite de este control.

Creación de CheckBox

Vamos a colocar dos **CheckBox**, con los que controlaremos si queremos el texto en **Negrita**, **Cursiva** o las dos cosas. Recuerda que los controles **CheckBox** pueden estar los dos activados, uno solo, o los dos desactivados.

*28. Pulsa doble clic sobre el **CheckBox** del **Cuadro de herramientas**.*

*29. Sitúalo en la posición **600, 840***

*30. Coloca otro **CheckBox** en la posición **600, 1200***

*31. Cambia el **nombre** del primero por: **Negrita** y al segundo **Cursiva**.*

*32. Cambia el **Caption** del primero de ellos por **Negrita** y el segundo por **Cursiva**. Observa cual será en cada caso la tecla que activará este objeto.*

Fíjate en la imagen del principio de la práctica para ver como han de quedar los controles.

Creación de OptionButton

Ahora colocaremos dos **OptionButton**, con estos nuevos controles podremos controlar si lo que queremos es que aparezca todo el texto en **Mayúsculas** o en **minúsculas**. Utilizamos este tipo de control ya que solo podemos hacer que el texto aparezca todo en mayúsculas o todo en minúsculas.

*33. Pulsa doble clic sobre el **OptionButton** del **Cuadro de herramientas**.*

*34. Sitúa el primer **OptionButton** en la posición: **2280, 840** y el segundo en la posición: **2280, 1200***

*35. Cambia el **nombre** de los dos controles por **Mayusculas**, el primero y **Minusculas**, el segundo.*

Observa que en el nombre no hemos puesto acentos. Podríamos ponerlos pero hay que pensar que muchos lenguajes de programación no los aceptan.

*36. Cambia el **Caption** de ambos por **Mayúsculas** y **Minúsculas**.*

Fíjate en la imagen del principio de la práctica para ver como han de quedar los controles.

El tamaño de estos controles no lo controlamos ya que los bordes de estos

elementos no se ven en el modo de ejecución.

Cambio del título e icono del formulario.

37. Selecciona el formulario.

*38. Accede a la propiedad **Caption** y escribe: **Primer programa**.*

Veras que mientras lo escribes aparece en el título del formulario.

*39. Ahora accede a la propiedad **Icono** y pulsa en este botón*

Te aparecerá una ventana típica de **Windows** para búsqueda de archivos.

*40. Accede al directorio donde tienes instalado **Visual Basic**. Selecciona el archivo **Trffc14.ico** que se encuentra dentro del siguiente directorio **Graphics\Icons\Traffic***

Acto seguido aparecerá un icono en el **formulario**.

Perfecto, ya tenemos colocados todos los elementos que forman parte de nuestra primera aplicación. Ahora solo nos queda completar el código con el cual la aplicación realizará su cometido.

Introducción al código

¿Dónde colocaremos el código de nuestra aplicación? En esta aplicación es muy fácil saber, ya que tenemos que colocar el código allí donde al realizar un evento se produzca una "reacción". Bien, en nuestro caso queremos que se realice cuando pulsemos el botón **Copiar**.

Tenemos que pensar que cada evento podrá tener una serie de instrucciones que se ejecutarán cuando éste se produzca. A este grupo de instrucciones dentro de un evento le llamaremos **procedimiento de evento**. Cada **procedimiento de evento** se distingue de otro porque aparece el nombre del control **(Nombre)**, más un carácter **_** y el nombre del evento. Por ejemplo **Boton_Click**, indica que el procedimiento se ejecutará cuando se hace un **clic** sobre el botón llamado **Boton**.

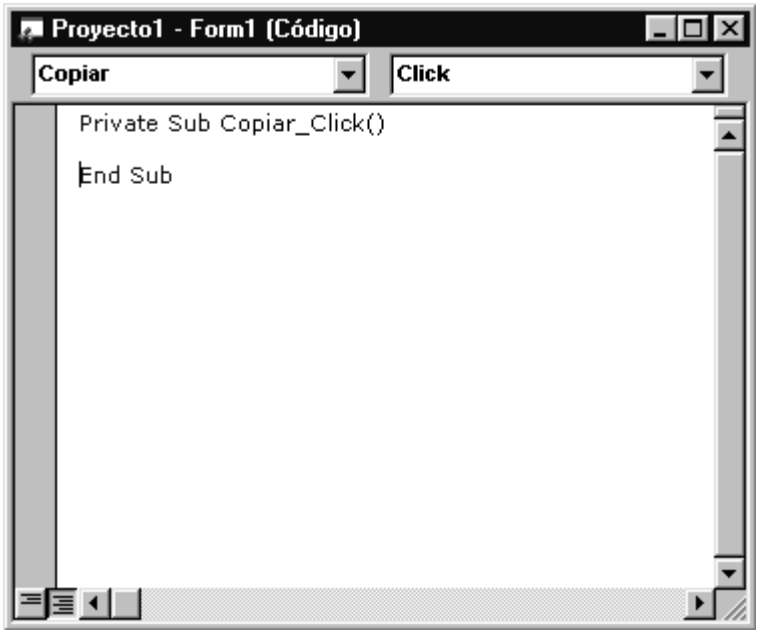
Nosotros desde el interior de un **procedimiento** podemos cambiar la propiedad de cualquier elemento que exista en nuestro formulario. Esto lo haremos indicando el **nombre del objeto** al que queremos cambiar la propiedad seguido de un punto (**.**) y el **nombre de la propiedad** a cambiar. Por ejemplo **Etiqueta.Caption = "Cambio de texto"**, con esto cambiaríamos el **Caption** de un **Label** llamado **Etiqueta** haciendo que aparezca "Cambio de texto". En lecciones posteriores veremos con mucho más detenimiento las instrucciones y comandos de **Visual Basic**.

En nuestro ejemplo queremos que al pulsar el botón **Copiar** el ordenador copie en el **Label** el texto que hay en el **TextBox** con los formatos que indique los demás elementos: **Mayúsculas** o **minúsculas**, **Negrita**, **Cursiva**.

. Práctica 4 (Segunda parte)

*1. Pulsa doble clic sobre el botón **Copiar**.*

Acto seguido aparecerá una ventana como esta:



En esta ventana será donde nosotros introduciremos el código que queremos que realice nuestro procedimiento.

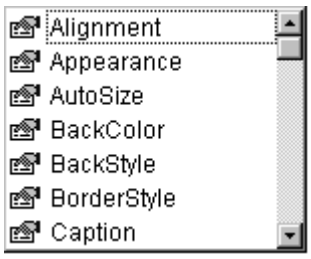
El código deberá estar entre las dos líneas que aparecen ya escritas, ya que estas nos indican el principio y el final de dicho **procedimiento de evento**.

La primera línea nos indica que estamos programando dentro del evento **Click** (hacer un clic con el ratón) dentro del objeto **Copiar**. Y la segunda línea nos indica el final de dicho procedimiento de evento.

Antes de empezar a copiar el código que irá en este botón explicaremos una "herramienta" que forma parte de **Visual Basic** que nos facilita un poco el trabajo y nos ayuda a la hora de escribir el código.

Vamos a introducir una primera línea de código poco a poco para ver que es lo que ocurre.

2. Escribe lo siguiente: ***Mayusculas.***



Observa como acto seguido de poner un punto te aparece una especie de menú contextual parecido a este:

En este menú contextual han aparecido todas las propiedades del objeto **Mayusculas**.

*3. Escribe **v**.*

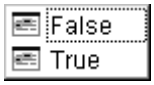
Observa como la lista ha saltado hasta encontrar la primera palabra que empezaba con **V**.

*4. Pulsa la tecla **Tab**.*

Observa como automáticamente ha aparecido escrito en pantalla **Value**.

5. Escribe **=**

Acto seguido aparece otro menú contextual con solo dos opciones:



*6. Escribe **T** (es igual en minúsculas que en mayúsculas).*

*7. Pulsa **Intro** ya que hemos llegado al final de la línea.*

Observa como **Visual Basic** coloca los espacios en los lugares correspondientes. Si **Visual Basic** hubiera encontrado algún error de escritura nos lo hubiera hecho saber con un mensaje de error y poniendo la línea en color rojo.

Cuando empieces a escribir el código podrás ver que según que tipo de instrucción introduzcas **Visual Basic** te ofrecerá otra especie de menú contextual con la estructura de esta instrucción. Este será el caso, por ejemplo, de la instrucción **UCase** que escribirás en las siguientes líneas de código.

8. Borra la línea de código que has escrito.

No borres las dos líneas de las que hemos estado hablando en el principio de este capítulo.

9. Copia el siguiente código, entre las líneas que te hemos indicado al principio de este capítulo:

```
Etiqueta.Caption = Texto.Text
If Negrita.Value = 1 Then
   Etiqueta.Font.Bold = True
Else
   Etiqueta.Font.Bold = False
End If
If Cursiva.Value = 1 Then
   Etiqueta.Font.Italic = True
```

```
Else
   Etiqueta.Font.Italic = False
End If
If Mayusculas.Value = True Then
   Etiqueta.Caption = UCase(Etiqueta.Caption)
Else
   Etiqueta.Caption = LCase(Etiqueta.Caption)
End If
```

Vamos a comentar un poco las líneas que hemos utilizado en nuestra aplicación:

Etiqueta.Caption = Texto.Text 'Copiamos el contenido de la casilla de texto **Texto.Text** a la etiqueta **Etiqueta.Caption**

If Negrita.Value = 1 Then 'Con la instrucción **If** hacemos una pregunta que el ordenador nos contestará con Verdadero o Falso. (Esta estructura la veremos con mucho más detenimiento en lecciones futuras pero ahora adelantamos la estructura para que sea más fácil el entendimiento del código).

```
If [Pregunta] Then
[Instrucciones cuando la Pregunta es verdadera]
      ...
Else
[Instrucciones cuando la Pregunta es falsa]
      ...
End If
```

En nuestro caso preguntamos si el **CheckBox** llamado **Negrita** está activado. Esto lo hacemos con la propiedad **Value** que solo puede tener dos valores **1 = activado** o **0 = desactivado**.

En el caso que la casilla **Negrita** esté activada (**Value = 1**), primera parte de la instrucción **If**, entonces el contenido de la **Etiqueta** se pondrá en Negrita poniendo la propiedad **Etiqueta.Font.Bold** a Verdadero (**True**) de la siguiente manera: **Etiqueta.Font.Bold = True**

En caso que la casilla **Negrita** no esté activada (**Value = 0**), segunda parte de la instrucción **If**, entonces el contenido de la **Etiqueta** no aparecerá en negrita, poniendo la siguiente instrucción **Etiqueta.Font.Bold = False**

En el siguiente **If** lo que hacemos es mirar si el **CheckBox** llamado **Cursiva** está activado. Si está activado pondremos la propiedad **Etiqueta.Font.Italic** a verdadero (**True**), mientras que si no está activado, **Else**, pondremos en valor a falso (**False**).

If Mayusculas.Value = True Then 'Con este otro **If** lo que hacemos es mirar si el **OptionButton** llamado **Mayusculas** está activado. Observa que en este tipo de objeto miramos si está **activado** con un **True** y **desactivado** con un **False**. En el caso de estar activado lo que hacemos, en la primera parte del **If** es: **Etiqueta.Caption = UCase(Etiqueta.Caption)**. Esta instrucción funciona de la siguiente manera. Siempre que tenemos una igualdad la tenemos que leer de derecha a izquierda, así esta instrucción se leería de la siguiente forma. Cogemos el contenido de **Etiqueta**, cosa que hacemos con **Caption**, lo convertimos en mayúsculas con **UCase** y lo que tenemos (el contenido de la **Etiqueta** en mayúsculas) lo volvemos a poner en el **Caption** de nuestra **Etiqueta**.

Ahora tendríamos que mirar si lo que está activado es el **OptionButton** llamado **Minusculas**, pero no lo haremos mediante otro **If** ya que como vimos en la explicación de los objetos cuando seleccionamos uno dejamos de tener seleccionado el otro de tal forma que siempre tendremos uno seleccionado. Por lo que utilizaremos el **Else** del mismo **If** para controlar ya que si no tenemos seleccionado **Mayusculas** lo estará

Minusculas. Para poner el texto en minúsculas utilizaremos la instrucción **LCase**.

Con estas líneas comprobamos todas las posibles combinaciones que podemos hacer con nuestra pequeña aplicación. Intenta entender el pequeño código, si algo no lo entiendes tranquilo ya que más adelante explicaremos con más detenimiento estructuras e instrucciones.

*10. Inicia una ejecución de prueba pulsando **F5** o pulsando el siguiente **botón**.*

11. Realiza las pruebas que quieras sobre la aplicación.

Recuerda que solo se copiarán y se visualizarán los cambios cuando pulsemos el botón **Copiar**.

12. Finaliza la ejecución de la aplicación cerrando la pantalla.

Guardar el formulario y el proyecto

Cuando realizamos una aplicación como la que hemos hecho en esta lección hemos creado una o varias ventanas llamadas **formularios** y al conjunto de estos **formularios** le llamamos **proyecto**.

. Práctica 4 (Tercera parte)

Para grabar el **formulario** que hemos creado realizaremos los siguientes pasos.

*1. Accede a **Guardar Form1 como...** dentro del menú **Archivo**.*

*2. Accede al directorio donde quieras guardar tus formularios, ponle el nombre que desees y pulsa en **Guardar**.*

Fíjate que el formulario que has guardado tiene como extensión **frm**

Ahora guardaremos el **proyecto**.

*3. Accede a **Guardar proyecto como...** dentro del menú **Archivo**.*

*4. Accede al mismo directorio donde has guardado tu **formulario**. Escribe **Primer programa** y pulsa en **Guardar**.*

Fíjate que el proyecto se guardará con extensión **vbp**.

Ahora vamos a abrir un formulario nuevo, para así poder abrir el formulario recién guardado.

*5. Escoge dentro del menú **Abrir** la opción **Nuevo Proyecto**.*

Si te aparece una pantalla preguntando si deseas guardar los cambios responde negativamente.

*6. En la siguiente pantalla pulsa en **Aceptar**.*

Ahora ya tenemos nuestra primera aplicación guardada y en pantalla un nuevo proyecto para seguir trabajando. En lecciones futuras veremos como crear un archivo ejecutable de nuestra aplicación.

Abrir el proyecto

Para abrir un proyecto que tenemos guardado solo deberemos abrir el proyecto y no los formularios que forman parte de él, ya que esto lo hará automáticamente **Visual Basic**.

*1. Accede a al opción **Abrir proyecto** del menú **Abrir**.*

En un momento aparecerá una pantalla típica para abrir ficheros, con la única diferencia que en la parte superior aparecen dos pestañas

Desde la carpeta **Reciente** podrás abrir los proyectos que has abierto o guardado recientemente con **Visual Basic**. Observa que en primera posición, si no has abierto ningún proyecto después de guardar el tuyo, aparece **Primer programa** junto con la carpeta donde ha sido guardado.

En cambio en la carpeta **Existente** podrás abrir cualquier proyecto que este en tu disco de trabajo. Solo tendrás que buscar el proyecto en las carpetas que tengas en tu disco de trabajo.

*2. Accede a la carpeta **Reciente**, y pulsa un doble clic sobre el proyecto **Primer programa**. En pocos segundos verás como aparece en pantalla el formulario de nuestra aplicación.*

Si no te aparece alguno de los componentes de la aplicación utiliza los métodos que hemos explicado al principio de la lección para poderlos ver.

Fin de la lección 1

LECCIÓN 2

En las dos siguientes lecciones vamos a crear una nueva aplicación mediante la cual iremos explicando nuevos objetos y propiedades. Es importante que tú también intentes averiguar para que sirven algunas de las propiedades de estos nuevos objetos que hasta este momento no se hayan explicado.

Crearemos una aplicación que nos permitirá realizar una simple operación matemática entre dos números que introduciremos en dos casillas de texto. Las posibles operaciones a realizar serán la **suma**, la **resta**, la **multiplicación** y por último la **división**. Tendremos una lista en la que podremos ir añadiendo las **operaciones** o las **soluciones** de las operaciones que vamos realizando. La apariencia de nuestra práctica será más o menos la siguiente:

Durante estas lecciones aprenderemos como depurar nuestra aplicación para evitar errores y hacer más fácil el manejo a un posible usuario. Haciendo aparecer, según nos convenga, cuadros de ayuda y mensajes de error.

Propiedades del formulario

En este capítulo vamos a familiarizarnos con algunas de las propiedades más importantes de los formularios, como puede ser la posición en la pantalla del formulario cuando se inicia la aplicación, el color de fondo, los botones de maximizar, minimizar y cerrar, etc.

. Práctica 1

*1. Inicia **Visual Basic** y haz lo necesario para que te aparezca un nuevo formulario en pantalla.*

Una vez tenemos el formulario en pantalla vamos a cambiarle el tamaño. Recuerda que tienes varias maneras de hacerlo. Utiliza el sistema que tú prefieras. Mira la lección anterior.

*2. Pon las propiedades **Height** a **5775** y **Width** a **6045**.*

Posición al iniciar la ejecución

*3. Haz un clic en **Ventana posición del formulario** del menú **Ver**.*

Observa como en algún lugar de la pantalla te ha aparecido una ventana como esta:

Esta ventana nos ofrece una simulación de lo que sería nuestro formulario dentro de la pantalla del ordenador.

*4. Sitúate encima del recuadro blanco donde aparece la palabra **Form1**.*

Observa como te ha aparecido un cursor, más o menos como este:

Si mantienes pulsado el botón izquierdo del ratón podrás ver como puedes mover el formulario a cualquier parte de la pantalla negra. Con esto conseguimos que el formulario en el momento de ejecutarse se inicie en el lugar que hemos situado el recuadro **Form1**.

5. Coloca el dibujo del formulario en una de las esquinas e inicia una ejecución de prueba. Acto seguido detén la ejecución de prueba.

Observa como el formulario aparece en el lugar de la pantalla que tu le has indicado. Con esta misma pequeña ventana podemos hacer que el formulario, siempre nos aparezca centrado en la pantalla.

6. Sitúate sobre el dibujo del formulario. Pulsa el botón derecho del ratón para que aparezca el menú contextual. Haz un clic en ***Guías de resolución****.*

Con esta opción podrás ver unas guías que te indican como sería la pantalla con resoluciones inferiores a la que tienes actualmente en tu ordenador.

7. Quita la opción ***Guías de resolución*** *(pulsando otro clic en esta opción) y activa* ***Centro de la pantalla*** *dentro de* ***Posición inicial****.*

Con esta otra opción lo que conseguirás es que el formulario siempre que se ejecute aparezca en el centro de la pantalla del usuario. En nuestra aplicación dejaremos activada esta opción.

Estas mismas opciones las podemos hacer desde la ventana de propiedades dentro de **StartUpPosition** con 4 opciones diferentes. **Manual**; **centrado dentro de un formulario padre** (está opción la explicaremos en futuras lecciones); **centrado en la pantalla** o **predefinido por Windows** (Esquina superior izquierda de la pantalla). Si te fijas son las mismas opciones que aparecen dentro del menú contextual al que hemos hecho referencia anteriormente.

Nosotros también podemos modificar la situación del formulario con respecto a los bordes interiores de la pantalla con las propiedades **Top** y **Left**. **Top** nos marca la distancia que existe entre la parte superior del monitor con la parte superior de nuestro formulario, mientras que **Left** nos marca la distancia entre la parte izquierda del monitor y la izquierda de nuestro formulario.

Si te molesta la ventana **Posición del formulario** la puedes cerrar.

Estilo del borde

Con el estilo del borde, **BorderStyle**, lo que podemos conseguir es hacer, por ejemplo, que nuestra aplicación no tenga ningún tipo de borde, que no se pueda cambiar su tamaño, que el tamaño lo podamos variar como a nosotros nos apetezca,...

En nuestro caso nos interesa que no se pueda modificar el tamaño del formulario ya que al hacer más pequeño el formulario se podrían ocultar botones y no podríamos utilizar la aplicación correctamente. Lo que si permitiremos es que el usuario pueda minimizar la aplicación, pero no la pueda maximizar.

Dentro de **BorderStyle** tenemos 6 posibles opciones.
0 - None: Hace que en nuestra aplicación no aparezcan bordes.
1 - Fixed Single: Hace que el borde de la aplicación siempre quede fijo. Con

esta opción podremos poner los botones minimizar o maximizar según nos convenga.

2 - Sizable: Esta opción es la que aparece por defecto al iniciar un nuevo formulario. Con esta opción podemos cambiar el tamaño del formulario a nuestro gusto.

3 – Fixed Double: Con esta opción podemos incluir el menú de control, la barra de título, pero no podemos incluir ni los botones maximizar ni minimizar. Esta ventana no podrá cambiarse de tamaño.

4 – Fixed Tool Window: Si activamos esta opción nos mostrará un formulario con la fuente del título reducida. No podremos modificar el tamaño del formulario. Este no aparecerá en la barra de tareas de Windows.

5 – Sizable Tool Window: Tendremos una ventana de tamaño ajustable. El tamaño de la fuente del título aparecerá reducida. El formulario no aparecerá en la barra de tareas.

Una cosa que hay que tener en cuenta es que estas opciones se ponen en funcionamiento en el momento que ejecutamos la aplicación. Otra cosa a tener en cuenta es que el menú de control que aparece sobre el icono de la aplicación también se modificará según las opciones de **BorderStyle** que hemos seleccionado y los botones de minimizar y maximizar que tengamos activados.

*8. Coloca la propiedad **BorderStyle** de nuestro formulario a **1 - Fixed Single**.*

Observa como los botones maximizar y minimizar han desaparecido de nuestro formulario, solo queda visible el botón cerrar.

9. Inicia una ejecución de prueba e intenta modificar el tamaño del formulario. Cuando termines detén la ejecución.

Vamos a colocar el botón **minimizar** para que el usuario pueda minimizar el formulario cuando le apetezca. Aunque esté esta opción activada el formulario seguirá sin "dejarse" cambiar el tamaño.

*10. Sitúate sobre la propiedad **MinButton**.*

Observa como esta propiedad tiene como valor **False**. Esto nos indica que el botón minimizar no está activado.

*11. Haz doble clic sobre la palabra **MinButton** y observa como su valor cambia a **True**.*

De esta manera hemos hecho que en nuestro formulario aparezca el botón minimizar. Observa como ha aparecido también el botón maximizar pero este no está activado. Para activarlo tendríamos que poner a **True** la propiedad **MaxButton**. En nuestro ejemplo no lo vamos a activar ya que no nos interesa que el usuario pueda maximizar nuestra aplicación.

Si queremos que el usuario no pueda mover por la pantalla la aplicación tendríamos que poner la propiedad **Moveable** a **False**. No es muy recomendado utilizar esta opción, excepto en casos muy específicos, ya que tenemos que dejar que el usuario pueda mover las aplicaciones por la pantalla para así poder visualizar el contenido de otras aplicaciones que están por detrás de esta.

Apariencia del formulario

Vamos a cambiar el texto que aparece en el título del formulario. Recuerda como se hace según lo explicado en la primera lección.

*12. Escribe **Pequeña calculadora** como título de nuestra aplicación.*

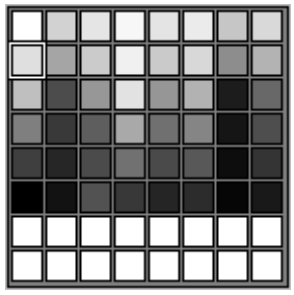
Ahora cambiaremos el icono que aparece en nuestra aplicación.

*13. Coloca como icono de la aplicación **Misc18.ico** que se encuentra dentro del direc-*

*torio **Graphics\Icons\Misc** dentro del directorio donde tengas instalado **Visual Basic**.*

Ahora vamos a cambiar el color de fondo de nuestra **Pequeña calculadora**.

*14. Accede a la propiedad **BackColor** y haz clic en la flecha para que se despliegue el menú de colores.*



Observa que aparecen dos carpetas. Una llamada **Sistema**, donde aparecen los colores de todos los objetos que vienen determinados por Windows y **Paleta** donde nos aparecen diversos colores para elegir. La **Paleta** es parecida a la que mostramos en la derecha.

Si haces clic con el botón izquierdo sobre uno de los cuadrados blancos inferiores te aparecerá una pantalla, en la que podrás elegir un color entre todos los disponibles dentro de la paleta de colores de Windows.

15. Haz clic sobre el color que desees para el fondo de nuestra aplicación. Yo he seleccionado el gris claro. (Segunda fila, primera columna).

No pongas colores que cansen mucho a la vista ya que debemos pensar que nuestras aplicaciones puede ser utilizada por usuarios durante mucho rato con lo que le puede producir sensación de cansancio.

Añadir objetos al formulario

Vamos a situar en nuestro formulario los elementos que ya conocemos: **Label**, **TextBox** y **CommandButton**. Para ello os mostraremos una lista en la que aparecerá el **tipo** de elemento que deberéis añadir en nuestro formulario, el **texto** que debe aparecer, cual debe ser su **nombre** (en caso de necesitarlo), su **tamaño** y por último su **posición**.

Más adelante modificaremos la apariencia de los objetos que añadiremos ahora.

Repasa la primera lección cuando se explica como se añaden objetos nuevos, como se cambian de tamaño y como se sitúan en una posición determinada dentro del formulario.

*16. Deberás añadir 6 objetos **Label**.*

Será mejor que los vayas añadiendo y modificando de uno en uno.

Modifica las propiedades de cada **Label** para que queden de la siguiente forma:

Label1
Caption: Calculadora
Left: 1920
Top: 0

Label2
Caption: Primer operando
Left: 240
Top: 1320

Label3
Caption: Segundo operando
Left: 2280
Top: 1320

Label4
Caption: Resultado
Left: 4680
Top: 1320

Label5
Caption: Lista de operaciones
Left: 480
Top: 3360

Label6
Caption: Operaciones con la lista
Left: 3360
Top: 4560

Label7
Caption: 0
(Nombre): MostrarResultado
Left: 4560
Top: 1560
BorderStyle: 1 - Fixed Single

Observa que a los 6 primeros **Label** que hemos añadido a nuestro formulario, no le hemos puesto **(Nombre)** a ninguno. Esto es debido a que durante la ejecución de esta aplicación estos objetos no deberán sufrir ningún tipo de modificación con lo que el nombre no nos interesa.

En cambio, el **Label7** será donde nos aparecerá el resultado de la operación que deseamos realizar.

*17. Añade 2 objetos **CommandButton**.*

Modifica las propiedades de cada **CommandButton** para que queden de la siguiente forma:

Command1
Caption: Borrar
(Nombre): Borrar
Left: 3000
Top: 2400
Width: 1215
Height: 495

Command2
Caption: Calcular

(Nombre): Calcular
Left: 4320
Top: 2400
Width: 1215
Height: 495

Recuerda como activar las teclas del modo abreviado de cada **Command**. Ejemplo: Botón **Calcular Alt+C**.

*18. Añade 2 objetos **TextBox.***

Modifica las propiedades de cada **TextBox** para que queden de la siguiente forma:

Text1
Text: (Borra el texto actual)
(Nombre): PrimerOperando
Left: 240
Top: 1560

Text2
Text: (Borra el texto actual)
(Nombre): SegundoOperando
Left: 2400
Top: 1560

Observa que en muchos de los objetos que hemos añadido hasta el momento no hemos especificado el tamaño, esto lo haremos más adelante cuando modifiquemos otras nuevas propiedades de estos objetos.

Modificar propiedades de varios objetos simultáneamente

Vamos ha modificar una propiedad que tendrán en común varios objetos.

Si varios objetos que tenemos en nuestro formulario cumplen una misma propiedad podemos hacer dos cosas: podríamos ir seleccionando objeto a objeto y modificar la propiedad en cada uno de ellos o seleccionarlos todos y modificar de una sola vez la propiedad con lo que quedarían todos los objetos modificados.

*19. Haz un clic sobre **Calculadora**.*

*20. Pulsa la tecla **Control** y mientras la mantienes pulsada haz clic en **Primer operando**, **Segundo operando**, **Resultado**, **Lista de operaciones** y **Operaciones con la lista**.*

Observa como han quedado seleccionados todos los elementos que hemos marcado. Observa también como la lista de propiedades ha cambiado, solo se muestran las propiedades que podemos cambiar de forma conjunta a todos los objetos seleccionados.

Si te fijas en los objetos seleccionados podrás observar que tienen un fondo de color gris oscuro que delimita su tamaño. (Esto solo lo podrás ver si el color que escogiste para el formulario es diferente a este gris). Lo que vamos ha hacer es ver una nueva propiedad que nos hará que estos objetos sean transparentes, de esta manera conseguiremos que solo se vea el texto y no el tamaño de dicho objeto.

*21. Pulsa **F4** para acceder a las propiedades.*

*22. Haz un doble clic sobre la propiedad **BackStyle** verás como todos los objetos selec-*

*cionados pasan de ser **opacos** a **transparentes**.*

Fuentes de letra en modo edición.

La gran mayoría de los objetos que podemos añadir a un formulario contienen texto. Este texto también puede modificarse para hacer más vistosa o más clara nuestra aplicación. El formato de texto se puede cambiar desde el modo diseño o desde el modo de ejecución (como ya vimos en la lección anterior). En este capítulo explicaremos con más detenimiento ambos sistemas.

*23. Si todavía mantienes seleccionados los objetos que hemos seleccionado en los anteriores puntos sólo debes hacer un clic, manteniendo pulsada la tecla **Control**, sobre **Calculadora**, para quitar la selección de este objeto. Si no mantienes la selección, vuelve a seleccionar los objetos que antes teníamos seleccionados pero esta vez sin el texto **Calculadora**.*

Esto lo hemos hecho porque todos los objetos que están seleccionados tienen el mismo formato de letra mientras que el título **Calculadora** tiene otro formato.

*24. Pulsa **F4**.*

*25. Accede a la propiedad **Font**.*

Observa que esta propiedad está vacía. Esto siempre ocurre en el momento en el que tenemos diferentes objetos seleccionados.

26. Pulsa un clic sobre el botón con tres puntos suspensivos que aparece en dicha propiedad.

Acto seguido aparece un cuadro de diálogo como este:

Dentro de la lista **Fuente** podremos seleccionar uno de los tipos de letras que tenemos instalado en nuestro ordenador. En el apartado **Estilo de fuente** podremos seleccionar entre cuatro opciones **Normal** (ejemplo), **Cursiva** (*ejemplo),* **Negrita** (**ejemplo**), **Negrita cursiva** (***ejemplo***). Juntamente con el estilo seleccionado podemos aplicar dos **Efectos** diferentes como es: **Tachado** (~~ejemplo~~) o **Subrayado** (ejemplo). También podemos hacer una mezcla de los diferentes formatos de letra para así poder obtener algo así (***~~ejemplo~~***): negrita cursiva con subrayado y tachado. También podremos modificar el **Tamaño** de la fuente seleccionada. Deberemos tener cuidado con esta propiedad ya que según el tamaño que seleccionemos podría ser que no se viera completamente el contenido de la información que deseamos mostrar.

*27. Haz un clic en **Negrita**. Acepta la ventana.*

Observa los cambios. Las demás opciones las dejaremos como están. Si no ves todo el contenido de estos elementos, no pasa nada.

28. Haz un clic en cualquier parte de la pantalla para quitar la selección.

*29. Haz un clic sobre **Calculadora**.*

*30. Accede a la propiedad **Font**.*

*31. Accede al cuadro de diálogo **Fuente**.*

*32. Modifica el **tamaño** a **18** y haz que aparezca **Subrayado**.*

Antes de aceptar la ventana observa el recuadro de **Ejemplo**. En este recuadro podrás ver una simulación de cómo quedarán las modificaciones que has hecho.

*33. **Acepta** el cuadro de diálogo.*

*34. Selecciona el **TextBox** que lleva como **(Nombre) PrimerOperando** y **SegundoOperando**, junto con el **Label** llamado **MostrarResultado**.*

*35. Accede al cuadro de diálogo **Fuente**.*

*36. Cambia el **Tamaño** a **18**. Acepta el cuadro de diálogo.*

Modificar tamaños

Vamos a modificar el tamaño de estos 3 últimos objetos modificados.

*37. Modifica el tamaño a **1215 x 540***

Fuentes de letra en modo ejecución

Como ya vimos en la lección anterior los estilos de fuente se pueden modificar mientras estamos ejecutando el programa. Esto se consigue modificando las propiedades de **estilo de fuente** de alguno de los objetos insertados en nuestro formulario.

Vamos a imaginarnos que tenemos un objeto llamado **Texto** en nuestro formulario de trabajo al cual le modificaremos los estilos de fuente.

Para modificar un estilo de fuente como puede ser **negrita**, **cursiva**, **tachado** y **subrayado** utilizamos unas propiedades de tipo **booleano**[1]. Su sintaxis es exactamente igual que en el caso de cualquier otra propiedad. Deberemos escribir el nombre del objeto que queremos modificar seguido de un punto y una de estas cuatro propiedades: **FontBold** (**Negrita**), **FontItalic** (*Itálica*), **FontStrikethru** (~~Tachado~~) o **FontUnderline** (Subrayado), después el signo igual (=) y el valor **True** o **False** según nos interese activarlo o desactivarlo. (También podríamos poner **Font.Bold**, **Font.Italic**, **Font.Strikethru** o **Font.Underline**).

Por ejemplo, imaginemos que tenemos un **botón** que al pulsarlo queremos que el objeto **Texto** cambie a **negrita**. Dentro del objeto **botón** escribiremos la siguiente línea de código **Texto.FontBold = True** esto hará que el **Texto** aparezca en **negrita**. Si ya está en **negrita** no ocurrirá nada. Si queremos que aparezca el texto "normal" podríamos poner en otro **botón** la línea **Texto.FontBold = False**, de esta manera tendremos un **botón** que activa la negrita y otro que la desactiva. Esto funciona exactamente igual para cualquiera de las otras propiedades.

Si te fijas en este caso tenemos que diseñar dos botones para activar y desactivar la negrita, pero podemos hacer que un mismo botón haga las dos cosas, o cualquier otra propiedad, según la que exista en este momento. Lo explicaremos de otra forma; si el texto está en negrita se desactivará la negrita y si el texto no está en negrita se activará la negrita. Esto se consigue con la siguiente línea**: Texto.FontBold = Not Texto.FontBold**. La partícula **Not** hace que la propiedad se alterne, si está en **False** se convierte en lo contrario **True** y si su valor es **True** se convierte en **False**.

También podemos cambiar el tipo de fuente, esto lo haremos con la propiedad **FontName**. Esta propiedad no es de tipo **Booleana** ya que tenemos que especificar el nombre de la fuente que queremos insertar. La sintaxis sería de la siguiente forma: Nombre del objeto seguido de un punto, la propiedad **FontName**, un igual y entre comillas dobles el nombre de la fuente. Por ejemplo: **Texto.FontName = «Verdana»**.

Otra propiedad que tenemos para cambiar nuestro estilo de fuente es: **FontSize**, con esta propiedad lo que conseguimos es modificar el tamaño de la fuente. Esta propiedad tampoco es de tipo **booleana** ya que deberemos especificar el tamaño de la fuente. El tamaño se expresa en puntos. El tamaño máximo es de **2160 puntos**. Los puntos son de tipo numérico con lo que la sintaxis sería de la siguiente manera: Nombre del objeto seguido de un punto, la propiedad **FontSize**, un igual y el número que

indicará el tamaño de la fuente de letra. Por ejemplo: **Texto.FontSize = 12**.

Como practica adicional puedes crear un nuevo formulario para practicar estas nuevas propiedades.

Tamaño automático

Ahora vamos a modificar el tamaño de los cuadros de texto que tenemos en nuestro formulario.

Emplearemos otra nueva propiedad de estos objetos que es el ajuste automático del tamaño con respecto al texto que hay en su interior.

*38. Selecciona todos los elementos de texto que tenemos hasta el momento, menos el que tiene como **(Nombre) MostrarResultado**.*

*39. Accede a las propiedades y cambia a **True** la propiedad **AutoSize**.*

Observa como los puntos de selección de cada uno de los objetos se ha aproximado hasta el texto. Si nosotros ahora modificásemos la propiedad **Caption** veríamos como el tamaño del objeto cambia según el tamaño del texto que hay dentro de dicho objeto.

Alineación del texto

En nuestra práctica nosotros vamos a trabajar con diferentes números que iremos introduciendo en las casillas de primer y segundo operando para obtener un resultado.

Si nosotros utilizamos casillas de texto o etiquetas para que el usuario introduzca o visualice texto, normalmente se alinea a la izquierda (ya que es por donde comenzamos a escribir texto) y si trabajamos con números los alienamos a al derecha (para que todas las comas decimales en los números enteros estén juntas).

*40. Selecciona solo **PrimerOperando** y accede a la propiedad **Text**.*

*41. Escribe la palabra **Texto**.*

Esto lo hemos hecho para poder explicar mejor como actúa la alineación del texto en los diferentes objetos.

Observa como en este objeto al igual que en el **Label MostrarResultado** el texto está a la izquierda.

*42. Accede a la propiedad **Alignment** de **MostrarResultado**.*

Observa que tienes 3 posibles opciones. Esto lo podrás ver si despliegas la lista de esta propiedad. **0: izquierda, 1: derecha, 2: centro**.

*43. Selecciona la alineación a la derecha (**1.- Right Justify**).*

Observa nuestro formulario y donde está alineado el texto de este objeto.

Vamos ha hacer lo mismo con los objetos: **PrimerOperando** y **SegundoOperando**. Si quieres ver mejor los cambios y para asegurarte que lo haces correctamente puedes poner algo en la propiedad **Text** de **SegundoOperando**.

*44. Selecciona **PrimerOperando** y **SegundoOperando** para trabajar con ambos objetos conjuntamente.*

*45. Accede a la propiedad **Alignment** y selecciona la opción correspondiente, para hacer que el texto de estos objetos aparezca alineado a la derecha.*

Observa como en los dos objetos de tipo texto que tenemos seleccionados no ha ocurrido absolutamente nada. ¿A que es debido este comportamiento? Muy sencillo, si queremos que esta propiedad "funcione" tenemos que activar otra propiedad.

*46. Accede a la propiedad **MultiLine** y ponla en **True**.*

Observa que inmediatamente después de cambiar está opción el texto pasa a estar alineado a la derecha.

La propiedad **MultiLine** lo que está haciendo es definir que en los dos objetos texto se puedan introducir varias líneas. Ten en cuenta que siempre que quieras una alineación a derecha o centro en objetos **Text** deberás activar la propiedad **MultiLine**.

47. Borra el contenido de los dos objetos seleccionados.

Observa que no podrás modificar el contenido de los objetos mientras estén los dos seleccionados.

Al acceder a la propiedad **Text** verás que hay la palabra **(Texto)** esto nos indica que **MultiLine** está activado y por lo tanto puede ser que dentro de este objeto puedan existir múltiples líneas de texto. Para eliminar lo que ya tenemos debemos pulsar en el botón con una flecha hacia abajo que aparece en esta propiedad y borrar el contenido.

Delimitación de tamaño

Ahora vamos a delimitar el tamaño de los números que podemos introducir en **PrimerOperando** y **SegundoOperando**. Esto lo conseguiremos con la propiedad **MaxLength**. Esta propiedad hará que no podamos introducir números con una cantidad de caracteres superiores a la que nosotros indiquemos. **Visual Basic** no nos dejará introducir más caracteres. No nos avisará de ninguna manera, simplemente no nos dejará introducir ningún carácter más.

*48. Selecciona **PrimerOperando** y **SegundoOperando**.*

*49. Pulsa **F4**, para acceder a las propiedades.*

*50. Escribe **4** en **MaxLength**.*

Texto de ayuda

Existe una propiedad, en la mayoría de los objetos que podemos añadir en nuestro formulario, que sirve para mostrar **ayuda** rápida al mantener el puntero del ratón durante unos segundos sobre el objeto deseado. Este texto suele ser corto y explícito dando una idea de para que sirve dicho control.

*51. Selecciona **PrimerOperando**.*

52. Accede a sus propiedades.

*53. Sitúate sobre la propiedad: **ToolTipText**.*

En esta propiedad podemos escribir lo que queremos que aparezca en el pequeño cuadro de ayuda al mantener el ratón durante unos segundos en el objeto seleccionado.

*54. Escribe: **Introduce el primer operando**.*

Realiza estas mismas operaciones con **SegundoOperando** y los botones **Calcular** y **Borrar**. Escribe el texto que creas conveniente, pensando que con el botón **Calcular** se realizarán los cálculos pertinentes según la operación seleccionada (opciones que veremos en la siguiente lección) y el botón **Borrar** borra el contenido de **PrimerOperando**, **SegudoOperando** y **MostrarResultado**, para poder iniciar una nueva operación con diferentes operandos.

OptionButton en modo gráfico

Vanos a insertar unos controles que nos servirán para poder seleccionar cual de las cuatro operaciones (suma, resta, multiplicación o división) es la que deseamos realizar. Hemos escogido este elemento ya que solo podremos marcar uno de ellos a la vez.

En la primera lección ya utilizamos este tipo de objeto, pero aquí vamos a ver una nueva propiedad de este, ya que no trabajaremos con él con la apariencia que lo hicimos en la pasada lección, sino que tendrá apariencia de botón, pero con una imagen en su interior.

*55. Inserta un **OptionButton**.*

Observa como es su apariencia.

*56. Ponle como **(Nombre)**: **Sumar**.*

*57. Accede a la propiedad **Style** y modifica su valor de **Standard** a **Graphical**.*

Observa como su apariencia ahora es como un botón.

*58. Borra el contenido de la propiedad **Caption**.*

*59. Accede a la propiedad **Picture** y selecciona **Misc18.ico** de **Graphics\Icons\Misc** dentro del directorio donde tengas instalado **Visual Basic**.*

*60. Cambia el tamaño a **540 x 540** y su posición a **1680, 600**.*

*61. Inserta 3 **OptionButton** más.*

63. Modifica sus propiedades para que queden de la siguiente manera:

Option1
Caption: (Borra su contenido)
(Nombre): Restar
Posición: 1680, 1200
Tamaño: 540 x 540
Style: Graphical
Picture: Misc19.ico

Option2
Caption: (Borra su contenido)
(Nombre): Multiplicar
Posición: 1680, 1800
Tamaño: 540 x 540
Style: Graphical
Picture: Misc20.ico

Option3
Caption: (Borra su contenido)
(Nombre): Dividir
Posición: 1680, 2400

Tamaño: 540 x 540
Style: Graphical
Picture: Misc21.ico

*64. Asegúrate que la propiedad **Value** de objeto **Sumar** está en **True**.*

*65. Escribe en la propiedad **ToolTipText** de cada uno de estos objetos algo que le pueda servir de ayuda a los usuarios de esta aplicación, tal como vimos en puntos anteriores.*

66. Realiza una ejecución de prueba. Selecciona las diferentes operaciones.

Observa que cuando se selecciona una, se quita la selección la que estaba seleccionada y así sucesivamente.

Creación de archivos ejecutables

Con **Visual Basic** podemos crear archivos ejecutables, con extensión (.exe) de una forma fácil y sencilla.

Al convertir una aplicación en un archivo ejecutable podremos poner en funcionamiento dicha aplicación sin necesidad de ejecutar **Visual Basic**. Pero deberemos pensar que esta aplicación no podrá funcionar en cualquier otro ordenador, solamente funcionará en los aquellos en los que esté instalado **Visual Basic**. Para que funcione en cualquier otro ordenador deberemos hacer un archivo de instalación, más adelante veremos como hacerlo.

Antes de crear el archivo ejecutable lo que tendremos que hacer es guardar el formulario y el proyecto con el nombre que quieras dentro del directorio que quieras.

Antes de continuar, decir que la aplicación no está finalizada pero ya podemos crear el archivo ejecutable para tener una muestra.

Otra cosa muy importante es ir grabando el formulario y el proyecto a medida que vamos haciendo cambios para hacer copias de seguridad. Esto lo podemos hacer de una forma fácil, una vez grabados ya el formulario y el proyecto con un nombre, deberemos pulsar el botón con el dibujo de un disco en la barra de herramientas estándar.

*67. Accede a **Generar (nombre del archivo) .exe...** dentro del menú **Archivo**.*

En **(nombre del archivo)** te aparecerá el nombre del proyecto que hayas puesto anteriormente.

Seguidamente te aparecerá una ventana llamada **Generar proyecto**. En esta ventana deberemos indicar la carpeta y el nombre del archivo ejecutable.

*68. Deja el nombre que aparece en la ventana **Generar proyecto**, antes de pulsar **Aceptar** haz clic en el botón **Opciones**.*

Aparecerá un nuevo cuadro de diálogo.

En este cuadro podemos modificar el número de versión de nuestro archivo ejecutable de forma manual, indicando el **número de versión** en **principal**, **secundario** y **revisión**. También lo podemos hacer de forma automática pulsando un clic en **Incremento automático**. Desde este cuadro de diálogo podemos modificar tanto el **título** de nuestra aplicación, como el **icono**. Podemos añadir información adicional como **el nombre de la compañía**, **nombre del producto**, **copyright**, etc.

*69. Cierra el cuadro **Propiedades del proyecto**. Haciendo clic en **Cancelar** o en **Aceptar** si has modificado alguna opción.*

*70. Haz clic en **Aceptar**, para crear el archivo ejecutable.*

Observa como en la barra de herramientas estándar aparece una banda de color que va incrementando con la palabra **Compilando...** y **Escribiendo Exe...** Esto nos indica que **Visual Basic** antes de crear el ejecutable compila el proyecto para revisar errores y después graba el archivo con extensión **Exe**.

Para ejecutar el archivo **Exe** que acabamos de crear puedes hacerlo como cualquier otro programa que tengas instalado en tu ordenador.

*71. Abre el **Explorador de Windows**, accede al directorio donde está el archivo ejecutable.*

Observa como este archivo tiene como icono el mismo que pusimos en el formulario.

72. Haz doble clic para ejecutarlo.

Observa como aparentemente la ejecución es igual que si la hubiéramos hecho desde **Visual Basic**. Podrás observar que en según que tipo de aplicación será mucho más rápido ejecutar el archivo **Exe** que ejecutarlo desde **Visual Basic**.

Vamos a explicar como podemos hacer un archivo de instalación. Recuerda que la aplicación no está finalizada. Este paso normalmente se hace en el momento en el que ya está la aplicación completamente terminada.

Archivo de instalación

Para realizar estos pasos deberemos poner en funcionamiento uno de los módulos que vienen junto a **Visual Basic**.

*73. Cierra **Visual Basic**.*

Si al realizar este paso, **Visual Basic** te pide si deseas guardar la aplicación contesta afirmativamente.

*74. Accede a **Inicio – Programas - Microsoft Visual Studio 6.0 - Herramientas de Microsoft Visual Studio 6.0 - Asistente para empaquetado y distribución**.*

Seguidamente te aparecerá una pantalla como la siguiente:

*75. Pulsa el botón **Examinar...***

En la ventana **Abrir proyecto** escoge el proyecto con el que hemos estado trabajando anteriormente.

*76. Cuando lo tengas seleccionado pulsa el botón **Abrir**.*

Observa como en la lista desplegable de la parte superior de la ventana **Asistente de empaquetado y distribución** aparece la carpeta y el nombre del archivo con el que deseas trabajar.

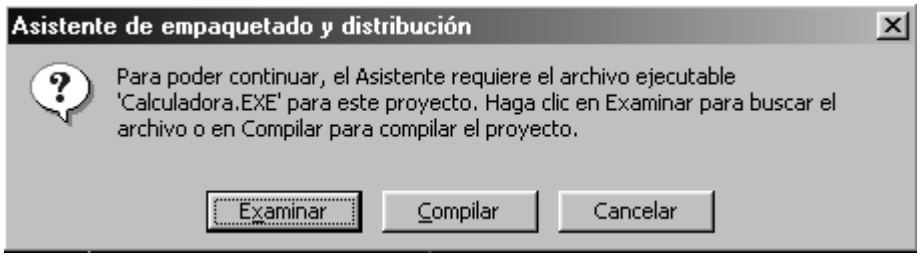
*77. Pulsa el botón **Empaquetar**.*

En este momento **Visual Basic** comprobará si tenemos realizado el archivo ejecutable (exe), si no lo hemos creado anteriormente nos aparecerá una ventana parecida a la siguiente:

En la que deberemos seleccionar la opción **Compilar**. De esta forma **Visual Basic** generará el archivo ejecutable.

*78. Pulsa en botón **Compilar**.*

En caso de ya estar creado dicho archivo no nos aparecerá esta ventana.



Seguidamente nos aparecerá una nueva ventana como esta:

*79. Selecciona la opción **Paquete de instalación estándar** y pulsa en **Siguiente**.*

Al elegir esta opción estamos indicando que se genere un archivo de instalación para nuestra aplicación.

Después de unos segundos nos aparecerá la siguiente pantalla:

En ella deberemos indicar en que carpeta deseamos guardar los archivos que se generarán para realizar la instalación de nuestra aplicación. Observa que él nos propone utilizar una carpeta nueva llamada **Paquete** que cuelga de la carpeta en la que tenemos nuestro proyecto.

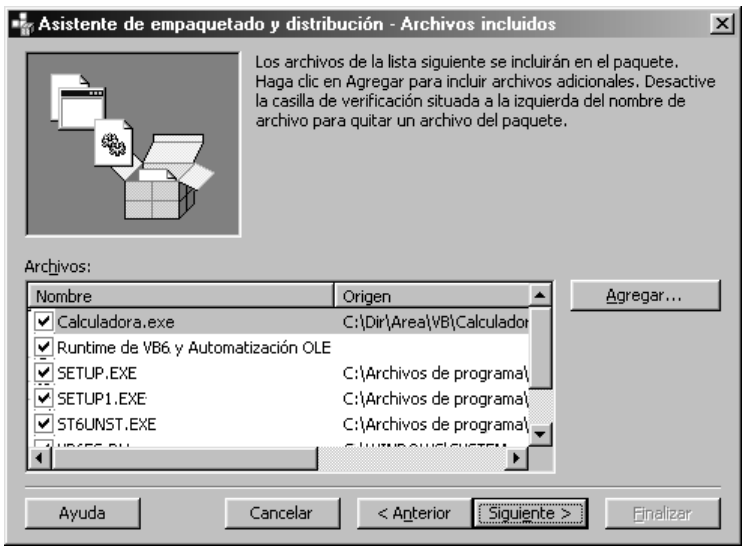
*80. Vamos a dejar esta carpeta como buena. Pulsa en **Siguiente**.*

Si el asistente ve que la carpeta no existe nos mostrará un mensaje para crearla.

81. Responde afirmativamente a este mensaje.

Seguidamente nos aparecerá una nueva ventana en la que se nos mostrarán todos los archivos necesarios para que funcione perfectamente nuestra aplicación cuando se instale en cualquier ordenador. Nosotros podremos indicar que no se instale un archivo de los que aparecen en la lista, el único problema que puede haber es que según el archivo que quitemos no funcione correctamente la aplicación en cualquier ordenador. También podemos hacer que se guarde algún otro archivo junto con la instalación. Para ello deberíamos pulsar sobre el botón **Agregar...**

La pantalla tiene un aspecto parecido a este:



Vamos a dejar seleccionados todos los archivos y no vamos a **Agregar** ningún archivo más.

*82. Pulsa en **Siguiente**.*

Seguidamente nos aparecerá una pantalla como la siguiente:

En esta pantalla podremos indicar si queremos que se genere un único archivo **.cab** o varios archivos más pequeños **.cab** para así poderlos guardar en discos. Los archivos **.cab** son los que contienen toda la información y todos los archivos que se descomprimen durante la instalación.

En nuestro caso vamos a generar un sólo archivo **.cab**

*83. Selecciona la opción: **Un único archivo .cab** y pulsa en **Siguiente**.*

En la siguiente pantalla que nos aparece de todo este proceso deberemos indi-

car el título que deseamos que aparezca en el momento en el que se realiza la instalación.

84. Escribe: ***Mi primera aplicación*** *y pulsa en* ***Siguiente****.*

En esta nueva ventana:

Podremos indicar en que grupo del menú **Inicio** deseamos que se guarde el acceso directo hacia nuestra aplicación.

Vamos a cambiar el nombre del grupo en el que estará situado el acceso directo a nuestra aplicación.

85. Pulsa sobre el grupo ***Mi primera aplicación****.*

86. Haz clic sobre el botón ***Propiedades...***

87. En la ventana que te aparecerá a continuación cambia el nombre y escribe: ***Mis aplicaciones****.*

88. Pulsa sobre el botón ***Aceptar****.*

Una vez cambiado el nombre del grupo ya podemos seguir con nuestro asistente.

89. Pulsa sobre el botón ***Siguiente****.*

En la siguiente ventana podríamos modificar en que carpetas queremos que se guarde la aplicación en el momento que se instale.

Vamos a dejar la configuración original y así permitiremos que sea el usuario

quien decida en que carpeta desea guardar la aplicación en el momento en el que se realice la instalación.

*90. Pulsa sobre el botón **Siguiente**.*

Seguidamente nos aparecerá una pantalla en la que podremos indicar si el archivo ejecutable que se generará después de realizar la instalación deseamos que esté compartido o no. El estar compartido quiere decir si este programa formará parte de otros o solamente será este el que lo utilice. Así en el momento de desinstalar la aplicación no se borrará dicho archivo para que pueda ser utilizado por otros programas.

*91. No selecciones este archivo y pulsa sobre el botón **Siguiente**.*

En la ventana que nos aparece a continuación podemos indicar un nombre para guardar gran parte de los pasos que hemos realizado durante este asistente para que así en futuras ocasiones podamos realizar dichos pasos automáticamente.

*92. Deja el nombre que aparece por defecto y pulsa sobre el botón **Finalizar**.*

Este es el final del proceso. Ahora se generarán todos los archivos necesarios para realizar la instalación de la aplicación.

Al final del proceso nos aparecerá un cuadro con un pequeño informe del proceso realizado.

*93. Pulsa en el botón **Cerrar**.*

*94. Por último cierra el **Asistente de empaquetado y distribución**.*

Ahora ya tienes la aplicación preparada para grabársela a quien desees.

*95. Abre el explorador, accede a la carpeta donde has guardado los archivos de instalación y ejecuta el archivo **Setup.exe**.*

96. Sigue los pasos, así podrás observar como se realiza la instalación de la aplicación.

La aplicación ya tiene un aspecto mucho más serio y profesional.

Fin lección 2

[1] Booleano: solo puede tomar dos valores, Verdadero o Falso.

Lección 3

Esta lección la vamos a dedicar a colocar en nuestro proyecto nuevos objetos no vistos hasta el momento. Una vez terminada toda la presentación pasaremos a depurar nuestra aplicación para que no se produzcan errores inesperados e introducir mensajes que aparecerán en determinados momentos para que el usuario sepa que es lo que debe hacer. En esta lección también introduciremos líneas de código que iremos comentando, aunque las principales instrucciones de **Visual Basic** las veremos en siguientes lecciones.

Controles de imagen

En nuestras aplicaciones es interesante insertar imágenes para mostrar algún tipo de información adicional o para facilitar el uso de esta a los usuarios.

En un principio tenemos **4** controles que nos permiten trabajar con gráficos: cuadro de imagen (**PictureBox**), imagen (**Image**), forma (**Shape**) y línea (**Line**).

Cada uno de ellos lo utilizaremos en casos diferentes, según nos interese una u otra característica de cada objeto.

PictureBox

El cuadro de imagen (**PictureBox**) lo podemos utilizar para mostrar imágenes sueltas, aunque suele utilizarse como contenedor de otros elementos. Con esto queremos decir que dentro de un cuadro de imagen pueden existir otros elementos que dependen de él. Si nosotros movemos el cuadro de imagen con elementos en el interior, todos ellos se moverán junto con él. Si mirásemos la propiedad **Top** y **Left** de cualquier elemento que está insertado dentro de un cuadro de imagen veríamos que están en relación con el borde de este y no con el borde del formulario como en la gran parte de los objetos.

Image

El control imagen (**Image**) solo lo utilizaremos para mostrar imágenes en su interior. No se puede utilizar como contenedor como en el caso del **PictureBox**.

En nuestra práctica vamos a insertar una imagen en la que mostraremos un icono que tendrá como dibujo el signo igual (**=**).

. Practica 1 (continuación de la lección anterior)

1. Abre el proyecto con el que estuvimos trabajando en la lección anterior.

Recuerda lo que teníamos hecho hasta este momento.

Vamos a inserta un control **Image**, para poder poner una imagen con un signo igual (**=**) en su interior.

*2. Haz doble clic en el control **Image**.*

Observa como en el centro de nuestro proyecto ha aparecido un nuevo objeto con unas líneas discontinuas que lo delimitan.

*3. Accede a la propiedad: **Picture**.*

Observa que aparece la palabra **(Ninguno)**, esto nos indica que en este momento el objeto **Picture** no está mostrando ningún tipo de imagen.

4. Haz un clic sobre el botón con tres puntos suspensivos que aparece a la derecha de

esta propiedad.

Acto seguido verás como te aparece un cuadro de diálogo donde podrás seleccionar la imagen que quieres que se muestre. Observa que en el apartado: **Archivos de tipo** está escrita la frase **Todos los archivos de imágenes**, esto quiere decir que en este cuadro de diálogo nos aparecerán todos los archivos que tengan como extensión (**Bmp, dib, gif, jpg, wmf, emf, ico** y **cur**).

*5. Accede al directorio **Graphics\Icons\Misc** dentro del directorio donde tengas instalado **Visual Basic 6**, y selecciona el archivo **misc22.ico**.*

Observa como el tamaño de dicho objeto se ha modificado automáticamente. Observa también como en la propiedad **Picture** ahora aparece la palabra **(icono)** esto nos da a entender que el objeto que se está mostrando en este momento es un icono.

Cambio de tamaño de un objeto Image

Nosotros en este momento podemos cambiar un poco el aspecto de nuestro icono.

6. Sitúate en uno de los vértices de este objeto.

7. Arrastra hasta aumentar el tamaño del objeto.

Observa como el dibujo no ha sufrido ninguna modificación. Solo ha cambiado el tamaño del control, pero no el del dibujo.

*8. Accede a la propiedad **Stretch** y pon su valor a **True**.*

Observa como ahora el dibujo ocupa todo el tamaño del objeto. Puede ser que el dibujo se vea deformado.

Para hacer que el dibujo vuelva a su tamaño real, nada más fácil que realizar el siguiente paso.

*9. Pon la propiedad **Stretch** a **False**.*

Observa como tanto el tamaño del icono, como el del objeto han cambiado.

*10. Mueve el objeto **Image** hasta la posición: **3840, 1560**.*

Puedes hacerlo utilizando cualquiera de los métodos explicados hasta el momento.

Bordes en la imagen

Una vez colocado nuestro objeto **Image** en su sitio vamos a modificar su contorno.

*11. Accede a la propiedad **BorderStyle** y modifica el contenido de **None** a **Fixed Single**.*

Observa como ha aparecido un borde en **3D** que rodea a todo el objeto. Con la propiedad **Appearance** podrás hacer que este borde mantenga la apariencia de **3D** o solo sea un cuadro con una línea simple. Deja esta propiedad como está.

Apariencia del ratón

Vamos a modificar la apariencia del ratón cuando pase por encima de este obje-

to. Esto lo haremos para que los usuarios no piensen que se debe pulsar esta imagen para conseguir el resultado de la operación. En un principio si nosotros no ponemos ninguna línea de código dentro del evento **Click** de este objeto no debería pasar nada al pulsar un clic, pero puede ser que el usuario piense que la aplicación funciona incorrectamente por lo que seguiría intentándolo. En cambio nosotros podemos mostrar un icono que facilite la utilización de la aplicación.

*12. Accede a la propiedad **MousePointer**.*

Aquí especificaremos que tipo de cursor queremos que aparezca en el momento en el que el cursor pasa por encima del objeto.

Tenemos diferentes tipos de cursores:

Valor	Descripción
0	Predeterminado.
1	Flecha.
2	Cruz
3	Forma de I.
4	Pequeño cuadrado dentro de otro cuadrado.
5	Flecha de cuatro puntas.
6	Flecha doble que apunta al NE y al SE.
7	Flecha doble que apunta al N y al S.
8	Flecha doble que apunta al NO y al SE.
9	Flecha doble que apunta al O y al E.
10	Flecha hacia arriba.
11	Reloj de arena.
12	No colocar.
13	Flecha y reloj de arena.
14	Flecha y signo de interrogación.
15	Ajustar todo.
99	Icono especificado en la propiedad **MouseIcon**.

Esta tabla está extraída de la ayuda de **Visual Basic.** En ella se especifican las diferentes formas que puede tomar el cursor al pasar por encima del objeto. Para poder ver con más exactitud cada una de las formas es recomendable ir seleccionando cada una de ellas e iniciar una ejecución de prueba. Esta propiedad no es visible en el modo de diseño.

*13. En nuestro caso deberemos seleccionar: **12 - No Drop**.*

*14. Ejecuta la aplicación y coloca el cursor sobre el objeto **Image**.*

Controles de gráficos

Los controles gráficos **Line** y **Shape** son mucho más simples que los que hemos visto anteriormente, pero nos ayudan a diseñar nuestra aplicación.

Line

Elemento que nos dibuja una línea en nuestro formulario. Este elemento no contiene eventos, sólo se pueden utilizar de forma decorativa.

Nosotros podemos añadir una línea haciendo un doble clic sobre el objeto **Line** o haciendo, en primer lugar, un clic sobre el objeto **Line**, después marcando el primer punto de la línea, mantener pulsado el botón del ratón y soltarlo en el momento en el que queramos el punto final. Después para modificar el tamaño solo deberemos situarnos sobre una de las puntas de la línea, hacer clic con el botón derecho y mientras lo mantienes pulsado movernos hasta la nueva posición. Si lo que queremos es mover toda la línea, manteniendo la inclinación y el tamaño, arrastraremos la línea haciendo

clic en cualquier parte de ella. Otra manera que tenemos para mover los puntos iniciales y finales de la línea es utilizando las propiedades **X1**, **X2**, **Y1** y **Y2**.

X1 nos marca la distancia del primer punto con la parte izquierda del formulario. **X2** es igual que **X1** pero se refiere al segundo punto de la línea. **Y1** nos marca la distancia del primer punto con la parte superior del formulario. **Y2** es igual que **Y1** pero haciendo referencia al segundo punto de la línea.

Podemos utilizar la propiedad **Visible** para mostrar (**True**) u ocultar la línea (**False**). La propiedad **DrawMode** modifica la apariencia de la línea. **BorderWidth** nos modifica el ancho de la línea y con **BorderColor** podemos modificar el color de la línea.

Puedes insertar una línea en el formulario con el que estamos trabajando donde creas conveniente y modificarla a tu gusto.

Shape

Con este control podemos insertar en nuestro formulario un rectángulo, un cuadrado, una elipse, un círculo o un rectángulo o cuadrado con las esquinas redondeadas.

Al igual que en el caso del control **Line**, este elemento no contiene eventos, solo nos sirve para decorar nuestros formularios.

*15. Inserta un objeto **Shape** en nuestro formulario.*

*16. Muévelo hasta la posición **120, 480** y cámbiale el tamaño a **5775 x 2655**.*

Cuando nosotros hemos insertado este objeto hemos obtenido un rectángulo, pero como ya hemos dicho anteriormente, nosotros podemos obtener diferentes formas geométricas. Esto lo conseguiremos modificando la propiedad **Shape**. Tenemos **6** posibilidades: rectángulo (**Rectangle**), cuadrado (**Square**), elipse (**Oval**), círculo (**Circle**), rectángulo con las esquinas redondeadas (**RoundedRectangle**) o cuadrado con las esquinas redondeadas (**RoundedSquare**).

*17. Prueba cualquiera de las opciones de la propiedad **Shape**. Al final deja **Rectangle**.*

*18. Cambia la propiedad **BackStyle** de transparente a opaco. Observa que ha ocurrido.*

Cambiando la apariencia

El rectángulo que nosotros hemos insertado se ha rellenado de color blanco y muchos de los objetos que están dentro de él han desaparecido. Bien, si queremos que todos los objetos vuelvan a verse lo que tenemos que hacer es "empujarlo" hacia el fondo del formulario, para que los demás objetos pasen a estar por encima de él.

19. Sitúate sobre el borde del rectángulo.

20. Pulsa el botón derecho del ratón para que aparezca el menú contextual.

*21. Selecciona la opción **Enviar al fondo**.*

Observa como todos los elementos han aparecido nuevamente.

Con la propiedad **BackColor** se puede modificar el color del fondo del rectángulo, en cambio si lo que deseas es modificar el color del borde utiliza **BorderColor**.

A este objeto también le puedes añadir una trama en lugar de un color opaco.

Si modificas la propiedad **FillStyle** podrás obtener diferentes tipos de tramas: Trama sólida (**Solid**), transparente (**Transparent**), líneas horizontales (**Horizontal**

Line), líneas verticales (**Vertical Line**), líneas diagonales de izquierda a derecha (**Upward Diagonal**), líneas diagonales de derecha a izquierda (**Downward Diagonal**), en cruz (**Cross**) y líneas diagonales cruzadas (**Diagonal Cross**)

22. Coloca una de estas tramas.

*23. Modifica el color de la trama con la propiedad: **FillColor**.*

*24. Una vez visto los cambios quita la trama. Poniendo la propiedad **FillStyle** a **Transparent**.*

Inserta otro **Shape** en la posición **120**, **3240** con un tamaño **5775** x **2055**. Modifica todas las propiedades pertinentes para que queden como en el caso anterior.

Frame

A nosotros nos interesa poder poner dos nuevos **OptionButton** en nuestra aplicación, que utilizaremos para marcar si lo que queremos que aparezca en la lista, que más adelante añadiremos, es el resultado de la operación o toda la operación completa.

Como ya dijimos en lecciones anteriores en un mismo formulario pueden existir el número de **OptionButton** que deseemos, pero solo puede estar activado uno de ellos simultáneamente. En nuestra práctica esto no nos interesa ya que por un lado deberemos indicar una de las 4 operaciones que deseamos realizar y por otro que es lo que deseamos que se añada a la lista. Veamos esto en la práctica.

*25. Añade un nuevo **OptionButton**.*

26. Sitúalo en algún sitio del formulario que no te moleste.

Este nuevo objeto es el que utilizaremos para marcar que es lo que deseamos ver en nuestra **Lista de operaciones** es el resultado de la operación.

27. Inicia una ejecución de prueba.

28. Selecciona la resta como operación ha efectuar.

Ahora nosotros antes de pulsar el botón calcular deberíamos indicar que es lo que querríamos ver en la **Lista de operaciones**.

*29. Marca el **OptionButton** que hemos insertado anteriormente.*

Observa como la selección que teníamos en la resta ha desaparecido. En el momento que tuviéramos que realizar la operación, el programa no nos enseñaría ningún tipo de solución ya que no está seleccionada ninguna de las operaciones.

30. Marca ahora la multiplicación.

Observa como ha desaparecido la selección del último **OptionButton** que habíamos insertado.

31. Detén la ejecución del programa.

Colocar un Frame

Este problema se puede solucionar insertando un nuevo objeto llamado **Frame**. Este objeto lo que hace es mantener separados diferentes objetos **OptionButton** que se encuentran dentro de un mismo formulario. Este elemento se utiliza para que así puedan marcarse de forma independiente grupos de **OptionButton**. Eso sí, solo se

podrán marcar, uno y solamente uno de los que tenemos dentro de cada **Frame**.

Cuando nosotros coloquemos diferentes **OptionButton** dentro de un **Frame** podremos hacer que todos estos se muevan a la vez al mover el **Frame**. Esto es debido a que el **Frame** actúa como contenedor de los **OptionButton**.

Para colocar este objeto y todos los que deberá llevar en su interior crearemos, en primer lugar el **Frame**, después seleccionaremos el objeto **OptionButton** y lo dibujaremos en el interior. De esta manera unos objetos dependerán de los otros y actuarán como si de un grupo se tratase. Si no lo hacemos de esta forma no conseguiremos nuestro propósito. Veamos lo que hemos explicado con un ejemplo.

*32. Elimina el **OptionButton** que creaste anteriormente.*

*33. Inserta un **Frame** en nuestro formulario.*

Observa que el objeto insertado es un recuadro con un borde y en la parte superior izquierda aparece un texto. Aquí pondremos texto para que nos aclare la utilidad de este grupo de **OptionButton**. Si deseamos que no se vea este borde al ejecutar la aplicación deberemos poner la propiedad **BorderStyle** a **0**.

Recomendamos utilizar siempre borde ya que de esta forma hacemos que el usuario de la aplicación sepa que **OptionButton** actúan conjuntamente.

*34. Mueve el **Frame** hasta la posición **3000, 3600** con un tamaño de **2775 x 735**.*

*35. Cámbiale el color de fondo con la propiedad **BackColor** para que sea igual que el fondo del formulario.*

*36. Accede a la propiedad **Caption** y escribe **Mostrar**.*

Observa como **Mostrar** ha aparecido en la esquina superior izquierda del **Frame**.

Insertando objetos en su interior

Ahora vamos a colocar dos **OptionButton** en el interior de este **Frame**. Para insertar cada uno de ellos sigue los siguientes pasos, no te saltes ninguno ya que puede ser que no conseguirás que los dos controles estén dentro del **Frame**. Para hacer todos estos pasos deberás asegurarte que tienes seleccionado el **Frame**. Esto lo podrás comprobar en el momento en el que aparecen los cuadros para modificar el tamaño de este objeto.

*37. Haz un clic sobre el objeto **OptionButton** del cuadro de herramientas.*

Si mueves el ratón hasta colocarte sobre el formulario podrás observar como este ha tomado forma de cruz.

*38. Coloca el puntero de ratón dentro de nuestro **Frame**.*

*39. Pulsa el botón izquierdo del ratón y mientras lo mantienes pulsado muévete hasta que el nuevo objeto tenga un tamaño de **1095 x 375** aproximadamente. Cuando consigas ese tamaño ya puedes soltar el botón del ratón.*

Ahora que ya tenemos colocado el primer **OptionButton** vamos a moverlo a su posición.

*40. Sitúa nuestro primer **OptionButton** a la posición **240, 240**.*

Observa que esta posición tiene referencia con el **Frame** que lo contiene y no con el resto del formulario.

*41. Coloca otro **OptionButton** dentro de nuestro **Frame** tal y como hemos hecho en los pasos: 37, 38 y 39. Recuerda tener seleccionado el **Frame**.*

*42. Sitúalo en la posición **1440, 240**.*

*43. Selecciona el objeto **Option1**, ponle de **(Nombre): Resulado** y de **Caption: Resultado**. Cámbiale el color de fondo con **BackColor** para que tenga el mismo que el formulario.*

*44. Selecciona el objeto **Option2**, ponle de **(Nombre): Operación** y de **Caption: Operación**. Cámbiale el color de fondo para que tenga el mismo que el formulario.*

*45. Asegúrate que la propiedad **Value** de **Resultado** está a **True**.*

Vamos a comprobar como estos dos últimos objetos que hemos insertado dependen del **Frame**.

*46. Haz un clic sobre el borde del **Frame**.*

47. Muévelo hasta cualquier otra posición del formulario.

Observa como al moverlo también has movido los dos objetos que hay en su interior.

48. Vuélvelo a colocar en la misma posición que estaba. (Mira el punto 34).

ComboBox (Lista desplegable)

Un **ComboBox** tiene características comunes de un **TextBox** y de un **ListBox**. Un **TextBox** ya que se puede escribir texto en el recuadro de texto que aparece y de un **ListBox** ya que podemos seleccionar uno de los elementos que aparecen en la lista desplegable de dicho control.

En nuestro caso utilizaremos este nuevo control para hacer que el usuario escoja entre: **Añadir a la lista** y **No añadir a la lista**. Con lo que añadirá o no a la lista, que insertaremos después, el resultado o la operación completa que hemos realizado anteriormente.

*49. Pulsa un doble clic sobre **ComboBox**, en la barra de herramientas.*

Observa como en todos los casos que hemos querido insertar un objeto y hemos hecho un doble clic el objeto se ha colocado en el centro del formulario.

*50. Muévelo hasta la posición **3360, 4800** con un tamaño de **1935 x 315**.*

*51. Accede a la propiedad **(Nombre)** y escribe **Añadir**.*

Insertando elementos a la lista desplegable

*52. Accede a la propiedad **List**, pulsa en el botón con una flecha hacia abajo que verás a la derecha de esta propiedad.*

Todo lo que escribamos aquí será lo que aparecerá cuando nosotros desplieguemos el **ComboBox** que acabamos de insertar. Cada línea corresponde a un elemento diferente.

*53. Escribe: **Añadir a la lista**.*

Nosotros en este momento ya hemos insertado uno de las dos líneas que debe aparecer dentro de este objeto. Si queremos insertar el siguiente elemento lo tenemos que escribir en la siguiente línea de esta lista. Para ello no deberemos pulsar **Intro** ya que si lo hacemos nos saldríamos de la propiedad **List**. Para seguir insertando elementos deberemos pulsar las teclas **Ctrl** e **Intro** conjuntamente.

*54. Pulsa **Ctrl** e **Intro**.*

*55. Escribe: **No añadir a la lista**. Pulsa **Intro** al terminar.*

Bloqueamos el objeto.

Como hemos dicho anteriormente este nuevo objeto tiene la propiedad de poder escribir y añadir elementos en su interior. En nuestro caso no nos interesa que el usuario modifique o escriba en el cuadro de texto, ya que solo queremos que utilice una de las dos propiedades que contiene.

*56. Accede a la propiedad **Locked** y pon su valor a **True**.*

Iniciar contenido

Para hacer que nuestro objeto ya se inicie con algún contenido en su interior solo tenemos que introducir texto dentro de la propiedad **Text**

En nuestro caso podemos hacer que se inicie con la opción que no inserta ningún tipo de operación en la lista.

*57. Accede a la propiedad **Text** y escribe **No añadir a la lista**.*

58. Realiza una ejecución de prueba. Despliega este objeto e intenta seleccionar alguna de las opciones que contiene en su interior.

Podrás observar que en el programa no ocurre nada de nada ya que todavía no hemos insertado las líneas de código pertinentes.

59. Detén la ejecución de prueba.

ListBox (Lista)

Un **ListBox** es un elemento que nos muestra una lista de elementos de los que el usuario de la aplicación puede escoger uno o más de ellos. Normalmente si el número de elementos que hay dentro de la lista excede del espacio que hemos reservado para la visión del contenido de esta aparecen unas barras de desplazamiento para podernos mover con facilidad sobre la lista.

Cada elemento de la lista tiene un número que nos indica el lugar que ocupa. Es muy importante tener en cuenta que el primer elemento de la lista tiene como índice **0**. La propiedad que nos indica el índice de cada elemento es **ListIndex**. Si no hay ningún elemento en la lista el valor de esta propiedad es **-1**. Utilizando esta propiedad podremos saber en que momento existen o no elementos dentro de la lista.

Si nosotros quisiéramos saber cuantos elementos hay en la lista utilizaríamos la propiedad **ListCount**.

*60. Haz un doble clic sobre el elemento **ListBox**.*

Observa que aparentemente tiene la misma estructura que un cuadro de texto.

*61. Sitúalo en **240, 3600** con un tamaño de **2655 x 1620**.*

*62. Accede a la propiedad **(Nombre)** y escribe **ListaOperaciones**.*

Recuerda que en la propiedad **(Nombre)** no pueden existir espacios en blanco.

Observa que en el interior de la lista ha aparecido el **(Nombre)** que hemos escrito anteriormente. Este objeto no tiene ni propiedad **Caption** ni **Text** con la cual cosa, si nosotros quisiéramos introducir algún contenido en nuestra lista tendríamos que hacerlo con la propiedad **List** al igual que hemos hecho con el **ComboBox** anteriormente. En nuestro caso no introduciremos ningún tipo de texto ya que los iremos insertando durante la ejecución de la aplicación.

Al realizar la ejecución del programa el texto que vemos en el interior de esta lista en el modo edición no aparece.

Podemos hacer que a medida que se introducen los valores en la lista se ordenen automáticamente. Esto lo haríamos poniendo la propiedad **Sorted** a **True**. Puedes activarla si lo deseas.

Nosotros podemos permitir que nuestro usuario pueda seleccionar uno o varios elementos que aparezcan en nuestra lista, para realizar algún tipo de operación con los elementos seleccionados. En nuestra aplicación esto no tiene ningún tipo de importancia ya que después de añadir elementos en la lista, el usuario no puede hacer ningún tipo de operación con estos elementos.

Para modificar esta opción deberemos utilizar la propiedad **MultiSelect**. Esta propiedad tiene tres valores: **None**: que solo nos permite hacer la selección de un solo objeto, **Simple**: nos permite hacer una selección múltiple haciendo clic con el ratón sobre cada elemento que queremos seleccionar y **Extended**: también nos permite hacer selecciones múltiples pero tendremos que utilizar las teclas **Mayúsculas** o **Ctrl** juntamente con el botón del ratón.

En nuestro caso dejaremos el valor de la propiedad **MultiSelect** a **None**. Ya que no haremos ningún tipo de operación al seleccionar los elementos que aparezcan en la lista.

Bloquear controles

Una vez tenemos todos y cada uno de los elementos que forman parte de esta pequeña aplicación, vamos a bloquear los controles para que de forma fortuita no los movamos por el formulario cambiando así su posición.

*63. Selecciona **Bloquear controles** dentro de **Formato**.*

Para bloquear los elementos no deberás tener seleccionado ninguno de los elementos que forman parte del formulario.

Si accedes a cualquier objeto que forma parte de nuestro proyecto e intentas moverlo verás que es completamente imposible. Con esta opción activada solo podrás acceder al código de cada elemento haciendo un doble clic en el elemento deseado.

Líneas de código

En esta practica solo te pediremos que copies las líneas de código que te mostraremos a continuación en los elementos que te indiquemos, no explicaremos ninguna instrucción, ni ninguna estructura, ya que esto lo haremos en lecciones posteriores.

Intenta averiguar para que sirven cada una de las líneas de código que forma parte de esta aplicación, si no las entiendes, tranquilo, más adelante sabrás para que sirven y como funcionan cada una de ellas. Conforme vas introduciendo las líneas de código observa las ventanas de ayuda que te van apareciendo en pantalla, familiarízate con ellas. Están explicadas en lecciones posteriores.

*64. Haz doble clic sobre el botón **Borrar**.*

65. Escribe las siguientes líneas, recuerda que la primera y la última de ellas no debes escribirlas ya que te las mostrará el ordenador.

```
Private Sub Borrar_Click()
   PrimerOperando.Text = «»
   SegundoOperando.Text = «»
   MostrarResultado.Caption = 0
   PrimerOperando.SetFocus
End Sub
```

Este botón lo utilizamos para iniciar la calculadora.

*66. Pulsa **Mayúsculas + F7** para visualizar el formulario.*

*67. Accede a la ventana de código del botón **Calcular** y escribe las siguiente líneas de código:*

Observa que al principio de algunas líneas aparece este símbolo **(*)**, no debes copiarlo. Este símbolo quiere decir que la línea que aparece a continuación va seguida, en la misma línea, a la anterior. Ten cuidado con esto ya que una misma instrucción debe ocupar una sola línea, más adelante veremos como escribimos una instrucción en diferentes líneas.

```
Private Sub Calcular_Click()
   Dim Operador As String
   If PrimerOperando.Text = "" Or SegundoOperando.Text
= (*) "" Then
      MsgBox ("Falta algún operando")
      Exit Sub
   End If
  If Sumar.Value = True Then MostrarResultado.Caption =
(*)          Val(PrimerOperando.Text)          +
Val(SegundoOperando.Text)
  If Restar.Value = True Then MostrarResultado.Caption =
(*)          Val(PrimerOperando.Text)          -
Val(SegundoOperando.Text)
          If Multiplicar.Value = True Then
MostrarResultado.Caption (*) = Val(PrimerOperando.Text)
* Val(SegundoOperando.Text)
   If Dividir.Value = True Then
      If Val(SegundoOperando.Text) = 0 Then
         MsgBox ("No se puede dividir entre 0")
         Exit Sub
      End If
    MostrarResultado.Caption = Val(PrimerOperando.Text)
/ (*) Val(SegundoOperando.Text)
   End If
   If Añadir.Text = "Añadir a la lista" Then
      If Operacion.Value = True Then
```

```
        If Sumar.Value = True Then Operador = "+"
        If Restar.Value = True Then Operador = "-"
        If Multiplicar.Value = True Then Operador = "*"
        If Dividir.Value = True Then Operador = "/"
        ListaOperaciones.AddItem PrimerOperando.Text &
(*) Operador & SegundoOperando.Text & "=" &
(*) MostrarResultado.Caption
      Else
                                ListaOperaciones.AddItem
MostrarResultado.Caption
      End If
   End If
End Sub
```

Vamos a comentar por encima que es lo que realiza este botón: Antes de mirar que operación tenemos seleccionada comprobamos que el usuario haya puesto algún número dentro de los dos operadores, si falta alguno aparece un mensaje de error en la pantalla informando al usuario. Después miramos cual de las operaciones está activada, si es la **suma**, la **resta** o la **multiplicación** se realiza la operación sin ningún tipo de problema. Miramos si la operación que debemos realizar es la **división**, si es así miramos que el segundo operador no sea igual a 0 ya que esto nos podría dar un error y el programa abortaría[1]. Si es así nos aparece un nuevo mensaje que nos informa que no se puede realizar una división entre 0, evitando así el error y el aborto del programa. Una vez realizada la operación miramos si está seleccionada la opción: **Añadir a la lista**, si no está seleccionada no pasa absolutamente nada, en cambio si esta opción está activada pasamos a mirar si debemos añadir a la lista es la operación completa o solo el resultado. Acto seguido añadimos lo que corresponda a la lista.

68. Guarda el proyecto.

69. Crea un archivo ejecutable.

70. Realiza una ejecución de prueba para poder observar como funciona la aplicación.

Te recomendamos que intentes averiguar para que sirven cada una de las líneas de código que hemos escrito para que funcione la aplicación. Más adelante las entenderás todas sin ningún tipo de problema.

Fin de la lección 3.

[1] Terminar la ejecución de un programa de forma incorrecta.

LECCIÓN 4

En esta lección vamos a ver que son y para que sirven: las variables, constantes, tablas y matrices. Todas ellas son estructuras de almacenamiento de datos temporales, pero que nos pueden facilitar la programación en muchas ocasiones.

Variables, constantes y matrices

No podemos decir que estos elementos sean estructuras básicas, ya que no son un grupo de instrucciones, sino que son elementos que nos pueden ayudar a almacenar valores, de forma temporal, para usarlos en nuestra aplicación de la forma que más nos convenga. También tenemos que pensar que en muchas estructuras que veremos en lecciones posteriores utilizaremos variables, constantes, tablas y matrices.

Variables

Las variables se utilizan, en cualquier lenguaje de programación, para almacenar valores de forma temporal (mientras dura la ejecución del programa). A las variables se les pone un nombre para poder trabajar con ellas y se indican de que tipo son. Este tipo nos informa que clase de datos se pueden almacenar dentro de esta variables, (los diferentes tipos de variables los veremos más adelante).

Constantes

Las constantes nos pueden parecer que son exactamente iguales que las variables, pero no es así. Las variables nos sirven para almacenar valores, que podemos modificar durante la ejecución del programa.

Las constantes, en cambio, no cambian de valor durante la ejecución de la aplicación. Se suelen utilizar para sustituir un número o valor, difícil de recordar o que suele salir muchas veces durante la aplicación. Imagina el caso de una aplicación en la que necesites utilizar muchas veces el valor Pi. Si siempre que necesitáramos este valor tuviéramos que escribir 3'1415... sería un poco engorroso. Pero, gracias a las constantes nosotros podemos definir una llamada Pi con valor 3'1415... y en el momento en el que necesitemos realizar una operación con el valor Pi tendremos que poner el nombre de la constante y el ordenador se encargará de sustituirlo por su valor.

Matrices

Las matrices son un grupo de valores que tienen un mismo nombre y se diferencian entre ellas por el lugar que ocupan. A este lugar que ocupa se le llama índice. Gracias a este índice podemos crear un código, utilizando estructuras de repetición que nos ayuden a trabajar con estos datos, ahorrando de esta manera código.

Normalmente, las matrices se definen con un límite inferior y uno superior. Con la resta de ambos tenemos el número de elementos que pueden entrar dentro de la matriz. Tenemos que pensar que si nosotros solo definimos el valor superior, el primer objeto tendrá como índice 0 y el último el número que nosotros hayamos definido como límite superior. De esta forma, si nosotros definimos una matriz de una sola dimensión con límete superior 5, en realidad tenemos 6 objetos con índices 0, 1, 2, 3, 4, 5.

Al igual que las variables y constantes, en las tablas también se tiene que definir el tipo de valor que vamos a almacenar dentro.

Las matrices podemos definirlas de solo una dimensión, como si se tratase de una gran fila de datos y en otras muchas ocasiones nos puede interesar utilizar estructuras de dos dimensiones o incluso más, en las que buscaremos los datos por la fila y la columna que ocupan. En este caso estas matrices tendrán 2 índices.

Por ejemplo en el caso de un tablero de ajedrez nos interesa saber en que fila **y en que** columna **se encuentra situado una ficha determinada para saber si el movimiento que deseamos realizar está o no permitido.**

Más adelante explicaremos como crear, definir y trabajar con variables, constantes y matrices.

Vida de una variable

Nosotros podemos definir una variable para que solo nos sea útil mientras dura el procedimiento en el que se le ha creado. En el momento de finalizar dicho procedimiento, el espacio reservado en memoria para la variable queda liberado. De esta forma, si nosotros queremos utilizar un mismo nombre para diferentes variables que se utilizan en diferentes procedimientos podemos hacerlo sin miedo a que los valores se mezclen o cambien sin que nosotros tengamos control sobre dichos cambios.

Definir una variable

Para definir una variable dentro de un procedimiento utilizaremos la instrucción: Dim [Nombre Variable] As [Tipo variable] **.**

Nombre Variable **: definiremos el nombre que tiene la variable. Este nombre no puede tener más de 255 caracteres, debe comenzar con una letra y no debe contener puntos.**

Tipo variables **: Especificaremos el tipo de datos que se pueden almacenar dentro de la variable.**

Tipos de variables

Vamos a especificar los diferentes tipos de datos que podemos definir al crear una variable, constante o matriz.

Byte **: Este tipo de dato lo utilizamos para contener números enteros positivos en el intervalo de 0 a 255.**

Boolean **: En este tipo de doto sólo tiene dos posibles valores,** True (-1) **o** False (0) **.**

Integer **: En este tipo de datos contiene variables enteras almacenadas como números enteros de 2 bytes en el intervalo de –32.768 a 32.767.**

Long **: Almacena números completos entre –2.147.483.648 y 2.147.483.647.**

Currency **: Es un tipo de datos con un intervalo de -922.337.203.685.477,5808 a 922.337.203.685.477,5807. Es recomendable utilizar este tipo de dato para cálculos de tipo monetario y cálculos de punto fijo donde es muy importante la exactitud.**

Single **: El intervalo que se puede almacenar en este tipo de datos es de: -3,402823E38 a -1,401298E-45 para valores negativos y de 1.401298E-45 a 3.402823E38 para valores positivos.**

Double **: El intervalo de almacenamiento de este tipo de dato es de: -1,79769313486232E308 a -4,94065645841247E-324 para valores negativos y de 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos.**

Es importante, una vez vistos los intervalos, de cada uno de los diferentes tipos de datos, saber utilizar y definir cada uno de ellos según nuestras necesidades. Debes pensar que si utilizamos un tipo de dato que tenga un intervalo muy pequeño para realizar cálculos complejos puede ser que nos de error en según que casos y si utiliza-

mos un tipo de variable con un intervalo muy grande para una operación muy sencilla estaremos ocupando memoria innecesariamente.

Declaración de una variable

En este apartado veremos como y que métodos podemos utilizar para declarar una nueva variable.

Declaración implícita

En un principio, dentro de un procedimiento, no hace falta que definamos el nombre y el tipo de una nueva variable. Si Visual Basic encuentra un nuevo nombre de una variable, este la define automáticamente. Esto en ocasiones puede ir muy bien pero en otras ocasiones nos puede producir un error que puede ser difícil de detectar. Imagínate que nosotros en una línea de código hacemos referencia a una variable llamada Contador **. Visual Basic como ve que es una nueva variable la crea automáticamente. Pero, si nosotros en otro momento queremos hacer referencia a esta nueva variable y escribimos** Contaidor **, Visual Basic cree que es una nueva variable creando así una diferente.**

De esta forma si queremos hacer cálculos con la variable Contaidor **, puede ser que tengamos algún tipo de error.**

Entonces, viendo esto nos tenemos que plantear si es mejor definir nosotros las variables o dejar que sea Visual Basic **quien defina las nuevas variables que vamos necesitando.**

Declaración explícita

Para no tener problemas como el anteriormente citado, nos podemos obligar a definir las variables que utilizamos. Al definirlas, tanto debemos indicar un nombre para la nueva variable, como el tipo de dato que vamos a almacenar. De esta forma, si Visual Basic **encuentra una nueva variable, en lugar de crearla, nos avisa que no la hemos definido, produciéndose así un error de compilación.**

Para hacer que Visual Basic **utilice la declaración explícita podemos utilizar dos sistemas diferentes.**

Podemos hacer que Visual Basic **active esta opción para todos y cada uno de los módulos que se creen a partir del momento de activar esta opción o activarlo nosotros manualmente.**

. Práctica 1

1. Abre un nuevo proyecto.

*2. Selecciona **Opciones** dentro del menú **Herramientas**.*

*3. De todas las carpetas selecciona **Editor** y activa la opción **Requerir declaración de variables**.*

4. Acepta el cuadro de diálogo actual.

*5. Mira el código de este proyecto, con el menú: **Ver - Código** o pulsa **F7**.*

6. Observa que no hay ninguna línea de código en nuestro proyecto.

7. Cierra el proyecto actual, sin guardar los cambios.

*8. Abre un nuevo proyecto, con la opción **Nuevo proyecto** de la opción **Abrir**. En el cuadro de diálogo que te aparece a continuación deja la selección actual y pulsa en **Aceptar**.*

9. Mira el código de este proyecto.

Observa como ahora dentro del apartado (General) (Declaraciones) **aparece la instrucción** Option Explicit **. Esta instrucción nos indica que en este proyecto solo se pueden utilizar variables que se definan dentro de este mismo apartado o dentro de los diferentes procedimientos de los que conste la aplicación.**

Si nosotros queremos insertar esta instrucción en una aplicación que ya tengamos en uso o que ya exista, tendremos que escribirla dentro del apartado que hemos especificado anteriormente.

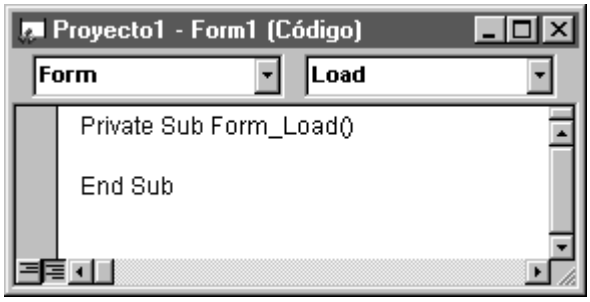
Vamos a ver como podemos llegar hasta este apartado de una forma fácil.

. Práctica 2

*1. Con el último proyecto en pantalla, quita la selección: **Requerir declaración de variables**.*

2. Abre un proyecto que tengas grabado.

3. Accede al código de cualquiera de los objetos que tienes en el formulario.



Observa la pantalla con el código. (En nuestro caso estamos enseñando el código que aparece en el momento de hacer doble clic en el formulario de una aplicación nueva).

Observa como en dicha ventana de código siempre aparece dos listas desplegables. La lista de la izquierda **es donde se irán situando los** nombres **de los diferentes objetos que están insertados en el formulario actual. Mientras que en la lista de la** derecha **aparecerán los** eventos **del objeto que esté seleccionado en la lista de la izquierda.**

*4. Despliega la lista de la **izquierda** y selecciona la opción **(General)** observa como la lista de la **derecha** cambia y aparece **(Declaraciones)**, si no aparece automáticamente despliega la lista y busca dicha opción.*

*5. Cuando estás en este apartado ya puedes escribir **Option Explicit**.*

De aquí en adelante, si continuas trabajando con este proyecto, en el momento de utilizar una nueva variable, si no está creada te aparecerá un mensaje de error.

¿Donde declaramos las variables?

Nosotros podemos declarar una variable en diferentes sitios, según el alcance

que le queramos dar. Nosotros podemos crear variables que solo se utilicen dentro de un procedimiento determinado, variables que se pueda utilizar en cualquier procedimiento de un formulario o en cualquier procedimiento de cualquier formulario de un mismo proyecto.

Es esta lección vamos a ver los dos primeros casos, mientras que el tercero lo veremos en el momento de insertar varios formularios dentro de un mismo proyecto.

De aquí en adelante pensaremos que tenemos escrita la instrucción Option Explicit **para que no podamos utilizar una variable sin que antes la tengamos definida.**

Variables útiles en un procedimiento

Si nosotros queremos crear una variable dentro de un procedimiento lo podemos hacer en cualquier punto dentro de este. (No hace falta crearlo en un lugar determinado siempre y cuando se declare antes de utilizarla).

Siempre, al utilizar este tipo de variables, deberemos pensar que el valor que tenga una variable dentro de este procedimiento no se podrá consultar o variar desde cualquier otro.

. Práctica 3

1. Crea un nuevo proyecto.

*2. Inserta dos **CommandButton** a los que llamaremos **Boton1** y **Boton2**.*

*3. Inserta un **Label** al que llamaremos **Valor**.*

*4. Escribe dentro del **Boton1**, haciendo doble clic, estas líneas de código.*

```
Private Sub Boton1_Click()
   Dim Contador As Integer
   Valor.Caption = Contador
End Sub
```

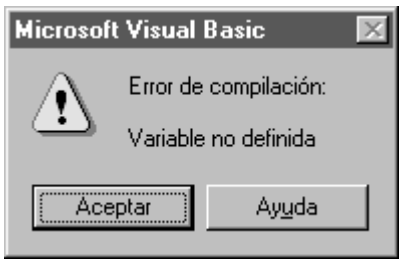
*5. Y dentro del **Boton2** estas otras:*

```
Private Sub Boton2_Click()
   Valor.Caption = Contador
End Sub
```

Observa como en el primer botón hemos definido una variable llamada Contador **, mientras que en el segundo botón no.**

*6. Realiza una ejecución de prueba. Pulsa en el primer **botón**. Observa como el valor de la variable a pasado a nuestro **Label**.*

*7. Pulsa ahora en el segundo **botón**.*



Se produce un error, apareciendo una ventana como la que mostramos en esta imagen. Este error nos avisa que existe una variable que no está definida.

Aunque parezca que la tenemos definida no es así. La definición de dicha variable está en otro procedimiento.

*8. Pulsa el botón **Aceptar** y observa donde se ha producido el error.*

9. Detén la ejecución de la aplicación.

Si deseas utilizar una variable con el mismo nombre en otro procedimiento deberías volverla a definir. Piensa que aunque se llamen exactamente igual, son variables diferentes ya que están en procedimientos diferentes.

Si nosotros creamos las variables con Dim **al volver a entrar dentro del evento donde se ha creado la variable, esta se vuelve a iniciar. Si queremos que dentro de un procedimiento el valor de una variable se conserve deberás definirla poniendo** Static **en lugar de** Dim **.**

10. Modifica el código de nuestra aplicación para que quede de la siguiente forma:

```
Private Sub Boton1_Click()
   Static Contador As Integer
   Contador = Contador + 1
   Valor.Caption = Contador
End Sub
```

```
Private Sub Boton2_Click()
   Dim Contador As Integer
   Contador = Contador + 1
   Valor.Caption = Contador
End Sub
```

Lo que pretendemos con este ejemplo es que veas como utilizando una variable definida como Static **se puede mantener el valor dentro de un procedimiento, mientras que la misma variable definida con** Dim **en otro procedimiento actúa completamente diferente.**

11. Realiza una ejecución de prueba.

*12. Pulsa repetidamente el primer **botón**.*

Aunque cada vez que pulsamos en botón estamos entrando en el procedimiento la variable guarda el valor y se le sigue sumando 1 gracias a la línea Contador = Contador + 1 **.**

*13. Pulsa repetidamente el segundo **botón**.*

Observa como cada vez que se entra en este botón el valor vuelve a ser el mismo, ya que no se guarda el valor de las veces anteriores.

*14. Vuelve a pulsar el primer **botón**.*

El valor que teníamos almacenados en este procedimiento vuelve a surgir.

15. Detén la ejecución.

Observa bien las diferencias entre estos dos tipos de asignación de variables.

Variables útiles en todos los procedimientos

Si lo que nosotros deseamos es poder utilizar una variable en cualquiera de los procedimientos que forman parte de un módulo (formulario) deberemos definir dicha variable en el apartado General - Declaraciones **.**

En un principio cuando queramos definir variables de este tipo lo haremos de la siguiente forma:

Private [Nombre Variable] As [Tipo Variable] **.**

16. Quita la declaración de las variables que hemos hecho anteriormente en cada uno de los botones y declara una sola variable para todo el formulario.

Constantes

Una constante posee las mismas características que una variable, ya que se puede definir en los mismo sitios, pero no se puede modificar su contenido.

Si deseamos utilizar la constante solo dentro de un procedimiento la definiremos allí, sin que esta puedas ser usada por otro procedimiento. Si lo que deseamos es que se pueda utilizar en cualquiera de los procedimientos definidos dentro de un formulario, deberemos definirla en el apartado General - Declaraciones **.**

¿Cómo se declara una contante?

Tanto dentro de un procedimiento como en General - Declaraciones **las constantes se pueden declarar de la misma forma. En un principio existen dos formas diferentes de definir las constantes: indicando que tipo de valor se guardará en su interior o sin indicarlo, con lo que** Visual Basic **adaptará el contenido de la constante a las utilidades que nosotros le demos. Este tipo de dato, que se adapta a todo, se le llama** Variant **. Con este tipo de datos tenemos la ventaja que no nos tenemos que preocupar de que tipo es el dato que guardaremos dentro de la constante, pero tiene el pequeño inconveniente que ocupa un poco más de memoria que muchos otros tipos de datos que veremos más adelante.**

Definición de la constante sin indicar el tipo de dato:

Const [Nombre Constante] = [Valor Constante]

Definición de la constante indicando el tipo de dato:

Const [Nombre Constante] As [Tipo de dato] = [Valor Constante]

. Práctica 4

Vamos a imaginar que queremos realizar una aplicación en la que partiendo de un número inicial de alumnos, cada vez que pulsemos un botón el número de alumnos aumente en 1.

1. Borra las líneas de código que hemos escrito en las prácticas anteriores y escribe el siguiente código allí donde corresponda. (Ten presente no estamos utilizando el segundo botón).

```
Option Explicit
Public Contador As Integer
Const Alumnos = 45
```

```
Private Sub Boton1_Click()
```

```
    Contador = Contador + 1
    Valor.Caption = Contador + Alumnos
End Sub
```

2. Realiza una ejecución de prueba.

3. Finaliza dicha aplicación.

Podríamos pensar que no hace falta crear una constante llamada Alumnos donde introdujéramos el número de alumnos que tenemos. Pero piensa que una constante es de suma utilidad en el momento que estamos realizando una gran aplicación en la que surge muchas veces una cantidad con la que tenemos que trabajar.

Ejemplo: imagina que tienes una aplicación con cientos de líneas en la que calculas el promedio de notas de la clase, el promedio de faltas en un trimestre, etc. Bien, pues en todos estos cálculos necesitas saber el número de alumnos que tienes. Si utilizaras este misma aplicación otros años deberías cambiar el número de alumnos. Entonces tendrías que buscar línea a línea allí donde realizas dichos cálculos, para cambiar el número de alumnos. En cambio si utilizas una constante, con solo cambiar el valor de la constante, todos los cambio ya están hechos.

4. Modifica el valor de la constante ***Alumnos****.*

5. Realiza otra ejecución del programa.

Observa que funciona exactamente igual, pero con valores diferentes.

6. Detén la ejecución del programa.

En prácticas posteriores trabajaremos con constantes.

Matrices

Como ya hemos explicado en la introducción las matrices nos permiten trabajar con un grupo de datos, los cuales se almacenan bajo un mismo nombre, pero se diferencian unos de otros por el lugar que ocupan, a este lugar le llamamos Indice.

Un peligro que deberemos tener siempre presente al trabajar con matrices, es no hacer referencia NUNCA a un índice que no esté definido. Con esto quiero decir que siempre debemos tener presente el tamaño de nuestra matriz. De esta forma si nosotros hemos definido una matriz de una dimensión con 5 elementos y hacemos referencia al elemento 10, Visual Basic nos dará un error de desbordamiento. Él a intentado acceder a un lugar que está fuera del tamaño al tamaño real de la matriz.

Como definir una matriz

Podemos decir que una matriz se define exactamente igual que una variable y una constante, pero con la diferencia que deberemos especificar el tamaño que deseamos que tenga. Esto lo hacemos de la siguiente forma, según si la matriz es de una o más dimensiones:

Matriz de una dimensión

```
Dim [Nombre Matriz] ([Tamaño]) As [Tipo de datos]
```

Tamaño: aquí definiremos, siempre entre paréntesis, el tamaño de nuestra matriz. Debemos recordar que si nosotros definimos una matriz de 5 elementos, el primer

elemento está ocupando la posición 0 y el último la 5, por lo tanto en realidad estamos trabajando con 6 elementos.

Si nosotros deseamos empezar a trabajar desde un número determinado de índice hasta otro deberemos definir la matriz de la siguiente manera:

Dim [Nombre Matriz] ([Índice inicial] To [Índice final]) As [Tipo de datos]

Una matriz definida de esta forma tendrá como primer índice el Índice inicial y como último el Índice final.

Matriz de más dimensiones

Dim [Nombre Matriz] ([Tamaño fila], [Tamaño columna]) As [Tipo de datos]

Una matriz de más de una dimensión podemos pensar que es como una cuadricula en la que necesitaremos dos o más índices para hacer referencia a alguno de los objetos que tenemos en su interior.

Tamaño fila: aquí definiremos el número de filas que deseamos tenga nuestra matriz.

Tamaño columna: aquí definiremos el número de columnas que deseamos tenga nuestra matriz.

Vamos ha hacer una representación gráfica de una matriz con 5 filas y 10 columnas **:**

	Columnas									
Filas	1	2	3	4	5	6	7	8	9	10
1										
2										
3										
4										
5										

Observa como las filas **están dispuestas** horizontalmente **, mientras que las** columnas **lo hacen** verticalmente **.**

Así de esta forma, en el momento en el que nosotros queremos hacer referencia a uno de los elementos introducidos en la matriz, deberemos indicar el número de fila **y el de** columna **que ocupa dicho elemento.**

Al igual que en las tablas de una sola dimensión podemos especificar donde queremos que empiece el índice de nuestra matriz, pero en este caso tendremos que especificar cada uno de los índices que utilizamos (fila y columna).

Esta podría ser la definición de una matriz de dos dimensiones definiendo tanto el inicio como el final del índice.

Dim [Nombre Matriz] ([Índice inicial fila] To [Índice final fila], [Índice inicial columna] To [Índice final columna]) As [Tipo de datos]

Asignar valores

En este apartado vamos a ver como podemos introducir valores tanto en variables como en matrices.

Variables

Para la asignación de valores tenemos que pensar, siempre, el tipo de datos que podemos almacenar dentro de estas. Si en algún momento asignamos algún tipo de dato diferente al que hemos definido al crear la variable se producirá un error.

Para almacenar un valor en una variable solo deberemos escribir una instrucción como esta:

[Nombre Variable] = [Valor]

Valor: será el valor que queremos almacenar en nuestra variable.

Recuerda que si no está definida y no tienes la opción Option Explicit, Visual Basic **definirá automáticamente la variable como tipo** Variant **.**

Podemos decir, para facilitar el entendimiento de la asignación de valores en una variable, que esta se realiza de derecha hacia izquierda.

Matrices

Para asignar un valor a una posición de una matriz, lo haremos exactamente igual que en la asignación de valores en una variables, pero con la diferencia que aquí tenemos que especificar el índice (posición) donde deseamos se almacene el valor.

En el caso de una matriz de una sola dimensión lo deberemos hacer de esta forma:

[Nombre Matriz] ([Indice]) = [Valor]

En caso de una matriz de dos dimensiones lo deberemos hacer de esta otra forma:

[Nombre Matriz] ([Indice fila], [Indice columna]) = [Valor]

En el caso de las matrices deberás tener mucho cuidado en no asignar valores a índices que están fuera de la tabla.

Más adelante, cuando veamos las estructuras de repetición, veremos algunas formas de poder movernos por toda la matriz de forma sencilla para poder realizar cálculos, ordenaciones de datos y búsquedas de datos en una matriz.

Matrices de controles

En este capitulo vamos ha ver un sistema con el cual nos podemos ahorrar algunas líneas de código. Sobre todo en el momento en que tenemos muchos objetos que son exactamente iguales y queremos que todos actúen de una misma forma.

Esto lo vamos a explicar mejor con un pequeño ejemplo.

. Práctica 5

Vamos a realizar una práctica en la que tendremos 5 botones, cada uno con una vocal. Nosotros lo que queremos es que cada vez que el usuario pulse un botón se acumulen las letras en un Label **. Por ejemplo si el usuario pulsa el botón con la A, la después la O y por último el botón de la E. En el** Label **deberá aparecer la cadena AOE .**

Es un ejemplo sencillo pero fácil para comprender el funcionamiento de las matrices de controles.

1. Abre un nuevo proyecto.

*2. Inserta un **Label**, ponle como nombre **Cadena**, borra el contenido de dicha etiqueta.*

*3. Inserta un **botón**. Cambia su nombre, poniéndole **Letra**.*

*4. Escribe en su interior la letra **A**.*

5. Ajusta el tamaño del botón, el tamaño de la letra, su apariencia, etc. realiza los cambios como desees.

Ahora vamos a pasar a copiar este elemento 4 veces más para tener los demás botones, los cuales representarán al resto de vocales.

*6. Pulsa un clic sobre el **botón** que tenemos en nuestro formulario para seleccionarlo.*

*7. Accede a la opción **Copiar** dentro del menú **Edición**.*

Aparentemente no ha pasado nada. Pero el ordenador ahora sabe que deseamos copiar este objeto.

*8. Vuelve a acceder al menú **Edición**, pero esta vez selecciona la opción **Pegar**.*

Acto seguido te aparecerá un cuadro de diálogo en el que te avisa que ya existe un objeto que tiene el mismo nombre y si deseas crear una matriz de controles.

9. Contesta afirmativamente.

*10. Repite la acción de copiar hasta que tengas **5 botones**.*

Las copias de los botones irán apareciendo en la esquina superior izquierda de nuestro formulario.

Observa que en el momento que ya copias uno de los botones, contestado afirmativamente a la creación de una matriz de controles, el resto de copias las realiza sin hacerte ningún tipo de pregunta.

11. Sitúa uno debajo del otro todos los botones que hemos creado.

*12. Selecciona el primero de los botones y observa la propiedad **(Nombre)**.*

Podrás observar como el nombre de dicho botón ya no es Letra, sino que pasa a ser Letra(0) **. Este número, que aparece entre paréntesis, es un índice el cual nos sirve para distinguir cada uno de los botones que hemos creado. Nuestra matriz de botones irá desde el botón con índice 0 hasta el que tiene el índice 4. En total 5 elementos.**

A partir de este momento si queremos modificar la propiedad de uno de los botones en concreto, deberemos especificar el número de índice de dicho botón. De este forma podremos utilizar estructuras de repetición para modificar las propiedades de muchos objetos en pocas líneas. Más adelante veremos ejemplos en los que utilizaremos está característica.

Otra característica que tienen estas estructuras de datos es que todo el código que escribamos dentro de uno de ellos afectará igualmente a cualquiera de las copias.

13. Pulsa doble clic en cualquiera de los botones.

Observa que en el momento de entrar en un evento, en la línea de código donde se especifica en que tipo de evento nos encontramos, aparecen las palabras (Index As Integer) **. Este es el índice de los elementos que hemos insertado.**

14. Escribe la siguiente instrucción:

```
Cadena.Caption = Cadena.Caption & Letra(Index).Caption
```

Esta instrucción lo que nos hace es "sumar" el contenido actual de la variable llamado Cadena **con el** Caption **del botón que hemos pulsado.**

Antes de realizar una ejecución de prueba deberemos hacer una última modificación.

Si te fijas todos los botones tienen como propiedad Caption **una** A**. Vamos a modificar cada uno de los botones para que contengan las diferentes vocales.**

15. Accede a cada uno de los botones y modifica la propiedad ***Caption*** *escribiendo dentro de cada uno las diferentes vocales. No hace falta que estén en orden según el índice de cada botón.*

16. Realiza una ejecución de prueba.

17. Pulsa los diferentes botones y observa como se van añadiendo las diferentes letras en nuestro ***Label****.*

18. Detén la ejecución y graba nuestra pequeña práctica.

Fin lección 4

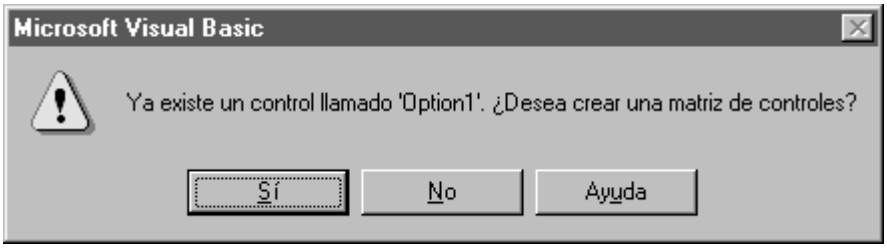
LECCIÓN 8

En esta lección hablaremos de unos objetos que ya hemos utilizado en la lección anterior, pero no vimos ni como funcionaban, ni como se utilizaban. Estamos hablando de los cuadros de mensajes.

¿Qué son los cuadros de mensajes?

En muchas ocasiones cuando realizamos acciones con cualquier programa de **Windows** nos aparecen pequeñas ventanas de información o de error.

Este por ejemplo, es un cuadro de diálogo con el que ya hemos trabajado en lecciones anteriores.



Podemos decir que tenemos dos tipos de cuadros de mensajes con los que podemos trabajar: los cuadros de mensajes propiamente dichos y los de entrada.

Estos cuadros los utilizaremos para mostrar algún tipo de mensaje al usuario de la aplicación, ya sea de error, aviso o de cualquier otro tipo.

Los cuadros de entrada en cambio son ventanas en las que se espera que el usuario escriba algún tipo de texto que nos servirá para continuar con la aplicación.

En ambos cuadros podremos modificar diferentes elementos como el *título*, el *icono*, los *mensajes de los botones*, la *cantidad de botones*, sus *funciones* y *otras características* que veremos a medida que vayamos hablando de cada uno de los tipos de cuadros.

Cuadros de mensajes (MsgBox)

Estos cuadros los utilizaremos para mostrar mensajes o para obtener por parte del usuario respuestas sobre determinadas acciones.

Vamos a enumerar las diferentes partes que podremos modificar en nuestros cuadros de mensajes.

Estos cuadros constan de un **título** en la parte superior de la pantalla. Estos cuadros carecen de menú de control y solo disponen del botón **cerrar** ya que no se puede modificar su tamaño. Suele aparecer un **icono** en la parte izquierda de la ventana. Este icono nos ayuda a identificar de que tipo es el mensaje. Suele aparecer un **mensaje** en el centro de la ventana. En la parte inferior aparecen los diferentes **botones**. Pueden aparecer 1, 2 o 3 botones con diferente texto en su interior.

Más adelante veremos como personalizar todas estas opciones.

Aquí vemos las partes de un mensaje de error de **Visual Basic**.

Sintaxis de un MsgBox

Nosotros mediante un **MsgBox** podemos saber que botón pulsa el usuario. Cada uno de los diferentes botones tiene un valor que se almacenará en una variable con la que después podremos trabajar.

Pongamos el caso del **MsgBox** anterior, si el usuario pulsa el botón **Aceptar** cerraremos dicho mensaje y detendremos la ejecución del programa, mientras que si pulsamos en **Ayuda** mostraremos una ayuda sobre este error. También hay **MsgBox** que no nos interesa saber que botón ha pulsado el usuario con lo que no hace falta almacenar el valor del botón en ninguna variable, este puede ser el caso de un mensaje de error en el que solo aparecerá un botón para cerrar el cuadro de mensaje.

Vamos a ver primero la sintaxis general de esta instrucción:

MsgBox Mensaje *[, Botones e iconos][, Título]*

Las partes entre corchetes indican parámetros opcionales.

Si no deseamos saber que botón ha pulsado el usuario de la aplicación deberemos poner la instrucción tal y como hemos indicado en la sintaxis anterior. En cambio si deseamos conocer que botón a pulsado y actuar en consecuencias deberemos almacenar en una variable el valor que se genera al pulsar dicho botón, entonces deberemos modificar la sintaxis de esta forma:

Valor = MsgBox (Mensaje *[, Botones e iconos][, Título]*)

Observa que hemos añadido unos paréntesis que engloban a todas las opciones del **MsgBox**.

Valor: esta será la variable en la que se almacenará el valor del botón pulsado en el mensaje.

Observa que hemos insertado el signo igual ya que lo que estamos pasando el valor del botón pulsado a la variable **Valor**.

El **mensaje** es la única opción obligatoria que deberemos poner en un **MsgBox**.

Título: si indicamos algún título, este nos aparecerá en el título de la nueva ventana. Si por lo contrario no indicamos título, nos aparecerá el nombre de la aplicación actual.

Botones e iconos: aquí pondremos un valor que nos servirá para especificar que icono y que botones queremos que nos aparezcan.

Botones e iconos del mensaje

Como ya hemos dicho anteriormente en este lugar deberemos indicar un valor que nos indicará el "tipo" de nuestro mensaje. Este valor se obtendrá sumando 4 valores diferentes de 4 tablas que presentamos a continuación:

Botones

Botones a mostrar	*Valor*
Aceptar	0
Aceptar y **cancelar**	1
Anular, **Reintentar** e **Ignorar**	2
Sí, **No** y **Cancelar**	3
Sí y **No**	4
Reintentar y **Cancelar**	5

Iconos

Iconos a mostrar	*Valor*
	16
	32
	48
	64

Botón activado por defecto

Botón por defecto	*Valor*
Primero	0
Segundo	256
Tercero	512
Cuarto	768

Modalidad del mensaje

Modalidad	*Valor*
Aplicación modal	0
Sistema modal	4096

Para conseguir el valor que deberemos poner en el apartado **Botones e iconos** de nuestra sintaxis escogeremos un valor de cada uno de los diferentes grupos que hemos visto anteriormente y los sumaríamos.

Antes de poner un ejemplo vamos a explicar cada uno de los diferentes grupos:

Botones: aquí tenemos una lista de las diferentes combinaciones de botones que podemos mostrar en nuestro mensaje.

Iconos: esta es una lista de los cuatro posibles iconos que podemos mostrar.

Botón activado por defecto: nosotros podremos indicar cual de los botones que tenemos en el mensaje se active en el momento de pulsar la tecla **Enter**.

Modalidad del mensaje: Vamos a definir las dos modalidades.

Aplicación modal: el usuario deberá "contestar" al cuadro de mensaje pulsando sobre alguno de los botones o cerrando dicho cuadro antes de proseguir con la aplicación actual. Con esta opción el usuario podrá seguir utilizando cualquier otra aplicación. Una vez contestada la pregunta el programa continuará según la respuesta.

Sistema modal: El usuario no podrá continuar el trabajo con ninguna aplicación hasta que se conteste el cuadro de mensaje actual. Esta no es una opción muy utilizada ya que se bloquean el resto de aplicaciones hasta que se responde el mensaje.

Vamos a ver como utilizar los objetos **MsgBox** en una aplicación. En este ejemplo veremos como diseñar nuestro mensaje.

Generar un MsgBox

Vamos a crear una simple aplicación en la que tendremos tres objetos, un **TextBox**, un **ListBox** y un **CommandButton**. Esta aplicación nos permitirá escribir algo en el **TextBox** y al pulsar el **CommandButton** nos deberá aparecer un **MsgBox** con la pregunta: *¿Estás seguro que deseas añadir este elemento?*. Si el usuario responde afirmativamente el contenido del **TextBox** pasará al **ListBox**, si el usuario responde negativamente, opción que aparecerá marcada por defecto, no se añadirá el texto al **ListBox**, pero nos aparecerá un nuevo **MsgBox** indicando que no se añadirá ningún elemento a la lista.

. Práctica 1

1. Crea un nuevo formulario.

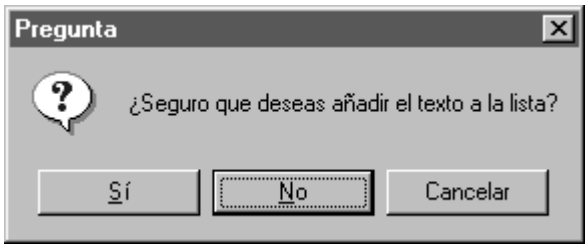
*2. Inserta un **TextBox**, borra el contenido y ponle como **(Nombre) Texto**.*

*3. Inserta un **ListBox**, ponle como **(Nombre) Lista**.*

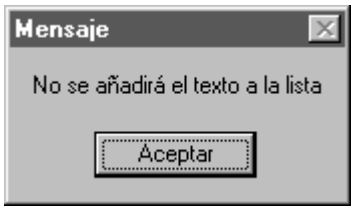
*4. Inserta un **CommandButton** cámbiale el **(Nombre)** por **Insertar**. Cambia también la propiedad **caption** por **Insertar**.*

Vamos a pasar a crear el código para que nos aparezca el mensaje deseado.

Nuestro **MsgBox** deberá mostrar como título: **Pregunta**. El mensaje interior deberá ser: **¿Seguro que deseas añadir el texto a la lista?**. Como icono nos aparecerá un signo de interrogación y nos deberán aparecer tres botones: **Sí**, **No** y **Cancelar**.



El primer mensaje deberá ser como este:



En cambio el segundo mensaje que mostrará esta aplicación tendrá este otro aspecto:

Vamos a ver como podemos generar el primero de los dos **MsgBox**.

En primer lugar vamos a calcular el valor para que nos aparezcan los **3 botones**, el **icono** y el **segundo botón como predeterminado**. Como ya hemos dicho anteriormente deberemos escoger un valor de cada uno de los cuatro grupos que hemos escrito anteriormente.

En el primer grupo tenemos los botones que deseamos aparezcan en el **MsgBox**, al mirar la tabla vemos que los botones **Sí**, **No** y **Cancelar** tienen como valor el ***3***. Para que aparezca el icono de la interrogación deberemos mirar en el segundo grupo, este icono tiene como valor ***32***. El tercer grupo nos determinará cual de los botones queremos que esté como predeterminado, en este caso será el **segundo**, mirando en la tabla veremos que tiene como valor ***256***. Nosotros queremos que el mensaje sea **modal** a la aplicación, por lo tanto el valor del cuarto grupo es un ***0***. Si sumamos los cuatro valores nos da: ***3+32+256+0=291***

Ahora que ya sabemos el valor que debemos poner dentro de la definición de nuestro primer **MsgBox**.

Vamos a ver como quedaría definitivamente el código.

Recuerda que deseamos conocer la respuesta del usuario por lo que necesitamos almacenar el valor del botón pulsado.

5. Haz doble clic dentro de nuestro botón.

6. Escribe el siguiente código. (Por motivos de espacio en nuestro manual el código aparece en dos líneas, pero en Visual Basic se debería escribir en una sola).

```
Respuesta = MsgBox(«¿Seguro que deseas añadir el texto a la lista?», 291, «Pregunta»)
```

Fíjate que hemos creado una variable llamada **respuesta** para almacenar el valor del botón pulsado.

Vamos a ver como trabajar con estos valores.

Valores de retorno de los botones

Vamos a ver a continuación otra tabla con los valores que se devuelven al pulsar los diferentes botones.

Botón	*Valor*
Aceptar	1
Cancelar	2
Anular	3
Reintentar	4
Ignorar	5
Sí	6

No 7

En nuestro ejemplo solo utilizaremos **2** valores el **6** para el **Sí** y el **7** para el **No**.

7. Modifica el código que tienes dentro del botón para que sea como este:

```
Private Sub Insertar_Click()
   Respuesta = MsgBox(«¿Seguro que deseas añadir el texto a la lista?», _ 291,
«Pregunta»)
  If Respuesta = 6 Then
     Lista.AddItem Texto.Text
  End If
End Sub
```

En el momento que escribimos el símbolo _ al final de una línea, **Visual Basic** entiende que la siguiente línea de código va seguida. No la entiende como líneas separadas. Tú puedes escribir el código en una misma línea. Una cosa que deberás tener en cuenta es que antes de este símbolo deberá existir un espacio en blanco.

8. Inicia una ejecución de prueba.

9. Escribe algo en la casilla de texto.

*10. Pulsa el botón **Insertar**.*

*11. Seguidamente aparecerá el **MsgBox** que hemos creado.*

Observa como el botón **No** aparece remarcado. Si pulsamos **Intro** este será el botón que actuará.

*12. Pulsa **Intro** y observa como no ocurre nada. (Más adelante insertaremos el código para que aparezca el otro cuadro de mensaje)*

*13. Vuelve a pulsar el botón **Insertar**.*

*14. Ahora pulsa en el botón **Sí**.*

El **MsgBox** desaparecerá y el texto pasará a estar dentro de la **lista**.

Ahora vamos a insertar el código necesario para que nos aparezca el segundo mensaje.

15. Modifica el código del botón para que quede de esta forma:

```
Private Sub Insertar_Click()
   Respuesta = MsgBox(«¿Seguro que deseas añadir el texto a la lista?», _ 291,
«Pregunta»)
  If Respuesta = 6 Then
     Lista.AddItem Texto.Text
  End If
 If Respuesta = 7 Then MsgBox «No se añadirá el texto a la lista», 0, _ «Mensaje»
End Sub
```

*16. Observa detenidamente las diferencias que existen entre los dos tipos de **MsgBox** que hemos colocado en nuestro código.*

Como en el primer mensaje nos interesa conocer cual es la tecla que ha pulsado el usuario, ponemos todas las opciones del **MsgBox** entre paréntesis y además asignamos esta estructura a una variable.

En cambio en el segundo **MsgBox** no nos interesa saber el valor del botón pulsado con lo que no asignamos ninguna variable.

Observa también que hemos puesto como valor ***0*** ya que solo queremos que aparezca un botón y ningún icono.

17. Haz una ejecución de prueba.

*18. Escribe algo en el **TextBox**, pulsa en **Insertar**.*

19. Contesta afirmativamente.

*20. Vuelve a pulsar en **Insertar**.*

*21. Ahora contesta **Negativamente** y observa como aparece un nuevo **MsgBox**.*

*22. Acepta el **MsgBox** actual y finaliza la ejecución.*

Vamos a depurar un poco el código de esta aplicación. Para facilitar la lectura del código vamos a crear unas constantes que tengan como valor **6** y **7** para que así durante el código no tengamos que estar pensando a que botones pertenecen dichos valores.

*23. Define dos constantes, a la primera le llamamos **Sí** y le asignamos como valor **6** y a la segunda le llamamos **No** y le asignamos un valor de **7**.*

Recuerda donde deberás declarar dichas constantes.

```
Const Sí = 6
Const No = 7
```

Ahora vamos a pasar a cambiar el código de la aplicación para que quede un poco más entendible:

24. Accede al código del botón y realiza los cambios pertinentes para que quede como el siguiente código:

```
Private Sub Insertar_Click()
    Respuesta = MsgBox(«¿Seguro que deseas añadir el texto a la lista?», _ 291, «Pregunta»)
   If Respuesta = Sí Then
      Lista.AddItem Texto.Text
   End If
  If Respuesta = No Then MsgBox «No se añadirá el texto a la lista», 0, _ «Mensaje»
End Sub
```

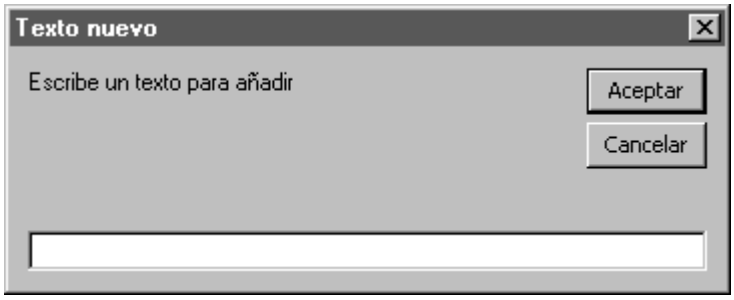
Ahora el código queda un poco más comprensible ya que no aparecen valores por medio.

Observa que en este código no hemos escrito nada para cuando el usuario pulsa el botón **Cancelar** ya que no deseamos que se realice ningún tipo de acción.

Una vez visto estos mensajes, vamos a ver como podemos introducir datos a través de otro tipo de mensajes.

Solicitud de datos (InputBox)

Vamos a ver una forma de pedir al usuario datos utilizando un nuevo tipo de ventana de mensajes.



En un **InputBox** sólo podremos modificar el texto que aparece en el **mensaje**, el **título** de la ventana de mensaje, si deseamos que aparezca algún tipo de **cadena** como predeterminada y la **posición** de la pantalla en la que deseamos que aparezca dicha ventana.

En la imagen anterior vemos un ejemplo de **InputBox** con las diferentes partes que lo componen.

Sintaxis de un InputBox

Al utilizar este tipo de ventana tenemos que asignar el contenido del cuadro de mensaje a una variable donde se almacenará lo que el usuario escriba dentro del **InputBox**.

Si el usuario pulsa **Aceptar**, el contenido del cuadro de texto pasará a la variable asignada para este efecto, mientras que si el usuario pulsa en **Cancelar** no se añade nada a variable.

Variable = InputBox (Mensaje, Título)

Vamos a ver como podemos trabajar con un **InputBox**.

Generar un InputBox

*25. Borra el **TextBox** que teníamos en el formulario que hemos estado utilizando anteriormente.*

*26. Accede al código del botón **Insertar** y borra todo el código que habíamos escrito anteriormente.*

Lo que vamos a pretender ahora es que al pulsar el botón **Insertar** nos aparezca un **InputBox** como el que hemos visto anteriormente. Dentro de este **InputBox** escribiremos el texto que deseamos añadir a la lista. Al pulsar **Aceptar** este texto pasará a la lista, mientras que si pulsamos en el botón **Cancelar** no ocurrirá nada.

Primero vamos a ver que debemos hacer para que nos aparezca el **InputBox** que hemos visto anteriormente.

*27. Escribe el siguiente código dentro del botón **Insertar**.*

```
Private Sub Insertar_Click()
   Nuevo = InputBox(«Escribe un texto para añadir», «Texto nuevo»)
End Sub
```

Observa que creamos una variable llamada **Nuevo** en la que almacenamos lo que escribe el usuario de la aplicación dentro del **InputBox**.

*28. Realiza una ejecución de prueba y pulsa sobre **Insertar**.*

*29. Pulsa en **Aceptar**.*

Podrás ver que no ocurre nada, ya que no hemos escrito el código para añadir el texto escrito en el **InputBox** dentro de la lista.

*30. Detén la ejecución y accede al código del botón **Insertar**.*

31. Modifica el código para que quede de la siguiente forma:

```
Private Sub Insertar_Click()
   Nuevo = InputBox(«Escribe un texto para añadir», «Texto nuevo»)
   Lista.AddItem Nuevo
End Sub
```

32. Haz una ejecución de prueba.

*33. Pulsa en **Insertar**.*

*34. Escribe cualquier cosa dentro del **InputBox**.*

*35. Pulsa en **Cancelar**.*

Observa como aparentemente no ha pasado nada.

*36. Pulsa nuevamente en **Insertar**.*

*37. Vuelve a escribir algo dentro del **InputBox**.*

*38. Ahora pulsa en **Aceptar**.*

Podemos ver como se ha añadido el texto en la lista, pero no en la primera posición. ¿Por qué ha ocurrido esto?

Esto ocurre porque en el momento que nosotros pulsamos el botón **Cancelar** del **InputBox** se le asigna un espacio en blanco a la variable y esto es lo que pasamos a añadir a la lista en la línea siguiente.

Depurando el código

Vamos a ver como podemos aprovechar la cualidad de asignar un espacio en blanco al pulsar el botón **Cancelar** para depurar la aplicación.

En el momento en el que cuando el usuario pulsa el botón **cancelar** no se debería añadir nada en la lista, esto lo podemos solucionar preguntando si la variable que se genera en el **InputBox** es diferente a **“”** con lo que se añadirá el texto a la lista. Vamos a ver como podemos hacer esto.

*39. Detén la ejecución de la aplicación y accede al código del botón **Insertar**.*

40. Modifícalo para que quede así:

```
Private Sub Insertar_Click()
   Nuevo = InputBox(«Escribe un texto para añadir», «Texto nuevo»)
   If Nuevo <> «» Then Lista.AddItem Nuevo
End Sub
```

41. Ejecuta la aplicación y observa como funciona.

Añadir sin parar

Imagina que deseas añadir utilizando este sistema varios elementos a la lista. Si tenemos el código como hasta este momento, para añadir una nueva entrada a la lista deberíamos ir pulsando consecutivamente a **Insertar** y después a **Aceptar** vamos a ver un sistema, utilizando un bucle, para que se repita la aparición de un **InputBox** y la inserción del elemento escrito a la lista hasta que el usuario no escriba nada dentro del **InputBox**.

42. Modifica el código para que quede de la siguiente forma:

```
Private Sub Insertar_Click()
   Do
      Nuevo = InputBox("Escribe un texto para añadir", "Texto nuevo")
      If Nuevo <> «» Then Lista.AddItem Nuevo
   Loop Until Nuevo = ""
End Sub
```

Este bucle nos repetirá la instrucción hasta que pulsamos la tecla **Cancelar** o **Aceptar** teniendo el cuadro de texto vacío.

43. Realiza una ejecución de prueba.

*44. Accede al botón **Insertar**.*

*45. Escribe cualquier cosa, pulsa en **Aceptar**.*

Seguidamente aparecerá otro **InputBox** con el cuadro de texto vacío. Si miras la lista podrás ver como el texto anterior se ha añadido. Si no ves la lista puedes mover el **InputBox** como si se tratase de cualquier otra ventana de **Windows**.

En el momento en el que no desees introducir más palabras a nuestra lista pulsa **Aceptar** sin haber escrito nada en el **InputBox**.

46. Detén la ejecución de prueba.

Texto por defecto

Si dentro del cuadro de texto de nuestro **InputBox** deseamos que aparezca algún tipo de texto por defecto lo podemos hacer de una forma muy sencilla.

Imagina que deseamos que en nuestra aplicación, siempre que aparece el **InputBox** aparezca la palabra **Texto** dentro del cuadro de texto.

*47. Accede al código del botón **Insertar**.*

*48. Modifica el código del **InputBox** para que quede de la siguiente forma:*

```
Nuevo = InputBox(«Escribe un texto para añadir», «Texto nuevo», «Texto»)
```

*49. Ejecuta la aplicación y pulsa en **Insertar**.*

Observa como aparece la palabra **Texto** seleccionada dentro del **InputBox**. Si pulsamos en **Aceptar** la palabra **Texto** pasará a la lista, si no nos interesa esta palabra la podemos sustituir por la que queramos. Si pulsamos en **Cancelar** no se añadirá nada a la lista.

Ahora vamos a ver como podemos colocar el **InputBox** en diferentes lugares de

la pantalla.

Cambiar la posición del InputBox

Si no indicamos en que posición deseamos situar el **InputBox** aparecerá en el centro de la pantalla. Pero puede ser que al estar situado en el centro nos oculte algún dato importante del formulario con el que estamos trabajando, con lo que podremos indicar en que lugar de la pantalla deseamos que aparezca.

El primer valor que introduciremos es la distancia entre el borde izquierdo del **InputBox** y el borde izquierdo de la pantalla.

El segundo valor es la distancia entre el borde superior del borde del **InputBox** con la parte superior de la pantalla.

50. Realiza los pasos necesarios para que el formulario aparezca centrado en la pantalla.

*51. Ahora coloca los valores necesarios dentro del **InputBox** para que al aparecer este podamos ver con claridad la lista del formulario.*

De esta manera podremos ver como se añaden los valores escritos dentro del **InputBox** en la lista sin necesidad de mover esta por la pantalla.

Por ejemplo:

```
Nuevo = InputBox(«Escribe un texto para añadir», «Texto nuevo», _ «Texto», 3000, 1500)
```

52. Realiza todas las ejecuciones de prueba que necesites hasta que consigas encontrar el lugar ideal.

Si te fijas en esta estructura estamos utilizando 5 parámetros diferentes, pero que pasaría por ejemplo si no deseamos que aparezca un texto predefinido.

Ausencia de elementos

Imagina que en el código anterior deseamos que el **InputBox** aparezca en un lugar determinado de la pantalla, pero no deseamos que aparezca un texto predeterminado.

Este problema se soluciona muy fácil. Solo deberás hacer como si estuviera el parámetro que quitamos respetando así la cantidad de comas que existen dentro del **InputBox** con todos los parámetros escritos.

Vamos a ver como quedaría el código, sin que aparezca **Texto** como palabra determinada.

```
Nuevo = InputBox(«Escribe un texto para añadir», «Texto nuevo», , 3000, _ 1500)
```

Observa como antes del valor de posición horizontal existen dos comas. Entre estas comas es donde estaba escrita la palabra que aparecía como predefinida en el **InputBox**.

En esta lección hemos aprendido como utilizar cuadros de diálogo de una forma fácil y rápida. Estos elementos se deben usar para hacer que el usuario encuentre la aplicación lo más fácil posible sin tener que estar intuyendo para que se utilizan los botones y los objetos que aparecen en ella.

Recomiendo utilizar los **MsgBox** para aclarar todo lo que se pueda, los errores y

las decisiones que debe tomar el usuario en determinados momentos. A partir de este momento espero que formen parte de las aplicaciones que realices y te familiarices con su funcionamiento.

Fin lección 8

LECCIÓN 7

En esta lección vamos ver una de las herramientas más comunes dentro de las aplicaciones que se utilizan en el entorno de Windows, los menús.

Concepto de menú

Para ver las partes de los menús y como podemos trabajar con ellos vamos a ver algunos ejemplos dentro de Visual Basic **.**

. Práctica 1

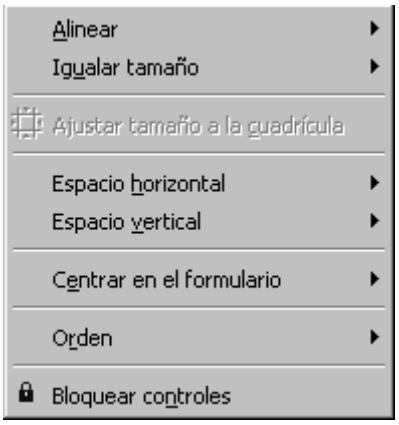
*1. Inicia **Visual Basic** con un formulario vacío.*

*2. Observa detenidamente la **barra de menús**.*

Podemos ver que a lo largo de esta barra de menús **aparecen una serie de palabras. Estas son los diferentes** títulos **de menús. Dentro de cada** título **tenemos un menú diferente.**

Vamos a desplegar uno de estos menús.

*3. Haz clic sobre **Formato**.*



Acto seguido aparecerá un menú como este:

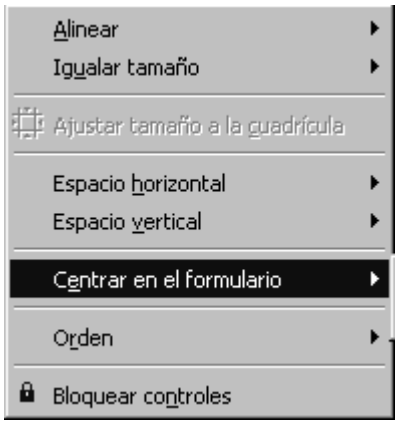
Dentro de este menú podemos encontrar una serie de opciones que al hacer clic sobre ellas realizarán una determinada acción. Veamos algunos ejemplos: Ajustar tamaño a la cuadrícula, Bloquear controles **.**

Si apareciese alguna opción con puntos suspensivos **en su parte derecha nos indican que al hacer clic sobre ellas nos aparecerá alguna ventana de dialogo.**

Las opciones de los menús que en su parte derecha tienen escrito, por ejemplo: Ctrl+T **nos indican la combinación de teclas que realizarán esta misma acción sin necesidad de abrir ningún menú ni seleccionar dicha opción.**

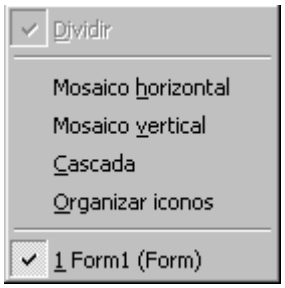
Las opciones que están de color gris **están desactivadas, no podemos utilizarlas.**

Existen otras opciones que tienen en su parte derecha una pequeña figura. Si nos situamos sobre una de estas opciones veremos como automáticamente nos aparece otro pequeño menú. Dentro de este submenú podemos encontrar muchas más opciones o incluso más submenús relacionadas con la opción principal.



Dentro de nuestro menú también podemos observar que existen líneas divisorias. Estas líneas dividen opciones dentro de un mismo menú.

Un elemento también importante que podemos encontrar dentro de las opciones que componen un menú son las teclas de acceso. Observa como la opción Espacio vertical, la letra v aparece subrayada, esto quiere decir que para activar esta opción de una forma rápida podremos pulsar Alt+v. De esta forma la opción se activará sin necesidad de utilizar el ratón o las flechas de control y la tecla Intro.



Un tipo de elemento que no aparece en este menú es la marca de verificación que nos indica si una opción está o no activada. Podemos ver un ejemplo en el menú Ventana. Observa la opción 1 Form1 (Form), tiene un símbolo a su izquierda que nos indica, en este caso, que el formulario Form1 está activo.

Ahora que ya tenemos un poco más claros los diferentes elementos que pueden formar parte de un menú vamos a realizar una práctica para aprender como trabajar con ellos y como crear menús en nuestras aplicaciones.

Es recomendable utilizar menús simples y claros, de esta forma facilitaremos el control a todos los usuarios de la aplicación.

Menús principales

Antes de seguir trabajando vamos a explicar un poco en que consistirá nuestra aplicación de ejemplo.

Nosotros dispondremos de un formulario con solo dos objetos, un TextBox **y un** ListBox **.**

En el ListBox **tendremos una serie de nombres ya escritos. Nosotros podremos añadir nuevos nombres en la lista utilizando el** TextBox **. Tendremos opciones en el** menú **para añadir el nombre a la lista y para borrar dicho nombre.**

Con los elementos de la lista **también trabajaremos, ya que podremos borrar alguno de los elementos o la lista completa, podremos bloquear la lista para que no se pueda ni borrar ni agregar elementos, y añadiremos una opción para cambiar el tamaño del texto de nuestra lista.**

Todo esto utilizando solamente las opciones del menú.

Ahora que ya tenemos un poco claro de que va nuestra pequeña aplicación de ejemplo, vamos a empezar a crear la estructura de menús.

Editor de menús

Para crear los diferentes menús que necesitaremos en una aplicación utilizaremos el Editor de menús **. Esta herramienta nos permitirá crear toda la estructura de menús de forma sencilla.**

. Práctica 2

*1. Inicia **Visual Basic 6.0** con un formulario en blanco.*

*2. Accede a la opción **Editor de menús** dentro del menú **Herramientas**.*

También puedes poner en funcionamiento el Editor de menús **utilizando la combinación de teclas** [Control] + [E] **o utilizando en la** Barra de herramientas estándar **el siguiente botón:**

Observa la nueva ventana que nos aparece en pantalla:

Vamos a comentar las principales partes de las que consta este Editor de menús **. Las demás las iremos viendo conforme las necesitemos.**

En los menús, como en la gran mayoría de objetos que forman parte de Visual Basic **, las dos principales propiedades son el** Name **y el** Caption **. El** Name **, será el nombre que utilizaremos para hacer referencia al control del menú a lo largo de toda la aplicación y el** Caption **será el texto que aparecerá en el menú y que será por el cual se debe guiar el usuario. Piensa que el** Caption **debe ser corto y lo suficiente explicativo como para que el usuario entienda que es lo que pasa cuando se utiliza este control.**

Título de menú

En nuestra aplicación vamos a necesitar dos menús diferentes. Uno que gestionará el TextBox **y otro el** ListBox **.**

Vamos a crear los dos títulos de menú **.**

El que gestionará el TextBox **vamos a llamarle** Nombre **ya que aquí es donde escribiremos los nombres para insertarlos en la lista y al menú del** ListBox **le llamaremos** Lista **.**

3. Sitúate sobre la casilla ***Caption****.*

4. Escribe ***&Nombre***

Recuerda que el símbolo & **se utiliza para crear una tecla de acceso. En este caso la tecla de acceso al menú** Nombre **sería la** N (Alt + N) **.**

Observa que mientras escribes, la palabra Nombre **también aparece en el recuadro inferior de la ventana. A este cuadro le llamaremos** Cuadro de lista **. En este cuadro vamos a ir viendo una representación de las opciones que vamos insertando en nuestros menús.**

5. Pasa a la casilla ***Name****.*

Recuerda que aquí escribiremos el nombre con el que haremos referencia a este menú durante el código de la aplicación.

6. Escribe ***Nombre****. (Utiliza siempre nombres que te sean fáciles de recordar).*

7. Pulsa el botón ***Siguiente****.*

Observa como en el cuadro de lista **la franja azul de selección pasa a la siguiente línea.**

*8. Sitúate sobre el **Caption** y escribe **&Lista**.*

*9. Ahora en el **Name** escribe **Lista**.*

Ahora ya tenemos creados los dos Títulos de menú **. Vamos a ver como quedan dentro de nuestro formulario.**

*10. Pulsa el botón **Aceptar**.*

Acto seguido estaremos de nuevo en el formulario de nuestra aplicación. Observa como han aparecido los dos títulos de menú que hemos creado anteriormente.

Antes de seguir trabajando con los menús vamos a colocar en nuestro formulario los dos objetos que necesitamos para llevar a cabo la aplicación.

*11. Sitúa donde quieras un **TextBox**.*

*12. Borra su contenido y ponle como **(Nombre): EntradaNombre**.*

*13. Sitúa donde quieras un **ListBox**.*

*14. Llámale **ListaNombres** y coloca en su interior **6 nombres** de persona. Repasa lecciones anteriores.*

Interior de un menú

Vamos a crear el contenido del menú Nombre **.**

*15. Vuelve a abrir el **Editor de menús**. Utiliza el método que prefieras.*

*16. Sitúate sobre el **cuadro de lista** en la palabra **Lista**.*

*17. Pulsa el botón **Insertar**.*

Observa como se ha creado un espacio en blanco entre Nombre **y** Lista **. Aquí vamos a crear las opciones que irán dentro del menú** Nombre **.**

Con los nombres que introduzcamos dentro de nuestro TextBox **vamos a realizar tan solo dos posibles operaciones. La primera sería: pasar el contenido del** TextBox **a la lista y la segunda: borrar el contenido del** TextBox **. Para ello vamos a crear dos opciones dentro del menú** Nombre **, la primera será** Añadir **y la segunda** Borrar **.**

*18. Sitúate sobre la casilla **Caption** y escribe **&Añadir**.*

*19. Ponle como **Name**: **NombreAñadir**.*

Vamos a tomar como costumbre poner nombres que nos ayuden a identificar rápidamente a donde pertenece esta opción. Con NombreAñadir **podremos ver que** Añadir **está dentro de la opción** Nombre **. De esta forma también podremos hacer distinción entre la opción** Añadir **que esté dentro de** Nombre **y otra opción a la que podremos poner el mismo** Caption **pero no el mismo** Name **en otro menú cualquiera.**

Si observas el cuadro de lista, podrás ver que la opción Añadir esta a la misma altura que Nombre y que Lista, cosa que no nos interesa. A nosotros nos interesaría que Añadir esté dentro de la opción Nombre. Vamos a ver como podemos arreglar esto.

Observa en el Editor de menús que disponemos de 4 botones con flechas en su interior. Vamos a ver para que se utilizan cada una de ellas.

Empezaremos a explicar de izquierda a derecha: la primera flecha sirve para bajar de nivel, la segunda para aumentar de nivel, la tercera para mover un menú a posiciones superiores y la cuarta para mover un menú a posiciones inferiores.

Vamos a ver estas opciones en funcionamiento.

Aumentar un nivel

Vamos ha hacer que la opción Añadir esté dentro del menú Nombre.

20. Haz un clic sobre el botón que tiene una flecha que apunta hacia la derecha.

Observa el cuadro de lista. Verás que en la opción Añadir han aparecido cuatro puntos a su derecha. Esto nos indica que Añadir ya forma parte de Nombre.

Vamos a ver como ha quedado nuestro menú en el formulario.

*21. **Acepta** el **Editor de menús**.*

*22. Haz un clic sobre el menú **Nombre**. Dentro de él aparecerá la opción **Añadir**.*

Vamos a colocar la segunda opción que debe estar dentro de Nombre.

*23. Abre nuevamente el **Editor de menús**.*

*24. Colócate sobre **Lista**.*

*25. Pulsa en **Insertar**.*

*26. Sitúate en la casilla **Caption** y escribe **&Borrar**.*

*27. Como **Name** escribe: **NombreBorrar***

Como esta opción también debe ir dentro del menú Nombre, deberemos aumentar el nivel de la opción Borrar.

28. Pulsa sobre la flecha que apunta hacia la derecha.

*29. **Acepta** el **Editor de menús**.*

Introducir código en los menús

Como si se tratase de cualquier otro objeto, las diferentes opciones de nuestros

menús también tienen eventos y también se puede escribir código en su interior.

*30. Abre el menú **Nombre**.*

*31. Haz un clic sobre la opción **Borrar**.*

Acto seguido te aparecerá la ventana del editor de código **.**

32. Escribe el siguiente código:

```
Private Sub NombreBorrar_Click()
   EntradaNombre.Text = «»
End Sub
```

Con esto lo que conseguiremos es borrar el contenido del objeto EntradaNombre **.**

33. Cierra la ventana de código.

*34. Haz un clic sobre la opción **Añadir** dentro del menú **Nombre**.*

35. Escribe el siguiente código:

```
Private Sub NombreAñadir_Click()
   ListaNombres.AddItem (EntradaNombre.Text)
End Sub
```

Con este código lo que conseguiremos es que el contenido del objeto EntradaNombre **se añada a la** ListaNombres **.**

Vamos a ver estas opciones en funcionamiento.

36. Realiza una ejecución de prueba.

*37. Escribe cualquier nombre en **EntradaNombre**.*

*38. Abre el menú **Nombre** y escoge la opción **Añadir**.*

Observa como el contenido de EntradaNombre **pasa a formar parte de la lista.**

*39. Abre el menú **Nombre** y escoge la opción **Borrar**.*

El texto que hay en EntradaNombre **desaparece, se borra.**

En un principio todo funciona bien, pero vamos a ver que ocurre en este caso:

*40. Con la **EntradaNombre** vacía, selecciona la opción **Añadir**.*

Aparentemente no ocurre nada.

*41. Escribe un nombre en **EntradaNombre** y añádelo a la lista.*

Como puedes observar, en el paso 40 lo que ha ocurrido es que hemos añadido un espacio en blanco a la lista, cosa que no nos interesa. Tendremos que pensar algo para que el usuario no introduzca elementos vacíos en la tabla.

Activar y desactivar menús

Vamos a ver como podemos *activar* **y** *desactivar* **un menú cuando a nosotros nos interese. Esto siempre dependerá del "estado" en el que se encuentra la aplicación.**

Vamos a desactivar todo el menú Nombre **en el momento en el que** EntradaNombre **no contenga nada en su interior y vamos a activarlo nuevamente cuando el usuario escriba cualquier cosa.**

42. Detén la ejecución del programa.

43. Pulsa doble clic sobre el objeto ***EntradaNombre****.*

De esta forma abriremos el evento Change **. Evento que se pondrá en funcionamiento cada vez que se modifique el contenido de** EntradaNombre **.**

44. Escribe el siguiente código:

```
Private Sub EntradaNombre_Change()
   If Len(EntradaNombre.Text) <> 0 Then
      Nombre.Enabled = True
   Else
      Nombre.Enabled = False
   End If
End Sub
```

Este código realiza lo siguiente: cuando se produce un cambio en EntradaNombre **miramos el tamaño de este objeto. Esto lo haremos utilizando la instrucción** Len() **. Dentro de los paréntesis escribiremos el objeto al que queremos mirar el tamaño. Si el tamaño es** diferente de 0**, quiere decir que hay algo, entonces hacemos que el menú** Nombre **esté activado, mientras que si el tamaño es** igual a 0 **desactivamos el menú. De esta manera controlamos que el usuario no haga clic en este objeto.**

En el momento que se inicia la ejecución el objeto EntradaNombre **está vacío, pero el menú** Nombre **está activado. Esto es así porque todavía no se ha entrado en el evento** Change **del objeto** EntradaNombre **y no se han realizado las instrucciones que hemos escrito anteriormente. Vamos a ver que podemos hacer para que desde un principio este objeto esté desactivado.**

45. Accede al ***Editor de menús****.*

46. Sitúate sobre la opción ***Nombre****.*

47. Busca esta opción dentro del ***Editor de menús****:*

48. Haz clic sobre ella.

49. ***Acepta*** *el* ***Editor de menús****.*

Observa como en nuestro formulario aparece la opción Nombre **de color gris. En este momento ya no tenemos acceso a este objeto hasta que cambiemos la opción** Enabled **, ya sea desde el código o desde el** Editor de menús **.**

Vamos a terminar de introducir las opciones que formarán parte del menú Lista

50. Abre nuevamente el ***Editor de menús****.*

51. Sitúate en la siguiente línea de ***Lista****.*

52. Escribe en el ***Caption****:* ***&Borrar lista****.*

*53. Escribe en **Name**: **ListaBorrar**.*

54. Aumenta su nivel.

*55. Pulsa en **siguiente**.*

Observa como el siguiente objeto que insertemos ya tendrá el mismo nivel que Borrar Lista **.**

*56. Escribe en el **Caption**: **Borrar &elemento**.*

*57. Escribe en **Name**: **ListaBorrarElem**.*

Ahora ya tenemos dos objetos que forman parte del menú Lista

Como ya hemos explicado al principio de esta misma lección en muchas ocasiones se utiliza una línea horizontal de separación para dividir opciones dentro de un mismo menú. Vamos a ver como podemos colocar nosotros una línea como esta dentro de nuestro menú.

Líneas de separación

*58. Pulsa el botón **Siguiente**.*

*59. Escribe en el **Caption** un guión: **-***

Como cualquier otro objeto deberá tener nombre, aunque no podamos modificar sus propiedades.

*60. Escribe en **Name**: **ListaLinea***

*61. **Acepta** el **Editor de menús**.*

*62. Despliega el menú **Lista** y observa como en la última posición ha aparecido una línea horizontal que ocupa todo lo ancho del menú.*

*63. Accede nuevamente al **Editor de menús**.*

64. Sitúate en la línea siguiente del último objeto.

*65. Escribe en el **Caption**: **&Proteger**.*

*66. Escribe en **Name**: **ListaProteger**.*

*67. Pulsa en **siguiente**.*

*68. Escribe en el **Caption**: **-**.*

*69. Y como **Name**: **ListaLinea2**.*

*70. Pulsa en **siguiente**.*

*71. Escribe en el **Caption**: **&Tamaño**.*

*72. Escribe en **Name**: **ListaTamaño**.*

Más adelante veremos para que utilizaremos las diferentes opciones que hemos puesto en nuestro menú e indicaremos el código que deberemos escribir dentro.

Creación de submenús

Vamos a crear un submenú dentro de la última opción que hemos insertado en el menú Lista **.**

*73. Pulsa en **siguiente**.*

*74. Escribe en el **Caption**: **8**.*

*75. Y como **Name**: **ListaTamaño8**.*

Ahora para que esta opción forme parte de un submenú de la opción Tamaño deberemos aumentar el nivel, con lo que en la parte izquierda de esta aparecerán 8 puntos.

76. Aumenta de nivel esta opción.

*77. **Acepta** el **editor de menús**.*

*78. Sitúate sobre la opción **Lista**.*

79. Despliega el menú.

Aparecerá un menú igual a este:

Observa como en la parte derecha de la opción Tamaño aparece una pequeña punta de flecha. Esto nos marca que en esta opción existe un submenú.

*80. Sitúate sobre la opción **Tamaño** y observa lo que pasa.*

A la derecha de la opción Tamaño ha aparecido el último elemento que insertamos.

Vamos a terminar de colocar los últimos elementos que forman parte de nuestros menús.

*81. Abre nuevamente el **Editor de menús**.*

*82. Sitúate en la línea siguiente al **8**.*

*83. Escribe en el **Caption**: **12**.*

*84. Escribe en **Name**: **ListaTamaño12**.*

85. Si es necesario aumenta su nivel hasta alcanzar el mismo que la opción anterior.

*86. Pulsa en **Siguiente**.*

*87. Escribe en el **Caption**: **18**.*

*88. Escribe en **Name**: **ListaTamaño18**.*

Como te puedes imaginar vamos a utilizar estas tres opciones para cambiar el tamaño de letra de los objetos de la lista. Vamos a utilizar una nueva característica que nos brindan los menú, la marca de verificación **.**

Marca de verificación

Las marcas de verificación **será una pequeña señal que aparecerá en la parte izquierda de una opción del menú. Esta marca nos servirá para saber si esta opción está o no activada. En nuestro caso en el momento que cambiemos el tamaño de la lista aparecerá una marca indicando cual es el tamaño actual de la lista.**

Vamos a marcar una de las opciones inicialmente que será exactamente el tamaño de letra que tiene al iniciar la aplicación nuestra Lista

Antes de realizar la marca vamos a mirar el tamaño de letra que tiene nuestra Lista **.**

89. Selecciona el objeto lista.

*90. Accede al cuadro de diálogo **Fuente**. (Mira lecciones anteriores).*

*91. Si es necesario pon el tamaño de la fuente a **8**.*

92. Acepta el cuadro de diálogo actual.

Ahora vamos a activar la opción Tamaño – 8 **, para indicar que este es el tamaño de fuente actual de la lista.**

93. Selecciona el formulario.

*94. Accede al **Editor de menús**.*

*95. Sitúate sobre el **8** del **cuadro de lista** y haz un clic sobre la opción **Checked**.*

Aparentemente no ha ocurrido nada.

96. Acepta el Editor de menús.

✔ 8

*97. Abre el menú **Lista**, dentro de él abre el submenú **Tamaño** y observa como al lado*

*del **8** aparece una marca como esta:*

Vamos a ver como podemos activar y desactivar marcas de verificación utilizando el código.

*98. Haz un clic sobre el **Tamaño – 12**.*

99. Escribe el siguiente código:

```
Private Sub ListaTamaño12_Click()
   ListaTamaño8.Checked = False
   ListaTamaño12.Checked = True
   ListaTamaño18.Checked = False
   ListaNombres.FontSize = 12
End Sub
```

Con este código estamos *activando* **la opción** ListaTamaño12 **y estamos** *desactivando* **las demás. Al desactivar las demás opciones nos ahorramos mirar de que opción venimos.**

Después lo que hacemos es cambiar 12 **el tamaño de la letra de la lista.**

100. Realiza una ejecución de prueba.

*101. Cambia el **tamaño** de la lista a **12** utilizando las opciones del menú.*

102. Observa como el objeto lista a cambiado de tamaño.

Esto es debido a que este objeto se adapta automáticamente para que ninguna de las líneas queden cortadas.

103. Detén la ejecución.

*104. Selecciona el **ListBox**.*

105. Accede a sus propiedades.

*106. Cambia la propiedad **IntegralHeigth** a **False**.*

A partir de este momento cuando cambiemos el tamaño de letra de la lista no reducirá su tamaño.

Vamos a terminar de introducir el código para el cambio de tamaño.

*107. Haz un clic sobre **Tamaño – 8**.*

108. Escribe el siguiente código:

```
Private Sub ListaTamaño8_Click()
   ListaTamaño8.Checked = True
   ListaTamaño12.Checked = False
   ListaTamaño18.Checked = False
   ListaNombres.FontSize = 8
End Sub
```

109. Cierra el editor de código.

*110. Haz un clic sobre **Tamaño – 18**.*

111. Escribe el siguiente código:

```
Private Sub ListaTamaño18_Click()
   ListaTamaño8.Checked = False
   ListaTamaño12.Checked = False
   ListaTamaño18.Checked = True
   ListaNombres.FontSize = 18
End Sub
```

Observa el código de las tres opciones de tamaño. Ahora ya podemos cambiar el tamaño de letra de nuestra lista de nombres utilizando nuestro menú.

Activar y desactivar Submenús

Ahora vamos a pasar a escribir el código de la opción Proteger . Dentro de esta opción lo que queremos es que la lista de nombres quede bloqueada de tal forma que no se pueda hacer nada con ella. Para ello deberemos impedir que el usuario tenga acceso a alguna de las opciones de nuestro menú.

En este trozo de código volveremos a utilizar una partícula que ya vimos en lecciones anteriores (Not).

*112. Accede a la opción **Proteger** dentro del menú **Lista** y escribe:*

```
Private Sub ListaProteger_Click()
   ListaNombres.Enabled = Not (ListaNombres.Enabled)
   ListaProteger.Checked = Not (ListaProteger.Checked)
   NombreAñadir.Enabled = Not (NombreAñadir.Enabled)
   ListaBorrar.Enabled = Not (ListaBorrar.Enabled)
   ListaBorrarElem.Enabled = Not (ListaBorrarElem.Enabled)
   ListaTamaño.Enabled = Not (ListaTamaño.Enabled)
End Sub
```

Observa cada una de las líneas e intenta averiguar para que se utilizan.

Borrar lista

Vamos a escribir el código para borrar el contenido de la lista. La primera línea de este código no se ha explicado, se hará en lecciones posteriores. Solo escríbela y en el momento de ejecutar la aplicación ya comprobarás para que sirve.

*113. Accede a la opción **Borrar lista** y escribe el siguiente código:*

```
Private Sub ListaBorrar_Click()
   Respuesta = MsgBox(«¿Estás seguro?», 36, «Pregunta»)
   If Respuesta = vbYes Then
      ListaNombres.Clear
      Lista.Enabled = False
   End If
End Sub
```

Con la instrucción Clear después del nombre de la lista, eliminamos el contenido de todos los elementos que forman parte de ella. También desactivamos la lista para que no se pueda trabajar con ella, ya que no contiene ningún elemento.

114. Haz una ejecución de prueba.

115. Borra el contenido de la lista.

Observa el mensaje que te aparece de confirmación.

116. Una vez borrada, escribe un nombre dentro de la casilla reservada para este efecto.

117. Añade el nombre mediante la opción del menú.

Observa como el menú Lista no se ha activado y nos interesa que lo hubiera hecho ya que ahora ya existen elementos en la lista para poder trabajar con ella.

Vamos a añadir una línea de código en una de las opciones que ya tenemos escritas.

*118. Detén la ejecución y accede al código de **Añadir** del menú **Nombre**.*

Recuerda que si el menú Nombre está desactivado no podrás entrar dentro de ninguna opción. Primero deberás activarlo utilizando el editor de menús.

119. Modifica el código para que quede de la siguiente forma:

```
Private Sub NombreAñadir_Click()
   ListaNombres.AddItem (EntradaNombre.Text)
   Lista.Enabled = True
End Sub
```

La línea con el código en cursiva son las instrucciones que añadimos.

Borrar elemento

Vamos a ver como podemos eliminar un determinado elemento de la lista de nuestra aplicación.

*120. Accede a **Borrar elemento** dentro del menú **Lista**.*

121. Escribe el siguiente código que pasaremos a explicar a continuación:

```
Private Sub ListaBorrarElem_Click()
   If ListaNombres.ListIndex = -1 Then
      MsgBox "Debes seleccionar algún Elemento"
   Else
      ListaNombres.RemoveItem (ListaNombres.ListIndex)
   End If
End Sub
```

Antes de borrar algún elemento de la lista, nos vamos a asegurar que el usuario de la aplicación haya seleccionado algún nombre. Para ello utilizamos la instrucción ListIndex que nos devolvería el índice del elemento seleccionado. El índice, podríamos decir, que es la posición que ocupa el elemento seleccionado dentro de la lista. Es importante saber que el primer elemento de una lista tiene como índice el valor 0.

Nosotros en la primera línea de este código preguntamos si ListIndex es igual a -1, si el ordenador nos devuelve verdadero quiere decir que el usuario no ha seleccionado ningún elemento de la lista, con lo que mostraremos un mensaje de aviso. (Los mensajes los veremos en lecciones posteriores).

En cambio si ListIndex es diferente de -1 quiere decir que el usuario tiene seleccionado algún elemento con lo que ya podemos proceder al borrado. Esto lo haremos utilizando la instrucción RemoveItem. Entre paréntesis deberemos indicar el índice del elemento que ha seleccionado el usuario. La instrucción quedará de la siguiente forma: ListaNombres.RemoveItem (ListaNombres.ListIndex) .

Ahora ya podemos realizar ejecuciones de prueba para ver el funcionamiento de cada una de las opciones de nuestros menús.

Vamos a facilitar un poco el acceso a las diferentes opciones del menú, para ello utilizaremos las teclas de método abreviado **.**

Teclas de método abreviado

Las teclas de método abreviado **nos permiten ejecutar las instrucciones de una opción determinada sin necesidad de desplegar ningún menú.**

En nuestra aplicación vamos a utilizar esta propiedad en tan solo dos de las diferentes opciones: Añadir Nombre **y** Proteger la lista **. No es conveniente abusar demasiado con las combinaciones de teclas ya que podemos "liar" al usuario.**

*122. Accede a **Editor de menús**.*

*123. Sitúate sobre la opción **Añadir**.*

*124. Busca dentro de la ventana **Editor de menús** la opción **Shortcut**.*

*125. Despliega la lista y busca dentro de toda esta lista, que representa las combinaciones de teclas de las que disponemos, **Ctrl+A**.*

*126. Pulsa un clic sobre ella y observa el **Cuadro de lista**.*

Verás como en la parte derecha de esta opción aparecerá la combinación de teclas que hemos puesto.

*127. Pulsa un clic sobre la opción **Proteger**.*

*128. Despliega la lista de **Shortcut** y selecciona **Ctrl+P**.*

*129. **Acepta** el **Editor de menús**.*

Vamos a ver como funcionan estas nuevas propiedades.

130. Inicia una ejecución de prueba.

*131. Sin desplegar ningún tipo de menú pulsa **Ctrl+P**.*

Observa como la lista ha quedado protegida, de la misma forma que si hubiéramos accedido a la opción Proteger **dentro del menú** lista **.**

Si abres el menú de la aplicación podrás ver como en el menú aparecen las combinaciones de teclas que hemos marcado para estas opciones.

132. Realiza todas las pruebas que desees, utilizando las opciones del menú.

Seguidamente vamos a introducir otro tipo de menú muy utilizado dentro de los programas creados para Windows **. Los menús contextuales.**

Menú contextual

Antes de trabajar con los menús contextuales vamos a ver que son y para que sirven.

Un menú contextual aparece haciendo un clic con el botón derecho del ratón en alguna parte de la pantalla. Normalmente la gran mayoría de lugares de un programa contienen un menú contextual.

133. Detén la ejecución de la aplicación.

134. Sitúate sobre el formulario que estamos creando.

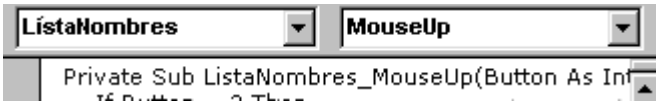
135. Haz clic con el botón derecho sobre él.

Seguidamente te aparecerá un nuevo menú con una serie de opciones. Estas opciones son las más utilizadas o las que nos puede interesar tener más a mano.

En nuestro ejemplo vamos a crear un menú contextual sobre la lista. De esta forma nos será mucho más fácil trabajar con las opciones que ya tenemos creadas dentro del menú Lista **.**

*136. Haz doble clic en el objeto **ListaNombres**.*

*137. Selecciona dentro de la lista de **procedimientos** de este objeto: **MouseUp**.*



Este evento se produce en el momento en el que el usuario suelta un botón del ratón.

Ahora nos interesaría introducir alguna instrucción que controlase si el usuario hace clic con el botón izquierdo del ratón o con el derecho. Recordemos que normalmente el menú contextual aparece haciendo clic con el botón derecho del ratón.

*138. Escribe el siguiente código dentro del evento **MouseUp**.*

```
If Button = 2 Then PopupMenu Lista
```

Vamos a comentar que es lo que hace el siguiente código.

Como ya hemos dicho anteriormente estas líneas de código se ejecutarán en el momento en el que el usuario pulsa y suelta **un botón del ratón. Con la instrucción** If **miramos cual de los dos botones ha pulsado el usuario.**

Cuando un usuario hace clic en uno de los botones del ratón, Visual Basic lo que hace es almacenar un valor en una variable llamada Button **. Si el valor de esta variable es el** 1 **el usuario ha pulsado el botón** izquierdo **, mientras que si el valor devuelto es un** 2 **el usuario ha pulsado el botón** derecho **.**

Para que se muestre el menú emergente o menú contextual utilizaremos la instrucción PopupMenu **seguido del nombre del menú que deseamos mostrar. En nuestro caso el menú que queremos ver es el llamado** Lista **.**

Con esta simple línea de código, situada dentro de este nuevo evento, conseguimos mostrar nuestro menú contextual. A partir de ahora en el momento en el que ejecutemos la aplicación, el botón izquierdo del ratón lo utilizaremos para seleccionar uno de los elementos de la lista, mientras que el botón derecho servirá para hacer aparecer nuestro menú contextual.

139. Realiza una ejecución de prueba y mira el funcionamiento de ambos botones dentro de nuestra lista.

Siempre que deseemos utilizar un menú contextual deberemos crearlo con el Editor de menús **. Si no deseamos que este aparezca en la barra de menús podremos**

ocultarlo.

Aunque este menú esté oculto podremos hacer que aparezca como menú contextual de la misma forma que hemos visto en esta lección.

Esta lección nos ha servido para ver las opciones más importantes de los menús, tanto en la barra de menús como los menús contextuales.

Fin lección 7

Lección 6

En esta lección vamos a familiarizarnos con las estructuras de repetición, las cuales nos sirven para realizar una misma instrucción un número determinado de veces o indeterminado dependiendo de una condición.

Introducción a las estructuras de repetición

El número de veces que se repetirá la instrucción o instrucciones puede depender de un contador o de una condición.

En esta lección vamos a ver los dos tipos de bucles: con contador o con condición.

For... Next

Esta es una estructura de repetición o bucle, la cual depende de un contador que nos controla el número de veces que se deberá repetir una o varias instrucciones.

En esta estructura siempre deberemos especificar la variable (**contador**), un **valor inicial** y un **valor final**. Normalmente el contador incrementará de uno en uno a no ser que nosotros indiquemos lo contrario.

La estructura del bucle utilizando un contador es la siguiente:

```
For Contador = Inicio To Fin [Step Incremento]
    [Instrucciones]
Next Contador
```

Vamos a explicar las diferentes partes de esta estructura:

Contador: Aquí es donde nosotros escribiremos el nombre de la variable que queremos utilizar como contador.

Inicio: Valor inicial de la variable.

Fin: Valor final de la variable. Cuando la variable llegue a este valor, el bucle no se volverá a realizar.

Step: Esta instrucción es opcional. Si no la ponemos el contador irá incrementando de uno en uno. Si especificamos un número detrás de **Step** hacemos que nuestro contador aumente un número determinado de pasos.

Incremento: Número que marcará los pasos que debe aumentar el contador. Este número puede ser tanto positivo como negativo. Eso sí, siempre deberemos tener cuidado con los valores iniciales y finales para que no se produzca ningún tipo de error. No podemos hacer, por ejemplo, que el valor inicial sea 10 y el final 1 siempre y cuando no pongamos como **step** un valor negativo.

Instrucciones: Aquí escribiremos la o las instrucciones que queremos que se repitan.

Next Contador: Línea que indica que se termina el bucle y hace que aumente el contador según el valor que nos indique **step** en caso de tenerlo.

. Práctica 1

Vamos a realizar una simple aplicación en el que utilizaremos una estructura de repetición utilizando un contador.

La aplicación consistirá en una simulación de una tirada de un dado.

Te iremos especificando que tipo de objetos deberás añadir en nuestro formulario y algunas de las propiedades que deberás cambiar. El aspecto de los objetos y su situación corren por tu cuenta. Puedes poner tantos objetos **Label** como quieras para aclarar para que sirven cada uno de los elementos insertados en el formulario.

*1. Sitúa en un formulario nuevo un **ListBox** al que deberás poner como **(Nombre): Dado**.*

Aquí será donde el ordenador nos muestre las diferentes tiradas que realizamos.

*2. Coloca un **CommandBotton**, que tendrá como **(Nombre)** y **Caption: Tirada**.*

Al pulsar este botón se realizarán las diferentes tiradas.

*3. Coloca un **TextBox** al que pondremos como **(Nombre): NumTiradas**. Borra el contenido que aparece por defecto dentro de este objeto.*

Aquí será donde indiquemos cuantas tiradas queremos realizar.

Una vez colocados los objetos vamos a pensar en el código.

Nosotros en esta práctica queremos que se realicen tantas tiradas de dado como nos indique el usuario dentro del **TextBox**. Para esto nos interesaría crear una estructura de repetición que debería empezar en **1** y terminar en el **número que indica el usuario**. Los incrementos que sufrirá el contador deberá ser de uno en uno, por lo que la parte del **step** no la especificaremos.

4. Haz doble clic dentro del botón y escribe el siguiente código.

```
For Contador = 1 To NumTiradas.Text
   Dado.AddItem (Int(6*Rnd)+1)
Next Contador
```

En la primera línea de este pequeño código, que más adelante depuraremos, hemos iniciado el **contador** (nueva variable) a 1. No hace falta que definamos la variable. Al no definirla esta es de tipo Variant[1].

En esta primera línea también definimos en que valor queremos que termine el bucle. Este valor será el valor que introduzca el usuario dentro del **TextBox**.

Generar valores aleatorios

En la segunda línea, **Dado.AddItem (Int(6*Rnd)+1))**, hacemos que **Visual Basic** nos busque un valor aleatorio. Esto lo conseguimos con la instrucción **Rnd**. Nosotros como queremos conseguir un número aleatorio dado un intervalo, del **1** al **6** (valores que tiene un dado común), necesitamos utilizar una estructura determinada:

Int ([*Valor superior*] – [*Valor inferior*] + 1) * Rnd + [*Valor inferior*]

Valor inferior: nos indica el valor mínimo que tiene el intervalo.

Valor superior: nos indica el valor máximo del intervalo.

En nuestro ejemplo esto quedaría de la siguiente manera. Recuerda que queremos valores enteros, por eso utilizamos la instrucción **Int(Valor)**, entre el **1** y el **6**.

Int(6-1+1*Rnd)+1 Resolviendo las operaciones la instrucción quedaría de la

siguiente forma **Int(6*Rnd)+1**. Con esto conseguiríamos números aleatorios entre el **1** y el **6**, ambos inclusive.

Pongamos otro ejemplo: imaginemos que ahora queremos obtener valores aleatorios entre el **10** y el **20**. La instrucción quedaría de la siguiente forma. **Int(20-10+1*Rnd)+10** resolviendo las operaciones, la instrucción quedaría así: **Int(11*rnd)+10**.

Añadir valores a una lista

Para añadir elementos a una lista deberemos utilizar la instrucción **AddItem**. Cada vez que se pasa por esta línea se inserta un nuevo elemento ocupando el puesto de **último índice + 1**. Recuerda que el primer elemento ocuparía la posición con **índice 0**. La estructura de esta instrucción es la siguiente:

[*Nombre de la lista*].AddItem [*Cadena a añadir*]

Nombre de la lista: es el nombre que le hemos puesto a la lista donde queremos que se añadan los diferentes elementos.

Cadena a añadir: es el valor, cadena, variable... que queremos añadir a nuestra lista.

Observa que esta instrucción, aunque se trate de una asignación, no utiliza el signo igual.

En nuestra aplicación queremos añadir el valor aleatorio obtenido anteriormente. Así que la línea de código quedará de la siguiente manera: **Dado.AddItem (Int(6*Rnd)+1)**

Observaciones con números aleatorios

5. Inicia una ejecución de prueba.

*6. Indica que quieres realizar **5** tiradas.*

*7. Pulsa sobre el botón: **Tirada**.*

8. Observa con detenimiento la secuencia de números que han aparecido en la lista.

9. Detén la ejecución del programa.

10. Vuelve ha ejecutar el programa.

*11. Indica que quieres realizar nuevamente **5** tiradas.*

*12. Pulsa sobre el botón: **Tirada**.*

13. Observa la secuencia de tiradas de la lista.

Si recuerdas la primera secuencia que ha aparecido en nuestra primera ejecución y la comparas con la actual, podrás ver que es exactamente igual. Esto es debido a que, mientras no indiquemos lo contrario, la secuencia de números aleatorios obtenidos con **Rnd** siempre será la misma. Como podrás ver esto no nos interesa en la gran mayoría de casos, con lo que utilizaremos una nueva instrucción que nos permitirá obtener valores completamente aleatorios.

14. Detén la ejecución del programa.

Inicio de valores aleatorios

Vamos a ver una manera para que cada vez que se inicia el programa los valores que se consiguen con la instrucción **Rnd** sean diferentes.

*15. Pulsa **doble clic** sobre el fondo del **formulario**.*

Te aparecerá la ventana de código con un nuevo evento **Form_Load()**. Este evento se ejecuta justo en el momento en el que se carga el formulario. En este caso, como solo disponemos de un formulario, este evento se ejecutará al poner en funcionamiento la aplicación.

*16. Escribe dentro de dicho evento **Randomize**.*

Esta instrucción nos sirve para iniciar con valores, cada vez diferentes, la secuencia de números aleatorios. De tal forma que cada vez que ejecutemos nuestra aplicación obtendremos secuencias aleatorias diferentes.

*17. Vuelva a realizar los pasos del **5** al **13**.*

Pero esta vez observa como la primera y la segunda secuencia son diferentes.

*18. Sin detener la ejecución del programa, vuelve a pedir que se realicen **5** tiradas más.*

Observa como en la lista se han añadido **5** valores más a los que ya teníamos, de esta forma ahora tenemos **10** valores (**5** tiradas anteriores y **5** actuales). Si nosotros seguimos realizando tiradas, los valores de las nuevas tiradas se van añadiendo a la lista de forma indefinida. Observa que cuando la cantidad de valores superan el tamaño de la lista aparece una barra de desplazamiento vertical que nos permite poder visualizar los valores que hemos conseguido en tiradas anteriores.

Si deseas ver los valores de la lista, en lugar de en filas en columnas deberías acceder a la propiedad **Columns** de la lista y cambiar el número de columnas que deseas ver. Si modificas este valor, en el momento que tengamos más elementos de los que caben en la lista aparecerá una barra de desplazamiento horizontal en lugar de vertical. Prueba esta propiedad.

A nosotros, en esta práctica, lo que nos interesaría es conseguir que cada vez que se realice una nueva tirada se borre el contenido de la tabla y aparezcan las nuevas tiradas.

Borrar una lista

Vamos a ver como podemos borrar la lista cada vez que realizamos una tirada nueva.

19. Detén la ejecución del programa.

*20. Pulsa doble clic en el botón: **Tirada**.*

21. Completa el código que ya tienes, para que quede de la siguiente forma:

```
Dado.Clear
For Contador = 1 To NumTiradas.Text
   Dado.AddItem (Int(6*Rnd)+1)
Next Contador
```

La instrucción **Clear** sirve para borrar el contenido de la tabla. La sintaxis de esta instrucción es la siguiente: **[*Nombre lista*].Clear**.

De esta forma cada vez que queramos realizar una nueva tirada, primero se

borrará el contenido de la lista y después se añadirán los elementos nuevos. Cada vez que se borran los elementos de la lista, el **Índice** de la lista vuelve a tener como valor **0**.

Filtrar la entrada de valores

En este apartado vamos ha hacer que el usuario solo pueda poner números en el número de tiradas que quiere realizar, y no pueda introducir ningún tipo de carácter más. Esto es una medida de depuración del programa, ya que de esta forma evitamos que la aplicación aborte al producirse un error.

Veamos que ocurre si introducimos una letra en el número de tiradas deseadas.

. Practica 2

1. Ejecuta la aplicación.

2. Escribe una letra en la casilla para indicar el número de tiradas que deseas realizar.

*3. Pulsa el botón: **Tirada**.*

Observa como te aparece una ventana indicando que se ha producido un **error**. El mensaje de error es: **No coinciden los tipos**. Esto quiere decir que **Visual Basic** no ha podido utilizar lo que nosotros hemos escrito en el interior del número de tiradas como contador para nuestro bucle. **Visual Basic** necesita un número y no una letra.

*4. Pulsa el botón **Terminar**, que aparece en la pantalla de **error**.*

De esta forma podemos volver a la edición del código.

Esta ventana de error es la que tendremos que evitar en muchos casos, para que el usuario no se encuentre con la aplicación colgada.

*5. Haz doble clic sobre el **TextBox**.*

Observa como el evento que se ha abierto ha sido **Change**. El código que escribimos dentro de este evento se ejecutará en el momento en el que se produce un cambio dentro del **TextBox**.

*6. Abre la lista desplegable de los eventos de este objeto y selecciona **KeyPress**.*

Fíjate como ha aparecido un nuevo procedimiento: **Private Sub NumTiradas_KeyPress(KeyAscii As Integer)**. La parte que se encuentra dentro de los paréntesis, devuelve al procedimiento un valor **KeyAscii** siendo este un valor numérico que representa la tecla que se ha pulsado. Esta tabla tiene 256 elementos numerados del 0 al 255, y cada uno de ellos representa un carácter diferente.

*7. Inserta estas líneas de código dentro de **KeyPress**.*

```
If (KeyAscii < 48 Or KeyAscii > 57) Then
   If (KeyAscii <> 8) Then KeyAscii = 0
End If
```

Con estas líneas de código, conseguiremos que el usuario en el momento de pulsar alguna tecla que no sea un número no se escriba dentro del **TextBox**.

8. Realiza una ejecución de prueba e intentar escribir alguna letra.

Observa como no se escribe nada en el interior de este objeto.

*9. Escribe cualquier valor numérico y pulsa **Tirada**.*

10. Detén la ejecución del programa.

Do... Loop

Ahora vamos a ver un tipo de estructura de repetición que depende de una condición. Las instrucciones que hay dentro del bucle se repiten **mientras se cumpla la condición**, mientras la condición sea **Verdadera**.

Tenemos dos tipos de estructuras **Do...Loop**, una en la que se mira la condición antes de realizar ninguna instrucción y otra que se mira después de realizar, al menos, una vez las instrucciones que tenemos dentro del bucle.

Vamos a ver las dos estructuras y después pasaremos a comentar sus diferencias:

Do While [*Condición*] [*Instrucciones*] Loop Do [*Instrucciones*] Loop While [*Condición*]

Condición: lugar reservado para colocar la pregunta que queremos realizar para ver si es **verdadero** o **falso**.

Instrucciones: líneas de código que se ejecutan mientras que la condición sea **verdadera**.

En la primera estructura de repetición lo primero que se mira es la **condición**, si esta se cumple pasamos a realizar las instrucciones que tenemos en el interior del bucle, si no se cumple nos saltamos todas las instrucciones hasta llegar al **Loop** que nos indica el final de dicho bucle.

La segunda estructura de repetición es diferente, primero entramos en el bucle y realizamos todas las instrucciones una vez, después miramos la condición, si esta se cumple volvemos a realizar las instrucciones que tenemos dentro del bucle, por lo contrario si esta no se cumple salimos del bucle.

Es difícil explicar en que momentos se necesitará una u otra instrucción ya que esto dependerá de cada caso y nada mejor que aprenderlo sobre la marcha.

Existen dos estructuras a las que hemos visto antes pero con la diferencia que el bucle se repetirá **mientras no se cumpla la condición**, mientras la condición sea **Falsa**.

La estructura sería la siguiente:
Do Until [*Condición*] [*Instrucciones*] Loop Do [*Instrucciones*] Loop Until [*Condición*]

Observa la diferencia de estas dos estructuras con las vistas anteriormente.

La única diferencia es que en las primeras utilizamos la palabra, **While** y en estas últimas **Until**, por lo demás todo el "funcionamiento" es exactamente igual.

Bucles anidados

En este apartado vamos a ver como podemos anidar, poner dentro de otro, diferentes estructuras de repetición.

Para esto vamos a realizar una práctica en la que intentaremos ordenar una tabla de elementos que inicialmente están desordenados. Para ordenar una tabla existen multitud de métodos diferentes. Algunos de ellos muy simples y poco eficaces, otros son complejos y con un alto grado de eficacia. La dificultad del sistema de ordenación la escogeremos según la cantidad de elementos que deseamos ordenar.

En nuestro caso realizaremos una aplicación que nos ordenará una pequeña tabla que contiene datos aleatorios. Utilizaremos el método de ordenación más sencillo y menos eficaz. Este método, llamado *Método de la burbuja*, es ideal para tablas con pocos datos.

Método de la burbuja

El método de la burbuja se basa en el intercambio de elementos de dos en dos. Si nosotros queremos ordenar la tabla **ascendentemente**, el intercambio de los elementos se produce cuando el primero de ellos es mayor que el segundo. Repitiendo este proceso por lo largo de la tabla conseguimos que el elemento más grande pase a estar en el último lugar de la tabla. El elemento sube por la tabla hasta que ocupa la posición más alta. De ahí viene el nombre de ordenación de la burbuja, el elemento sube como si se tratase de una burbuja dentro de un recipiente.

Los pasos que se siguen exactamente en esta ordenación son los siguientes:

1. Se compara el primer elemento con el segundo de la tabla. Si están desordenados (el primero es más grande que el segundo, en el caso de la ordenación **ascendente**) se intercambian. Luego comparamos el segundo con el tercero, si es necesario los intercambiamos. Continuamos con los intercambios hasta que comparamos el penúltimo con el último.

2. Como segundo paso, volvemos a repetir el primero pero esta vez hasta llegar a comparar el antepenúltimo con el penúltimo, ya que el último elemento ya está ordenado gracias al primer paso.

3. Volvemos a repetir exactamente lo mismo que en el paso uno, pero esta vez con un elemento menos, ya que los dos últimos ya están ordenados.

Este método termina en el momento en el que hemos hecho tantas pasadas como **elementos menos 1** hay en la lista. Realizamos una pasada menos de la cantidad de elementos que hay en la tabla, ya que si todos los elementos de la tabla se han ido ordenando según hemos pasado, como es lógico este último elemento a ordenar ya estará ordenado.

. Práctica 3

Vamos a realizar esta aplicación. Como en todas las prácticas sigue los pasos que te indicamos.

Insertar los elementos

1. Abre un proyecto nuevo.

*2. Inserta un **CommandButton** al que le pondrás como **(Nombre)**: **Nueva**. Cambia su **Caption** y escribe **Nueva**.*

Este botón servirá para borrar la tabla que tengamos en pantalla y crear otra.

Para poder visualizar las tablas que vallamos creando utilizaremos un **ListBox**.

*3. Inserta un **ListBox**. Cambia su tamaño hasta llegar aproximadamente a **855 x 3180**.*

*4. Cambia el **(Nombre)** de este **ListBox** por **Lista**.*

No introduzcas nada en su interior.

*5. Inserta otro **CommandButton**. A este llámale **Ordenar** y ponle como **Caption**: **Ordenar**.*

Al pulsar este botón realizaremos la ordenación de la tabla y la visualizaremos en nuestro **ListBox**.

Recuerda que puedes cambiar todas las propiedades que desees de los objetos insertados en este formulario.

Creación de la tabla

Vamos ha definir la tabla en la que guardaremos todos los valores.

Vamos a pensar como crear esta aplicación para que sea fácil de modificar en el momento en el que deseemos cambiar el número de elementos que componen la tabla. Para ello vamos a crear una **constante** que utilizaremos a lo largo del programa. En el momento que deseemos utilizar una tabla con más o menos elementos y que el programa funcione exactamente igual, solo deberemos cambiar el valor de esta **constante**.

*6. Accede al apartado **General - Declaraciones** de nuestra página de código y escribe lo siguiente:*

```
Const Elementos = 12
```

Con esto crearemos una constante llamada **Elementos** que podremos consultar a lo largo de todo nuestro programa.

Vamos a crear en el mismo apartado una tabla que tenga el número de elementos que marca la constante anteriormente creada. Además esta tabla, para facilitar la comprensión de nuestro código, pondremos como primer elemento el número **1** y como último **Elemento**.

7. Escribe la siguiente línea de código a continuación de la que ya teníamos:

```
Dim Tabla(1 To Elementos) As Integer
```

Observa la declaración del tamaño de la tabla, desde el elemento número 1 al elemento **Elementos**.

La tabla la hemos definido como **Integer** (números enteros).

Iniciar proyecto

Vamos a escribir el código necesario para que al iniciar el proyecto nos aparezca una tabla de número aleatorios dentro de nuestra tabla.

8. Pulsa un clic en el fondo del escritorio.

Seguidamente te aparecerá la ventana de código. Observa el evento que tenemos abierto. **Private Sub Form_Load()** Esto nos indica que todo lo que escribamos dentro de este evento se realizará en el momento en el que carguemos el formulario.

9. Copia las siguientes instrucciones dentro de dicho evento.

```
Private Sub Form_Load()
   Randomize
   For Contador = 1 To Elementos
      Tabla(Contador) = Int((9 * Rnd) + 1)
      Lista.AddItem Tabla(Contador)
   Next Contador
End Sub
```

Observa que en la primera línea, dentro del evento, hemos escrito la instrucción **Randomize** para iniciar la función de números aleatorios.

En la segunda línea hemos iniciado un bucle que se repetirá hasta que **Contador** llegue hasta el número de elementos que hemos definido en **Elementos**. Observa como la variable **Contador** no la hemos definido anteriormente.

Dentro de este bucle se realizará el relleno de los elementos de nuestra **Tabla** con números aleatorios generados mediante la instrucción **Int((9 * Rnd) + 1)**. Observa que para movernos por la tabla utilizamos como índice, nuestro **Contador**.

A la vez que llenamos la tabla vamos añadiendo a nuestra **Lista** los elementos que se acaban de crear. De esta manera a la vez que los creamos los pasamos a la lista, así no tenemos que volver a realizar otra pasada por la tabla.

10. Cierra la ventana de código.

*11. Haz doble clic en el botón **Nueva**.*

Como ya hemos dicho, en el momento en el que pulsemos este botón borraremos lo que haya en la **Lista** e introduciremos una nueva tabla.

12. Sube por el código de la aplicación hasta llegar al evento que hemos escrito anteriormente.

Copiar y pegar

13. Selecciona el contenido de dicho evento.

Para seleccionar líneas de código, simplemente debes ponerte en el margen izquierdo de la ventana de código a la altura de la primera línea de código. Seguidamente pulsa el botón izquierdo del ratón y mientras lo tienes pulsado muévete hasta la última la ultima línea de código dentro de este procedimiento. Fíjate como ha cambiado el color del fondo del texto.

Con el texto seleccionado:

*14. Selecciona la opción **Copiar** del menú **Edición**.*

*15. Sitúa el cursor en el interior del evento del botón **Nueva**.*

*16. Selecciona la opción **Pegar** del menú **Edición**.*

El código se ha copiado.

Vamos ha realizar unas pequeñas modificaciones.

17. Selecciona la primera línea de código y bórrala.

*18. Escribe lo siguiente: **Lista.Clear***

Recuerda que esta instrucción sirve para borrar el contenido de la **Lista** para así

poder poner insertar otra lista nueva.

19. Realiza una ejecución de prueba para ver el funcionamiento de los dos eventos programados hasta el momento.

20. Detén la ejecución.

Ordenación

Ahora vamos a dedicarnos a lo que es en sí la ordenación de la **Tabla.**

*21. Haz doble clic sobre el botón **Ordenar**.*

22. Escribe dentro de este evento las siguientes instrucciones.

```
1  Private Sub Ordenar_Click()
2     I = 1
3     Do
4        For J = 1 To Elementos - I
5           If Tabla(J) >= Tabla(J + 1) Then
6              Cambio = Tabla(J)
7              Tabla(J) = Tabla(J + 1)
8              Tabla(J + 1) = Cambio
9           End If
10             Next J
11             I = I + 1
12          Loop Until I > (Elementos - 1)
13          Lista.Clear
14          For Contador = 1 To Elementos
15             Lista.AddItem Tabla(Contador)
16          Next Contador
17       End Sub
```

Los números que aparecen en cada línea no debes copiarlos, los utilizaremos para facilitar la explicación del funcionamiento del código.

En la línea **2** iniciamos una variable llamada **I** a **1**. Esta variable nos controlará, dentro de un bucle que crearemos en líneas consecutivas, las veces que tenemos que recorrer la tabla, para que esté completamente ordenada. El número de veces será: **el número de elementos de la tabla menos 1**.

En la línea **3** escribimos la primera línea de nuestra estructura **Do...Loop** que termina en la **12**. Utilizamos una estructura **Loop Until** ya que deseamos que se repita este bucle **mientras no se cumpla la condición**. Recuerda que en este tipo de estructuras la condición está en la última línea del bucle.

En la línea **4** iniciamos otro bucle, en este caso un **For... Next** ya que nos interesa que se repitan unas serie de instrucciones un número de veces determinado. En esta línea definimos una nueva variable llamada **J** con valor **1**. Esta variable es la encargada de controlar el bucle. Este bucle se repetirá hasta que **J** llegue al valor que se le indica después del **To**. En nuestro caso cada vez que se ejecute este bucle se repetirá hasta un número diferente, ya que deberemos llegar hasta **Elementos – I**. Vamos a explicarlo con un ejemplo: imaginemos una tabla con **5 elementos**, el ordenador la primera vez que entre en el bucle principal la variable **I** tendrá como valor **1**. Entonces el segundo bucle deberá repetirse hasta que **J** llegue a **5** menos el valor de **I** que es **1**, por lo tanto **4**. Así nos aseguramos que al comparar el elemento que nos marca **J** con el siguiente, en la línea **5** no nos pasemos del índice de la tabla produciéndose un desbordamiento.

En la línea **5** realizamos la comparación del elemento de la tabla cuyo índice es

J con el siguiente. Si resulta que **Tabla(J)** es más grande **(>)** que **Tabla(J+1)** realizamos el cambio de los valores, haciendo pasar el valor de **Tabla(J+1)** a **Tabla(J)**. Este cambio lo efectuamos de la siguiente manera en las líneas **6**, **7** y **8**.

En la línea **6** acumulamos el valor de **Tabla(J)** en una nueva variable que utilizaremos de puente entre las dos posiciones de la tabla. A esta variable puente la llamaremos **Cambio**.

En la línea **7** pasamos el contenido de **Tabla(J+1)** a **Tabla(J)** con lo que escribimos encima del valor que esta tenía, pero no importa ya que este valor está copiado en la variable **Cambio**.

En la línea **8** completamos el cambio pasando el contenido de la variable **Cambio** a **Tabla(J+1)**, machacando el valor antiguo que ya está copiado en **Tabla(J)**.

En la línea **12** termina nuestro **Do... Loop Until**.

Cuando se ejecuta la línea **13** la tabla ya debe estar completamente ordenada, con lo que podemos pasar a visualizarla en nuestro **ListBox**.

Para que no se mezclen la tabla ordenada con la desordenada, primero es preferible borrar la lista, para ello utilizamos la instrucción **Lista.Clear**.

Para pasar el contenido de la tabla a la lista utilizamos un nuevo bucle (línea **14** y **16**), utilizando la variable **Contador** que vuelve a tomar como primer valor **1** y como último el número de elementos de la tabla definido por la constante definida en un principio.

Al pasar por la línea **15** el contenido de la tabla en la posición que nos indica el **contador** pasa a añadirse a la lista. De esta forma volvemos a ver el contenido de la lista que en este caso ya está completamente ordenada.

23. Realiza una ejecución de prueba.

24. Observa los valores de la lista.

*25. Pulsa en el botón **Ordenar**.*

Observa como en un breve espacio de tiempo vuelve a aparecer los mismos valores, pero esta vez completamente ordenados.

*26. Pulsa en el botón **Nueva**.*

27. Vuelve a ordenar la lista.

28. Detén la ejecución del programa.

29. Vuelve al modo diseño.

*30. Accede a la línea en la que se le asigna un valor a la constante **Elementos**.*

*31. Modifica este valor, poniendo **5**.*

32. Realiza otra ejecución de prueba.

Observa como en esta ocasión solo aparecen **5** elementos en la lista. Esto a sido gracias a que en todo nuestro código utilizábamos una constante que controlaba el número de elementos que deseábamos que aparecieran en nuestra tabla. Mira cuantas líneas de código hubiésemos tenido que cambiar si no hubiésemos utilizado una constante.

33. Detén la ejecución.

34. Graba el formulario y el proyecto.

Antes de seguir adelante vamos a aprovechar el ejemplo que acabamos de realizar para explicar un elemento que podemos utilizar en muchas ocasiones el cual nos facilitará un poco la tarea de programar.

Procedimientos

Si observamos el ejemplo que acabamos de realizar podremos observar como hay unas cuantas líneas que se repiten en dos eventos diferentes.

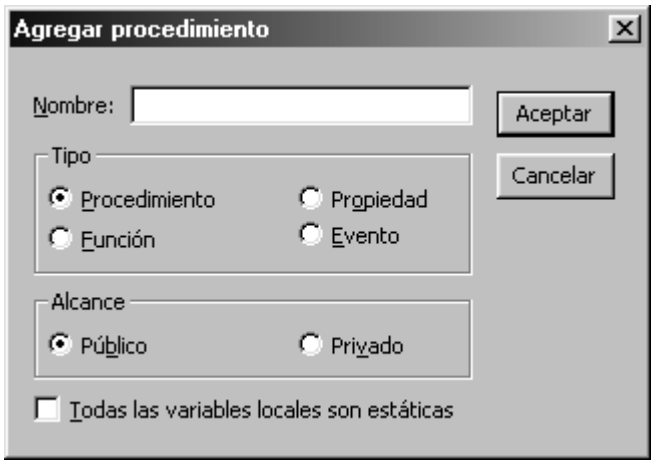
```
For Contador = 1 To Elementos
    Tabla(Contador) = Int((9 * Rnd) + 1)
    Lista.AddItem Tabla(Contador)
Next Contador
```

Estas líneas están dentro de los eventos: **Form_Load()** y **Nueva_Click()**.

En esta ocasión no ocurre no es importante que estas líneas se repitan ya que son sólo 4. Pero podemos tener otras aplicaciones en las que el número de líneas que se repitan puedan ser muchas más, con lo que el número total de líneas de código se vería incrementado haciendo más difícil la localización de un posible error.

Vamos a ver una forma de poder compartir estas líneas de código y utilizarlas en el momento en el que deseemos.

*35. Selecciona la opción: **Agregar procedimiento** dentro de la opción **Herramientas**.*



Seguidamente te aparecerá una ventana como la siguiente:

De esta nueva ventana vamos a explicar las opciones que nos interesa.

Dentro del apartado **Alcance** tenemos dos posibles opciones: **Público** o **Privado**. La primera de ellas se utilizaría en el momento que deseamos crear un procedimiento que se pueda mirar desde cualquier formulario que tuviera una aplicación. Mientras que la segunda opción la utilizaríamos en el momento en el que queremos que el procedimiento sólo pueda ser consultado por el formulario en el que nos encontramos.

Dentro del apartado **Tipo** tenemos 4 opciones de las cuales sólo nos interesan

2: **Procedimiento** y **Función**. Vamos a ver que son cada una de estas opciones.

Procedimiento o Sub:

Un procedimiento ejecuta una tarea específica dentro de un programa sin devolver ningún valor.

Función o Function:

Una función ejecuta una tarea específica dentro de un programa, pero nos devuelve un valor.

En nuestra aplicación lo que nos interesa es un **Procedimiento (Sub)** ya que lo que deseamos es que se realicen una serie de instrucciones, pero no necesitamos que se nos devuelva ningún valor concreto.

*36. Dentro de la ventana **Agregar procedimiento** escribe **Crear** en el apartado **Nombre**.*

*37. Deja seleccionada la opción **Procedimiento** y escoge la opción **Privado** dentro del apartado **Alcance**.*

Observa como dentro del código han aparecido estas líneas de código:

```
Private Sub Crear()

End Sub
```

Podrás ver que son muy parecidas a las líneas de código de los eventos de los diferentes elementos.

Dentro de estas dos nuevas líneas de código vamos a escribir el código que se repite dentro de nuestra aplicación.

```
Private Sub Crear()
   For Contador = 1 To Elementos
      Tabla(Contador) = Int((9 * Rnd) + 1)
      Lista.AddItem Tabla(Contador)
   Next Contador
End Sub
```

Ahora ya estamos preparados para hacer que estas líneas se ejecuten en el momento en el que nosotros deseemos.

Para llamar a este procedimiento simplemente deberemos poner el nombre de este en el punto de la aplicación que deseemos. Por ejemplo vamos a ver como lo haríamos dentro del evento: **Form_Load()**

Anteriormente este evento estaba creado de la siguiente forma:

```
Private Sub Form_Load()
   Randomize
   For Contador = 1 To Elementos
      Tabla(Contador) = Int((9 * Rnd) + 1)
      Lista.AddItem Tabla(Contador)
   Next Contador
End Sub
```

Si miramos las líneas que aparecen dentro del procedimiento que hemos creado

anteriormente podremos ver donde deberemos hacer la llamada a **Crear**.

```
Private Sub Form_Load()
   Randomize
   Crear
End Sub
```

Concatenación de texto

En este apartado vamos a ver una operador que nos permite concatenar elementos. La concatenación es la unión de dos o más elementos que están separados, para formar uno nuevo.

Vamos a ver este operador mediante un ejemplo muy simple.

. Practica 4

1. Crea un nuevo formulario.

*2. Inserta dos **TextBox**. Elimina el contenido y deja los nombres que tienen por defecto.*

*3. Inserta un **Label**. Elimina el contenido y ponle como nombre: **Union**.*

*4. Inserta un **CommandButton**. Pon el texto que quieras. No hace falta que cambies su nombre.*

Con este pequeño ejemplo queremos que al pulsar el botón, aparezca en el **Label** de nuestro formulario la concatenación del contenido de los dos **TextBox**.

*5. Pulsa doble clic en el **botón**.*

6. Escribe las siguientes líneas de código:

```
Private Sub Command1_Click()
   Union.Caption = Text1.Text & Text2.Text
End Sub
```

Recuerda que siempre que se realiza una asignación, pasa el contenido de la derecha del igual a la izquierda de este.

Para unir el contenido de los dos objetos insertados utilizamos el operador **&**.

7. Inicia una ejecución de prueba.

*8. Escribe lo que quieras dentro de los dos **TextBox**.*

*9. Pulsa el **Botón**.*

Observa como el contenido de los dos **TextBox** pasa dentro del **Label**.

*10. Cambia el contenido de los dos **TextBox**.*

*11. Pulsa nuevamente el **Botón** que hemos insertado en el formulario.*

Nuevamente vuelven ha aparecer los dos **TextBox** unidos dentro de nuestro **Label**.

¿Qué tendríamos que hacer para que entre ellos apareciera un espacio en blanco?

La respuesta es muy sencilla, tendríamos que concatenar entre ellos un espacio en blanco.

12. Modifica la línea de código de la siguiente manera:

```
Private Sub Command1_Click()
   Union.Caption = Text1.Text & " " & Text2.Text
End Sub
```

Observa detenidamente la línea donde se produce la concatenación de los diferentes objetos. Primero concatenamos el contenido de **Text1**, después un espacio en blanco **" "** y por último el contenido de **Text2**.

13. Realiza una ejecución de prueba.

14. Detén la ejecución una vez hayas introducido diferentes textos en las correspondientes casillas.

Si nosotros quisiéramos concatenar las casilla de texto separadas entre sí mediante una conjunción **y** deberíamos hacerlo de la siguiente manera:

15. Modifica las líneas de código para que queden así:

```
Private Sub Command1_Click()
   Union.Caption = Text1.Text & " y " & Text2.Text
End Sub
```

Observa que el texto que delante y detrás del texto que deseamos aparezca entre los elementos a concatenar hemos dejado un espacio, esto lo hacemos para que no salga junto a los elementos concatenados y la conjunción **y**. Observa igualmente que este texto está entre comillas.

16. Realiza una ejecución de prueba.

17. Detén la ejecución.

18. Modifica el código de nuestro ejemplo para cambiar el orden de la concatenación, escribiendo un punto posterior, el texto concatenado entre comillas, etc. Realiza todos los cambios que se te ocurran

*19. Sal de **Visual Basic** sin guardar los cambios.*

Recomendamos repasar con profundidad las **estructuras de decisión** (lección anterior) y las **estructuras de repetición** (lección actual).

Fin de la lección 6

[1] Variable en la que podemos almacenar cualquier tipo de dato.

Lección 5

En cualquier tipo de lenguaje de programación usaremos instrucciones y funciones para que la aplicación funcione. En esta lección vamos a conocer algunas de estas estructuras básicas para la programación. Estas estructuras pueden utilizarse solas o conjuntas de forma que formen un programa. Si las instrucciones básicas las tenemos claras desde un principio, tenemos mucho terreno ganado ya que en la gran parte de aplicaciones que programemos utilizaremos estas instrucciones combinadas con más o menos dificultad.

Estructuras básicas en la programación

Vamos a ver que muchos programas necesitan de unas estructuras para su buen funcionamiento. En los siguientes apartados vamos a ir viendo estas estructuras una a una, comentando su utilidad y su escritura. Veremos que una vez conozcamos su funcionamiento es muy fácil pensar en ellas cuando se nos plantea un nuevo problema. Por esta razón pensamos que esta lección es de suma importancia y te recomendamos le dediques todo el tiempo que puedas.

Estructuras de decisión

Las líneas de un programa, dentro de un evento, se ejecutan de arriba abajo. Pero en muchas ocasiones no nos interesa esta linealidad con lo que podemos cambiar el orden de las líneas de código según nos convenga y según la utilidad le queramos dar. Estas estructuras nos permiten tomar decisiones según las condiciones que se den en nuestra aplicación. Por ejemplo, podemos tener una aplicación en la que nos interese calcular el peso ideal de una persona dada, su estatura y su sexo. Pues bien, el sexo será la condición que marcará que camino hay que tomar, ya que si es un hombre tendremos que multiplicar su estatura por una valor y si es una mujer por otra.

Vamos a ir viendo las principales estructuras de decisión una a una.

If...Then...End If

La estructura básica de esta instrucción es la siguiente:

```
If (Condición) Then
   [Instrucciones Verdadero]
Else
   [Instrucciones Falso]
End If
```

Los corchetes muestran partes opcionales de la instrucción.

Condición: Aquí escribiremos la condición a evaluar, para que el ordenador nos devuelva una respuesta y según esta respuesta podamos actuar. Vamos a retomar el ejemplo que hemos planteado anteriormente. Nosotros podríamos tener un **TextBox** llamado sexo, en el que escribiremos una "V" para varón o una "H" para hembra. En el lugar reservado para la **condición** podremos escribir **Sexo.Text = "H"**, quedando la línea completa de la siguiente manera **If Sexo.Text = "H" Then**. Fíjate que aquí no estamos haciendo una asignación, si no que lo que hacemos es plantearnos una pregunta que podemos leer más o menos de la siguiente forma: **Si el contenido del objeto Sexo es H, entonces...** a lo que el ordenador nos devolverá **verdadero** o **falso**. Según la respuesta que nos retorna el ordenador nosotros podremos actuar de una forma u otra.

[Instrucciones Verdadero]: Aquí pondremos la instrucción o instrucciones que se deberá realizar si la respuesta a la condición es **Verdadera**.

Si nosotros solo tenemos una instrucción para colocar en el lugar de **[Instruc-**

ciones Verdadero] y ninguna en **[Instrucciones Falso]** podemos escribir esta estructura de la siguiente manera:

If [Condición] Then [Instrucción Verdadero]

Recuerda que en esta estructura sólo se puede escribir una instrucción en caso que la **condición** sea **verdadera**.

Si nosotros lo que queremos escribir son varias instrucciones, pero no queremos que el ordenador realice nada en el caso de que la condición sea **Falsa** podemos escribir la estructura de la siguiente forma:

```
If [Condición] Then
   [Instrucciones Verdadero]
End If
```

Las instrucciones para realizar en caso que la condición sea **verdadera** se escribirán una debajo de la otra. Estas instrucciones están escritas entre la partícula **If** y **End If**.

. Práctica 1

Vamos a realizar una pequeña aplicación para que el ordenador nos diga, después de introducir la edad de una persona si es mayor de edad o no. Consideraremos la mayoría de edad a los 18 años.

En esta práctica simplificaremos los controles a utilizar, si lo deseas puedes ampliar la práctica tanto como desees.

*1. Coloca, en el lugar que desees de un formulario nuevo, un **TextBox** y un **Label**.*

El **TextBox** lo utilizaremos para introducir la edad y el **Label** para que el ordenador nos devuelva la cadena "**Mayor de edad**" en caso de ser más grande o igual a 18 años, o la cadena "**Menor de edad**" en caso de ser menor de 18 años.

*2. Cambia la propiedad **(Nombre)** del **TextBox** y escribe: **Edad**.*

*3. Cambia la propiedad **(Nombre)** del **Label** y escribe: **Comentario**.*

Puedes borrar el contenido de ambos objetos y modificar el aspecto como tú quieras.

4. Inserta un botón, en el cual colocaremos el código que se ejecutará al hacer clic sobre él.

*5. Pon en la propiedad **Caption** de dicho botón **Calcular**. (No hace falta que cambies la propiedad **(Nombre)**).*

*6. Haz doble clic en el botón **Calcular**.*

7. Escribe el siguiente código:

```
If Val(Edad.Text) < 18 Then
Comentario.Caption = "Es menor de edad"
Else
Comentario.Caption = "Es mayor de edad"
End If
```

Vamos a comentar el código anterior. El ordenador lo primero que hace es mirar

el valor del contenido del objeto llamado **Edad**. Este objeto es de tipo texto y nosotros lo que estamos haciendo es mirar si es mayor o menor que un número. Por esta razón nosotros convertimos el texto en valor numérico utilizando la orden **Val**. El ordenador se hace la pregunta "¿El contenido de **Edad** es menor que 18?. Si la respuesta es **verdadera**, pasa a la primera parte del **If** y escribe en el objeto **Comentario** la frase "Es menor de edad", si la respuesta es **falsa**, pasa a la segunda parte del **If**, donde se escribe "Es mayor de edad".

8. Haz una ejecución de prueba.

. Práctica 2

Vamos a utilizar este mismo ejemplo, para explicar como escribiríamos el código en caso que solo quisiésemos que el ordenador nos devolviese un mensaje en caso que la edad fuera menor de 18 años.

*1. Accede al código del botón **Calcular**.*

2. Borra el código que hemos escrito anteriormente y escribe el siguiente código:

```
If Val(Edad.Text) < 18 Then Comentario.Caption = "Es me-
nor de edad"
```

Todo este código deberá estar escrito en la misma línea.

Observa que no hemos utilizado la parte **Else** de la estructura, ni instrucciones que van en la parte de **falso**, ni **End If** indicando el final del código. Esto es debido a que solo queremos realizar una operación en caso de que sea Verdadero, con lo que no hace falta "cerrar" el **If**, y no ponemos **Else** porque no queremos instrucciones en el caso que sea la respuesta falsa.

3. Realiza un ejecución de prueba.

Ten en cuenta que si cuando ejecutas el programa escribes 21, la aplicación nos devolverá un mensaje diciendo "Es menor de edad". Si acto seguido borras la edad y escribes 10, al pulsar el botón continuarás viendo en el cuadro de mensaje "Es menor de edad". Esto no quiere decir que la aplicación funcione mal, ya que no existe ninguna instrucción que haga que en el caso de no ser mayor de edad se borre el mensaje.

Podemos decir que la aplicación funciona correctamente, pero está mal diseñada, ya que nos proporciona información incorrecta.

. Práctica 3

Imagina que lo que queremos ahora es que el ordenador nos devuelva, mirando la edad que introducimos, si es menor de 10 años, si tiene entre 10 y 20 años, si tiene entre 20 y 30 años o si es mayor de 30. Hasta este momento solo hemos visto la instrucción **If** para poder controlar un caso como el que planteamos.

Veamos que código tendríamos que usar para que funcione el caso anteriormente planteado.

*1. Escribe estás líneas dentro del botón **Calcular**. Borra las instrucciones escritas anteriormente.*

```
1  If Val(Edad.Text) < 10 Then
2        Comentario.Caption = "Menos de 10 años"
3  Else
```

```
4       If Val(Edad.Text) < 20 Then
5           Comentario.Caption = "Entre 10 y 20 años"
6       Else
7           If Val(Edad.Text) < 30 Then
8               Comentario.Caption = "Entre 20 y 30
años"
9           Else
10                  Comentario.Caption = "Más de 30
años"
11              End If
12          End If
13      End If
```

2. Realiza una ejecución de prueba y observa el funcionamiento introduciendo diferentes valores.

Vamos a plantear un caso hipotético para ver como funcionaría este código.

Imagina que el usuario tiene 25 años. El ordenador comenzaría en la línea **1**, donde le preguntamos si el usuario tiene menos de 10 años. Como la respuesta es falsa pasamos a la línea **4**. Aquí preguntamos si el usuario tiene menos de 20 años. En este caso también es falso, con lo que pasamos a la línea **7**. Hacemos otra pregunta mirando si el usuario tiene menos de 30 años. En este caso la respuesta es Verdadera con lo que pasamos a la línea **8** donde escribimos el mensaje correspondiente en el objeto **Comentario** después de esto pasamos a las líneas **11**, **12** y **13** donde termina el código del evento.

En el ejemplo anterior hemos visto como añadir un **If** dentro de otro, observa como cada **If** tiene su **Else** y su **End If** asociados. Observa las líneas **1, 3** y **13**, las líneas **4**, **6** y **12**, y las líneas **7**, **9** y **11**. Habrás podido comprobar que hemos utilizado tabuladores en el momento de escribir el código, esto se hace para poder facilitar la lectura de las líneas de código, ya que de esta manera podemos ver con un golpe de vista, donde empieza y donde acaba un **If**.

Vamos a aprovechar este ejemplo para explicar otra estructura de decisión.

Select Case

Esta estructura la utilizaremos en los casos en los que tengamos muchas condiciones a evaluar, ya que con la estructura **If** se podría complicar bastante.

En esta nueva estructura de decisión se valoran los diferentes valores que puede tomar una determinada expresión y según el valor que tenga esta se actúa en consecuencia.

Es importante destacar que en esta instrucción no miramos si una pregunta o condición es verdadera o falsa, actuando así en consecuencia, sino que miramos el valor que toma un determinado objeto o variable.

```
Select Case [Expresión para comparar]
   Case [Expresión 1]
        [Instrucciones 1]
   ...
   Case [Expresión n]
        [Instrucciones n]
   [Case Else]
        [Instrucciones Else]
End Select
```

En **[Expresión para comparar]** pondremos el objeto sobre el cual queremos

preguntar por su valor.

En **[Expresión 1]** escribiremos cual es la pregunta que deseamos hacer sobre el valor escrito anteriormente.

En **[Instrucciones 1]** pondremos las instrucciones que se realizarán en caso de que **[Expresión 1]** sea verdadera.

Podremos poner tantas **[Expresiones]** como queramos.

Si queremos que se haga algo en caso que ninguna de las expresiones que hemos puesto anteriormente se cumpla, podemos escribir **[Case Else]** y seguidamente la o las instrucciones que se tienen que realizar.

No olvides, igual que en el caso del **If**, cerrar la expresión utilizando **End Select**.

. Práctica 4

*1. Borra el código del botón **Calcular**.*

2. Escribe en su interior el siguiente código:

```
Select Case Val(Edad.Text)
    Case < 10
        Comentario.Caption = "Menos de 10 años"
    Case < 20
        Comentario.Caption = "Entre 10 y 20 años"
    Case < 30
        Comentario.Caption = "Entre 20 y 30 años"
    Case Else
        Comentario.Caption = "Más de 30 años"
End Select
```

3. Haz una ejecución de prueba.

Intenta comparar los dos últimos códigos que te hemos planteado y observa que el segundo es mucho más claro y fácil de entender que el primero en el que utilizábamos **If** anidados.

Operadores de comparación

En los ejemplos anteriores hemos podido ver como estabamos mirando en todo momento si la cantidad introducida era menor (**<**) a unos valores determinados. Como puedes suponer hay varios operadores que nos permitirán hacer diferentes comparaciones. Ha estos elementos le llamaremos **Operadores de comparación**.

Esta es la lista de **operadores de comparación** que podemos utilizar en **Visual Basic**.

=	Igual a
<>	Distinto
<	Menor
>	Mayor
<=	Menor o igual
>=	Mayor o igual

Estos operadores nos permiten hacer comparaciones entre diferentes elementos.

Vamos ha realizar el mismo ejemplo que antes pero utilizando como operador de comparación Mayor (**>**).

```
Select Case Val(Edad.Text)
    Case > 30
        Comentario.Caption = "Más de 30 años"
    Case > 20
        Comentario.Caption = "Entre 20 y 30 años"
    Case > 10
        Comentario.Caption = "Entre 10 y 20 años"
    Case Else
        Comentario.Caption = "Menos de 10 años"
End Select
```

Observa las modificaciones realizadas en el código.

Operadores lógicos

Con los operadores lógicos podemos mirar condiciones que tengan de depender de más de un criterio. Como puede ser: Tener más de 18 años **y** medir 1'70 m.

Estos son los operadores lógicos principales: Y, O y No.

And	Y
Or	O
Not	No

Todos estos operadores lógicos se utilizan para hacer una pregunta entre dos elementos.

Vamos a verlos uno a uno.

And

La conjunción **And** nos permite hacer una pregunta entre dos elementos.

Para que el resultado total sea **verdadero** cada una de las partes de la pregunta tiene que ser **verdadero**.

Pongamos un ejemplo: imagina que tenemos una aplicación en la que debe aparecer un mensaje en caso de que una persona sea mayor de 25 años y esté casada. Para esto tendríamos que escribir la línea de código de la pregunta de la siguiente manera:

```
If Edad.Text > 25 And Casado.Value = True Then
```

(Los objetos de este ejemplo son imaginarios)

Hagamos una tabla en la que podemos ver con más claridad en que casos se mostraría el mensaje.

Edad.Text	**Casado.Value**	**Mensaje**
Verdadero	Verdadero	Sí
Verdadero	Falso	No
Falso	Verdadero	No
Falso	Falso	No

Observa que solo se mostrará el mensaje en el momento en el que las dos

partes de la pregunta sean **verdaderas**. En los demás casos no mostramos el mensaje.

Con la conjunción **And** en el momento en el que aparece un **Falso** en alguna de las dos partes toda la expresión se considera falsa.

Or

La conjunción **Or** nos permite, al igual que **And** hacer una pregunta entre dos elementos. En este caso para que el resultado total sea **verdadero**, solo necesitamos que uno de los elementos sea verdadero.

Pongamos un ejemplo: imaginemos que tenemos una aplicación en la que queremos que se nos muestre un mensaje en el momento que una persona sea mayor de 25 años **o** esté casada. Vamos a escribir la línea de código correspondiente a este ejemplo:

If Edad.Text > 25 Or Casado.Value = True Then

Hagamos una tabla para ver en que casos se mostraría el mensaje .

Edad.Text	**Casado.Value**	**Mensaje**
Verdadero	Verdadero	Sí
Verdadero	Falso	Sí
Falso	Verdadero	Sí
Falso	Falso	No

Con la conjunción **Or** en el momento en el que aparece un **Verdadero**, toda la expresión se considera verdadera.

Not

Solo se utiliza para obtener la negación lógica de un objeto.

Un ejemplo de la utilización de **Not** lo vimos en la lección 2. Cuando al pulsar un botón queríamos poner o quitar la negrita, según si el texto a cambiar estaba o no en negrita.

La instrucción en cuestión era la siguiente: **Texto.FontBold = Not Texto.FontBold**. Si el valor que está detrás del **Not** es **verdadero** pasa a ser **falso** y si es **falso** pasa a ser **verdadero**.

Unión de las estructuras de decisión

En este apartado vamos a ver como podemos utilizar las estructuras de decisión en un mismo programa.

Según el caso que se nos plantee podremos ver que las estructuras de decisión las podemos utilizar una detrás de la otra o anidadas (una dentro de la otra).

. Práctica 5

Vamos a realizar una práctica en la que utilizaremos las dos estructuras de decisión que hemos visto a lo largo de esta lección.

Crearemos una aplicación en la que después de seleccionar un elemento de entre una lista de objetos nos devuelva el precio. El precio podrá aparecer **con** o **sin IVA**.

Para crear esta aplicación nosotros solo indicaremos que objetos se necesitan y

alguna de las propiedades que se deberán cambiar. El aspecto de la aplicación no es lo más importante, pero puedes dedicarle un rato a perfeccionar la apariencia de los objetos insertados para hacer más atractiva la aplicación.

*1. Inserta un **ListBox** al que llamarás **ListaObjetos**.*

*2. Introduce en el **ListBox** 5 nombres de diferentes objetos que puedas comprar en cualquier tienda a los que más tarde pondremos precio.*

*3. Inserta un **Frame**.*

*4. Dentro de este **Frame** inserta dos **OptionButton**.*

*5. El primero tendrá como nombre **ConIVA** y como **Caption: Con IVA**.*

*6. El segundo tendrá como nombre **SinIVA** y como **Caption: Sin IVA**.*

*7. Activa uno de los dos **OptionButton** que has insertado anteriormente.*

Según el **OptionButton** que se seleccione se mostrará el precio **con** o **sin IVA**. Imaginemos que el IVA (Impuesto Valor Añadido) es de un 16% sobre el precio del producto.

*8. Inserta un botón al que le llamaremos **MostrarPrecio** y como **Caption** tendrá **Mostrar Precio**.*

Al pulsar dicho botón la aplicación nos enseñará el precio del producto señalado.

*9. Inserta un **Label** al que le borraremos el contenido y le pondremos como nombre **Precio**.*

En dicho **Label** mostremos el precio del producto seleccionado.

Empezando a codificar

Para saber cual de los objetos de la lista está activado vamos a utilizar la propiedad **ListIndex** del objeto **ListaObjetos**, la cual nos devuelve el índice del elemento seleccionado. Recuerda que dicho índice siempre empieza a contar desde 0.

Vamos a utilizar la estructura de decisión **Select Case** la cual nos permitirá tomar un camino u otro en el código según el índice que esté seleccionado de nuestra lista.

Observa la estructura del código que más adelante escribiremos en el evento **Click** del botón **MostrarPrecio**.

```
Select Case ListaObjetos.ListIndex
    Case 0
` Instrucciones primer objeto seleccionado.
    Case 1
` Instrucciones segundo objeto seleccionado.
    Case 2
` Instrucciones tercer objeto seleccionado.
    Case 3
` Instrucciones cuarto objeto seleccionado.
    Case 4
` Instrucciones quinto objeto seleccionado.
End Select
```

Las líneas que tienen este símbolo al principio ' solo son líneas de comentario. El ordenador en el momento de ejecutar este código las pasaría por alto. Es recomendable utilizar los comentarios para así facilitar la lectura del código.

El ordenado al pulsar el botón, miraría cual es el índice de la tabla que está seleccionado y ejecutaría las líneas de instrucciones que están dentro del índice indicado. En dichas líneas podemos hacer que el ordenador mire si tenemos seleccionado el precio **con** o **sin Iva** y nos lo muestre. También podríamos hacer que el precio de dicho objeto seleccionado lo guarde en una variable y después de la estructura **Select Case** mire si está seleccionado el Iva o no y se realicen los cálculos pertinentes en cada uno de los casos.

Vamos a poner los dos códigos y después miraremos cual de ellos es más correcto o más útil.

Primero escribiremos el código utilizando un **If** dentro de cada uno de los objetos seleccionados. Puedes poner los precios que tú quieras dependiendo de los objetos que hayas escrito dentro de la lista.

10. Completa el código de la siguiente manera.

```
Private Sub MostrarPrecio_Click()
   Select Case ListaObjetos.ListIndex
      Case 0
         If SinIVA.Value = True Then
            Precio.Caption = 1000
         Else
            Precio.Caption = (1000 * 16 / 100) + 1000
         End If
      Case 1
         If SinIVA.Value = True Then
            Precio.Caption = 2000
         Else
            Precio.Caption = (2000 * 16 / 100) + 2000
         End If
      Case 2
         If SinIVA.Value = True Then
            Precio.Caption = 3000
         Else
            Precio.Caption = (3000 * 16 / 100) + 3000
         End If
      Case 3
         If SinIVA.Value = True Then
            Precio.Caption = 4000
         Else
            Precio.Caption = (4000 * 16 / 100) + 4000
         End If
      Case 4
         If SinIVA.Value = True Then
            Precio.Caption = 5000
         Else
            Precio.Caption = (5000 * 16 / 100) + 5000
         End If
   End Select
End Sub
```

Observa que el **IVA** lo calcula el ordenador, multiplicando el precio por un 16% (16 / 100) y sumándoselo al precio de dicho objeto.

*11. Realiza un ejecución de prueba seleccionando diferentes objetos, indicando si deseas ver el precio **con** o **sin IVA**.*

Ahora vamos a poner el mismo código utilizando la estructura **If** una sola vez y utilizando una variable.

```
Private Sub MostrarPrecio_Click()
   Select Case ListaObjetos.ListIndex
      Case 0
            Precios = 1000
      Case 1
            Precios = 2000
      Case 2
            Precios = 3000
      Case 3
            Precios = 4000
      Case 4
            Precios = 5000
   End Select
   If SinIVA.Value = True Then
      Precio.Caption = Precios
   Else
      Precio.Caption = Precios + (Precios * 16 / 100)
   End If
End Sub
```

Si te fijas para el primero de los dos códigos hemos utilizado muchas más líneas que para el segundo. Esto es debido a que en cada uno de los casos producidos por el índice de la lista poníamos una estructura **If**, para comprobar si teníamos seleccionado el precio **con** o **sin IVA**. En el segundo ejemplo solo ponemos la estructura **If** al final del código justo antes de mostrar el precio. En cada uno de los casos almacenamos el precio del objeto seleccionado en una variable para después poder calcular con él el precio total del objeto.

Tanto como en entendimiento, como en cantidad de líneas que ocupa el código es mucho mejor utilizar el segundo caso que no el primero. Siempre tenemos que evitar escribir líneas que puedan estar repetidas o que podamos evitarlas de alguna forma.

Fin lección 5

LECCIÓN 9

En esta lección vamos a ver como podemos trabajar con diferentes formularios dentro de una misma aplicación y como podemos insertarlos en aplicaciones futuras o ya creadas.

Para ver el funcionamiento de los formularios vamos a crear una nueva aplicación con la que podremos jugar al ahorcado.

Crear formulario principal

En esta aplicación no nos vamos a entretener demasiado con la presentación (colores, fuentes, líneas, etc.) ya que nos centraremos mucho más en el trabajo con los formularios y archivos de texto. Tampoco nos centraremos en depurar completamente la aplicación, así que puede ser que en según que momentos no funcione correctamente.

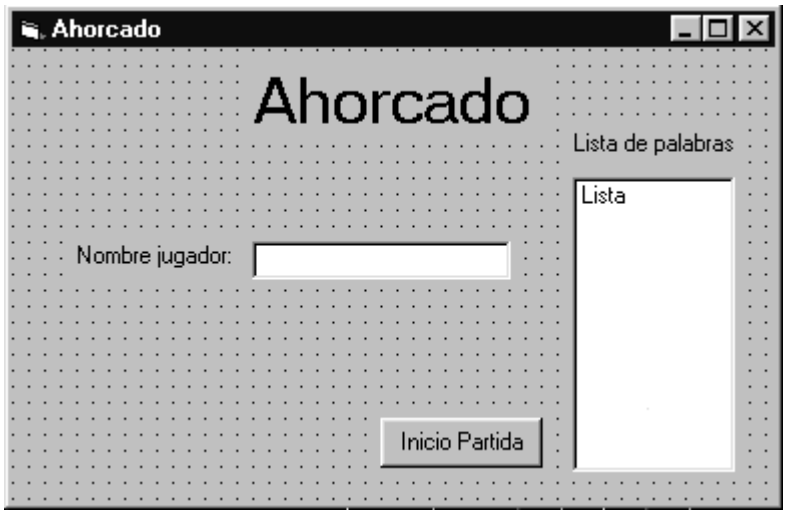
Vamos a crear lo que será el formulario principal de nuestra aplicación.

Práctica 1

1. Abre un nuevo proyecto.

2. Coloca en el formulario actual los objetos que aparecen en la siguiente imagen. (No te indicamos que tipo de objetos hemos insertado)

3. Cámbiales el nombre a los diferentes objetos. (Solo indicamos el nombre de los objetos que intervendrán en el programa, a los demás puedes ponerles el nombre que desees).



Vamos a explicar para que utilizáremos los diferentes elementos de este formulario principal:

Nombre: en este **TextBox** el usuario deberá introducir su nombre. Este nombre lo utilizaremos más adelante para personalizar un poco el juego.

InicioPartida: este **Command Button** será el botón que nos abrirá el siguiente formulario. Formulario donde jugaremos al ahorcado.

Lista: será una lista de las posibles palabras que el ordenador ocultará para que el jugador la adivine.

Más adelante iremos explicando el código que deberemos poner en cada uno de los diferentes objetos.

Antes de insertar el nuevo formulario donde se realizará el juego, vamos a ver el **Explorador de proyectos**, elemento que utilizaremos para movernos por los diferentes formularios de nuestra aplicación.

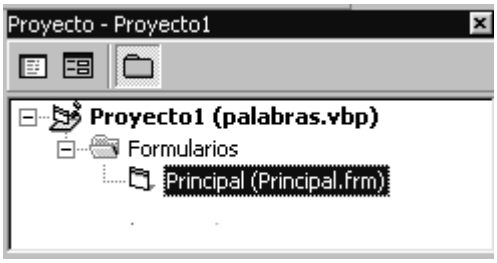
4. Cambia el nombre del formulario por: ***Principal****.*

5. Guarda el formulario con el nombre: ***Principal.frm***

6. Guarda el proyecto con el nombre: ***Palabras.vbp***

Explorador de proyectos

Como ya he dicho anteriormente, el **explorador de proyectos** es un elemento que utilizaremos para movernos entre los diferentes elementos de nuestra aplicación, ya sean, nuevos *formulario*, nuevos *módulos* u otros *elementos* que se puedan añadir.



7. Observa el ***Explorador de proyectos*** *de la aplicación actual.*

Puede ser que no esté visible, vamos a ver las diferentes opciones de las que disponemos para poder visualizar el **Explorador de proyectos**.

8. Abre el menú ***Ver****.*

9. Observa la opción: ***Explorador de proyectos****.*

Ahora que ya conocemos tres maneras diferentes de mostrar el **explorador de proyectos**, utiliza la que desees para que aparezca en pantalla.

En esta ventana nos aparecerán todos aquellos elementos que forman parte de nuestro proyecto actual.

Vamos a explicar las diferentes opciones del **Explorador de proyectos**.

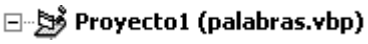
Observa como en esta nueva pantalla aparecen en la parte superior tres nuevos botones.

Ver código: Pulsando sobre este botón veremos el código del elemento que tengamos seleccionado en el **Explorador de proyectos**.

Ver objeto: al hacer un clic sobre este otro botón veremos la parte gráfica del elemento seleccionado.

Alternar carpetas: si pulsamos sobre esta opción mostraremos o ocultaremos las carpetas que aparecen en la ventana de la parte inferior de este **Explorador de proyectos**. Normalmente trabajaremos con esta opción seleccionada ya que así tenemos todos los objetos clasificados por grupos (*carpetas*).

Dentro de la **ventana de lista** (recuadro inferior del explorador) podemos ver los siguientes elementos:



Aquí nos aparecerá el **nombre del proyecto** y entre paréntesis el **nombre del archivo** con el cual se ha guardado. Todo lo que "cuelgue" de este elemento, es lo que forma parte del proyecto. Observa el icono que aparece a la izquierda de esta opción.

Esta carpeta nos indica que dentro de ella todos los elementos que aparezcan serán **formularios** que pertenecen al proyecto.

En esta opción aparecerá el **nombre de los formularios** que forman parte del proyecto y entre paréntesis el **nombre del archivo** que se crea al guardar el formulario. Observa el icono que aparece a la izquierda de esta opción. Más adelante podrás ver como van apareciendo los diferentes formularios que añadiremos al proyecto.

Ahora que ya conocemos las diferentes partes de este **Explorador de proyectos**, vamos a ver los pasos necesarios para añadir un nuevo formulario a nuestro proyecto.

Añadir formulario

Vamos a añadir un formulario nuevo en nuestra actual aplicación.

*10. Despliega el menú **Proyecto**.*

Desde este menú podremos insertar diferentes elementos que podamos ir necesitando a lo largo de nuestra aplicación.

En este caso vamos a añadir un nuevo **formulario**.

*11. Selecciona la opción: **Agregar formulario**.*

Seguidamente te aparecerá una ventana llamada **Agregar formulario** donde podríamos escoger el tipo de formulario que deseamos añadir. La gran mayoría de estos formularios tienen elementos ya programados que nos pueden facilitar el trabajo en según que momentos.

Yo, personalmente, no soy muy partidario en utilizar ventanas "prefabricadas" ya que muchas aplicaciones hechas por diferentes programadores tienen una aparien-

cia idéntica y creo que un programador, como un artista que crea algo, debe tener su propio estilo en el momento de confeccionar cualquier pantalla e incluso en el momento de programar.

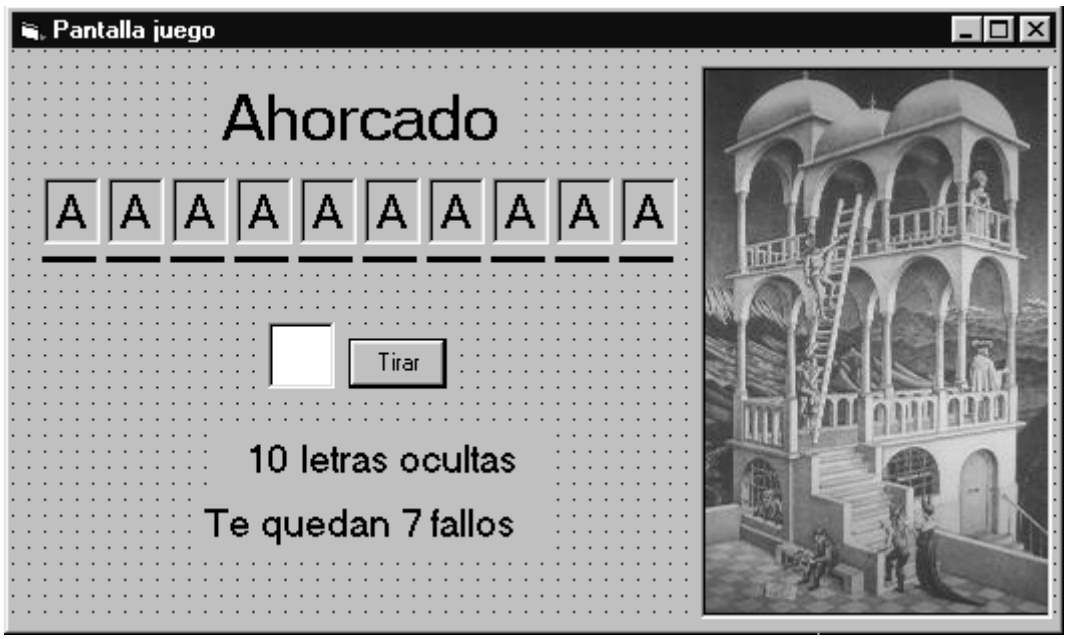
Lo que sí es cierto es que algunas de estas pantallas nos pueden facilitar el trabajo en cierto momento. Más adelante veremos como utilizar una de ellas.

12. De todas las opciones escoge: ***Agregar Formulario****.*

Esta opción nos añadirá un formulario en blanco.

Si observas el **Explorador de proyectos** podrás ver como ha aparecido otro formulario debajo del que ya teníamos anteriormente.

Vamos a crear la pantalla donde llevaremos a cabo la acción del juego.



Recuerda como se creaban las matrices de elementos. Observa que en este formulario hemos eliminado los elementos que tienen como índice el 0. Hemos hecho esto, ya que nos será mucho más fácil movernos por los diferentes elementos contando del 1 al 10 que no del 0 al 9.

La imagen está colocada en un objeto **Image**. Deberás acceder a la propiedad **Picture** y buscar la imagen: **belveder.jpg** puedes utilizar cualquier otra imagen.

Esta imagen en el momento de comenzar la partida estará completamente oculta y ha medida que el usuario comenta errores irá apareciendo por trozos. Para que la "aparición" de esta imagen sea completamente efectiva, deberemos poner el tamaño de la imagen (**Height**) con un valor de: **4140**. Nos interesa que sea así ya que hemos dividido el tamaño de la imagen entre la cantidad de fallos que permitimos que cometa el usuario, haciendo que así cada vez que comete un error nos muestre un trozo.

El objeto llamado **Marcador** sólo hace referencia a un objeto **Label** que tiene como **Caption** un **10**. Mientras que el texto **letras ocultas** es otro **Label** que tiene como **Caption** el texto indicado anteriormente.

*13. Añade los objetos anteriores teniendo cuidado con sus nombres**.***

Al mostrarse esta pantalla no se verá la imagen, ya que tendrá como propiedad **Height** = **0**, no se mostrará ningún objeto de la matriz de elementos **Letra** y se verán tantas líneas como letras tenga la palabra que ha elegido el ordenador.

El objeto llamado **Marcador** nos indicará la cantidad de letras que deberemos adivinar. Este **Marcador** irá disminuyendo cada vez que el jugador adivine una de las letras ocultas.

En el objeto **Errores** irán apareciendo la cantidad de errores de los que disponemos antes de perder la partida.

Cada vez que el usuario introduzca una letra y pulse el botón **Tirar** el ordenador mirará cuantas letras coinciden. Las letras acertadas nos las mostrará en su lugar, si no se acierta la letra se contará como un fallo y la imagen cambiará de tamaño.

*14. Cambia el **(Nombre)** de este nuevo formulario por: **Letras**.*

*15. Guarda el nuevo formulario con el nombre**: LetrasTablero.frm***

*16. Graba nuevamente el proyecto**.***

Orden de los formularios

Aunque no tengamos ningún tipo de código introducido en ninguno de los elementos que forman parte de nuestra aplicación, vamos a realizar una ejecución de prueba.

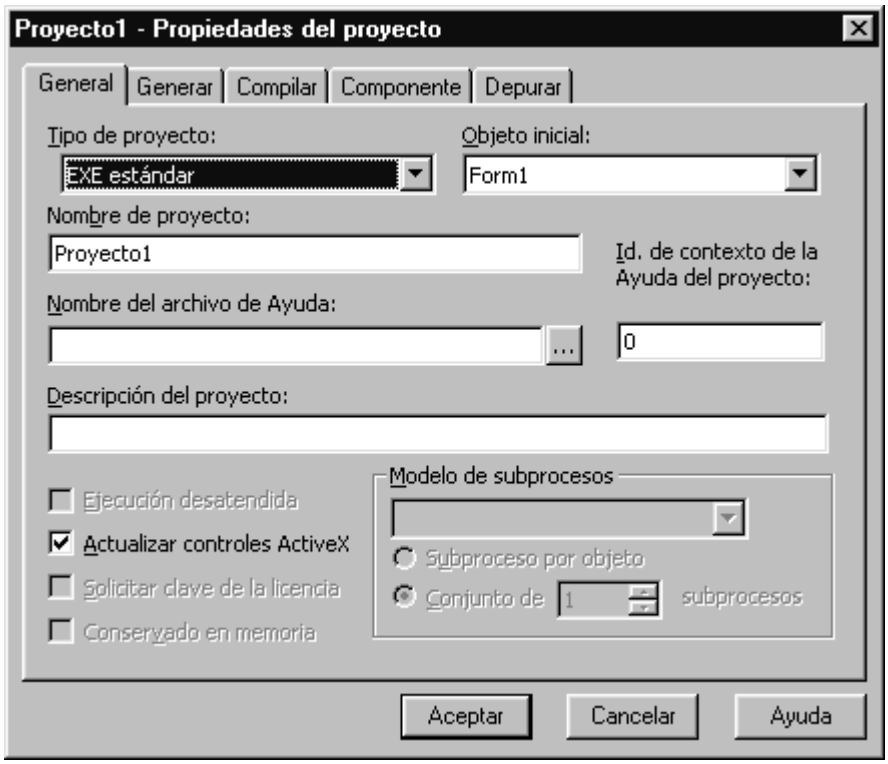
17. Realiza una ejecución de prueba.

Verás que aparece como formulario activo **Principal**.

En nuestro caso ya está bien ya que este será el formulario de presentación del juego, pero ¿qué deberíamos hacer en el caso que quisiéramos que apareciese el formulario **Letras** como primer formulario activo?

*18. Accede a la opción **Propiedades de proyecto...** dentro del menú **Proyecto**.*

Seguidamente aparecerá una pantalla como la siguiente:



De esta pantalla lo único que debería modificar, para que apareciera el otro formulario de la aplicación como principal, sería la opción **Objeto inicial**.

*19. Despliega esta opción y cambia la opción por **Letras**.*

*20. **Acepta** la pantalla.*

*21. Realiza una ejecución de prueba y comprobarás como ahora el primer formulario que nos aparece será el de **Letras**.*

*22. Vuelve a poner el formulario **Principal** como **Objeto inicial**.*

Presentar formularios

Hasta este momento hemos visto como insertar nuevos formularios dentro de nuestra aplicación y como hacer que aparezca un determinado formulario como el principal.

Ahora vamos a ver que es lo que deberemos hacer para abrir un formulario en el momento de ejecutar la aplicación.

Vamos a presentar diferentes acciones que podemos realizar en el momento de presentar los formularios.

Las instrucciones que vamos a ver de aquí en adelante tienen el siguiente formato: **Nombre del formulario.Instrucción**

- Podemos cargar un formulario en la memoria, pero sin mostrarlo en la pantalla. Para hacer esto utilizaremos la instrucción **Load**.

- Si utilizamos la instrucción **Show** cargaremos y presentaremos un determina-

do formulario en la pantalla. Este formulario no será *modal*. Si un formulario no es modal, podremos trabajar con cualquier otro formulario que tengamos en pantalla, mientras que si indicamos que este formulario sea modal, sólo podremos trabajar con el formulario que mostramos.

- Para cargar y presentar un formulario con estilo modal sólo tendremos que añadir la instrucción **vbModal** después de **Show**.

- Si deseas mostrar un formulario que está en memoria deberemos poner la propiedad **Visible** del formulario a **True**. También podemos mostrar el formulario utilizando la instrucción **Show**.

- Si lo que deseamos es que desaparezca el formulario de la pantalla, pero no de la memoria utilizaremos la propiedad **Visible** poniendo su valor a **False**.

- Podemos utilizar la instrucción **Hide** la cual sólo nos cambia la propiedad **Visible** a **False**.

- En cambio si lo que nosotros deseamos es ocultar y descargar el formulario de la memoria utilizaremos la instrucción **Unload**.

En muchos casos deberemos pensar si nos interesa más ocultar el formulario y mantenerlo en la memoria u ocultar el formulario quitándolo de la memoria.

El primero de los casos lo utilizaremos en aquellas aplicaciones en la que debe aparecer un mismo formulario repetidamente, de esta no tendremos que abrir el formulario nuevamente, sólo deberemos visualizarlo.

En cambio el segundo caso lo utilizaremos en el momento que los formularios no los utilicemos muy frecuentemente, con lo que liberaremos los formularios de la memoria quedando así libre para poderla utilizar con otros elementos.

En nuestra práctica haremos que aparezca el formulario con el que jugaremos al pulsar el botón **InicioPartida** del formulario **Principal**.

*23. Haz doble clic sobre este elemento**.***

*24. Escribe el siguiente código: **Letras.Show vbModal***

Con esto haremos que se muestre el formulario **Letras** en estilo **Modal**. Así no permitiremos que el usuario pueda acceder al formulario **principal** hasta que cierre la ventana actual.

25. Realiza una ejecución de prueba.

26. Comprueba como al pulsar dicho botón aparece el siguiente formulario.

27. Intenta cambiar de formulario activo.

El programa no te dejará cambiar de formulario ya que el actual es modal.

Vamos a modificar un poco el código para que al mostrar el nuevo formulario desaparezca el anterior.

*28. Accede nuevamente al código del botón **InicioPartida** y modifica el código para que quede como este:*

```
Principal.Visible = False
Letras.Show
```

En la primera de las líneas lo que hacemos es ocultar el formulario actual (**Principal**) y seguidamente mostrar el formulario de juego (**Letras**) observa que ahora este formulario no hace falta que sea modal, ya que el usuario no podrá cambiar entre diferentes formularios de la misma aplicación.

29. Inicia una ejecución de prueba.

*30. Pulsa sobre el botón **InicioPartida**.*

Podrás observar como el formulario **Principal** desaparece y aparece el formulario **Letras**. Ahora el usuario podría realizar una partida, ¿pero que ocurriría si por cualquier motivo el usuario de la aplicación desea cerrar esta pantalla?.

31. Cierra el formulario actual.

Observa como ha desaparecido el formulario actual y no aparece el principal. Con lo que la única solución que nos queda para finalizar la aplicación es detener la ejecución del programa.

32. Detén la ejecución del programa.

Cuando trabajábamos con los dos formularios en pantalla no ocurría nada, ya que al cerrar uno, todavía teníamos abierto el anterior. Pero, en este caso esto no es así, ya que al cerrar el segundo formulario abierto no podemos trabajar con el primero ya que está oculto.

Vamos a ver un nuevo evento que se realiza en el momento de cerrar el formulario activo.

Cerrar formularios

Cuando cerramos el formulario se iniciará un evento llamado **Unload**. Podríamos decir que este evento es todo lo contrario del evento **Load**.

Vamos a ver como podemos utilizarlo en nuestro ejemplo.

Necesitamos que al cerrar el formulario **Letras**, ya sea utilizando el botón de cerrar o finalizando la partida, se vuelva a ver en pantalla el formulario principal de nuestra aplicación.

*33. Accede al formulario **Letras**.*

*34. Haz doble clic en el fondo del formulario actual, para acceder a los eventos del formulario **Letras**.*

*35. Despliega la lista de los eventos y selecciona de toda la lista: **Unload**.*

Este será el evento que se ejecutará en el momento que se cierre el formulario utilizando el botón cerrar.

En este evento escribiremos la instrucción correspondiente para que se vuelva a visualizar el formulario **Principal**.

36. Escribe las siguientes líneas de código.

```
Private Sub Form_Unload(Cancel As Integer)
   Principal.Visible = True
End Sub
```

37. Inicia una ejecución de prueba.

*38. Accede al segundo formulario utilizando el botón **Inicio partida**.*

Observa que desaparece el formulario **Principal** y aparece el formulario **Letras**.

*39. Cierra el formulario **Letras** mediante el botón **Cerrar**.*

Ahora podremos ver como desaparece el formulario en el que nos encontramos y vuelve a aparecer el formulario **principal**.

Si nosotros modificamos algún tipo de elemento antes de ocultar el formulario, en el momento de volverlo a visualizar, los elementos estarán exactamente igual que antes.

Ahora ya tenemos nuestros dos formularios en nuestra aplicación. Vamos a agregar un nuevo formulario donde pondremos información de la aplicación y de quien la ha realizado.

Añadir formulario prediseñado

*40. Accede a la opción **Agregar formulario** del menú **Proyecto**.*

*41. De la ventana que nos aparece a continuación escoge: **Cuadro de diálogo Acerca de**.*

*42. Pulsa el botón **Abrir**.*

El nuevo formulario es el típico que aparece en la gran parte de las aplicaciones utilizadas por Windows, donde se ofrece información del programador o programadores que han llevado a cabo la aplicación, la versión de dicha aplicación e incluso información del sistema.

En este nuevo formulario podrás modificar, la imagen, el **Label** descripción de la aplicación y el **Label** advertencia...

Los demás elementos se modificarán según el nombre de la aplicación y la versión de esta.

El botón **Aceptar** nos servirá para cerrar la ventana actual y volver a la anterior. Con el botón **Información** nos aparecerá una nueva ventana donde podremos ver información del sistema.

Lo único que nos falta es crear un botón o alguna opción con la que podamos acceder al nuevo formulario.

43. Crea al nuevo objeto.

Dentro de este nuevo objeto deberás escribir esta instrucción:

```
frmAbout.Show Modal
```

Utilizamos **Modal** para que el usuario se vea obligado a cerrar el formulario antes de finalizar con la aplicación.

44. Inicia una nueva ejecución de prueba y accede al nuevo formulario creado.

*45. Pulsa sobre **Información...***

Observa como después de unos segundos aparecerá una nueva ventana donde podremos ver las características del ordenador con el que estamos trabajando.

46. Cierra esta ventana y termina la ejecución de la aplicación.

Ahora ya hemos creado todos los formularios necesarios para esta nueva aplicación. Esta aplicación la volveremos a utilizar en siguientes lecciones con lo que es conveniente que te asegures de tener guardados los tres formularios utilizados en esta aplicación.

*47. Guarda el proyecto con el nombre **Ahorcado**.*

Ahora vamos a ver como podemos añadir un formulario en alguna de las aplicaciones que hemos creado en lecciones anteriores.

Insertar formularios en proyectos

Vamos a crear un formulario como el que hemos visto anteriormente donde podremos ofrecer información de la aplicación y de nosotros mismos, como programadores, esta puede ser una ventana que nos servirá como firma para todas nuestras aplicaciones futuras.

Podemos utilizar un formulario como el que hemos visto anteriormente, pero puede ser que muchos programadores utilicen la misma ventana, con lo que todos los programas pueden dar la sensación que están creados por la misma persona.

Vamos a nombrar los elementos que suelen formar parte de un formulario de este estilo.

Varios **Label** para poner el título de la aplicación, la fecha de creación, la empresa o el programador y observaciones o notas importantes sobre la aplicación.

Sería interesante que apareciera alguna imagen, logotipo de la empresa o cualquier otra para hacer más atrayente el formulario.

En este tipo de formularios aparece un botón para cerrarlo. Deberás insertar dentro de este botón el código **Unload Me** que nos servirá para quitar de la memoria el formulario actual sin preocuparnos del nombre del formulario. La partícula **Me** hace referencia al formulario activo.

En este botón puedes poner la propiedad **Cancel** a **True** para que en el momento que el usuario pulse la tecla **[Esc]** se genere el código de dicho botón.

En el título de este nuevo formulario se suele poner el texto: **Acerca de** seguido del nombre de la aplicación y se suele quitar el icono de la ventana.

Una vez indicado estos objetos típicos, realiza un formulario como quieras que te sirva como "firma" para las aplicaciones que crees a partir de este momento y quieras que sepan quien la ha realizado.

*48. Guarda el formulario actual con el nombre que desees, por ejemplo: **AcercaDe***

Recuerda que será un formulario que podrás utilizar en todas tus aplicaciones, por lo tanto pon un nombre que te sea fácil de recordar.

No hace falta que guardes el proyecto, ya que lo único que nos interesa es el formulario.

Vamos a ver los pasos necesarios para utilizar este nuevo formulario dentro de alguna aplicación que tengamos ya creada.

49. Abre alguna de las aplicaciones que ya tengas creadas.

*50. Accede a la opción **Agregar archivo** del menú **Proyecto**.*

Acto seguido te aparecerá una ventana para abrir uno de los archivos que ya tenemos guardados.

51. Selecciona el formulario que hemos creado anteriormente.

*52. Pulsa en **Aceptar**.*

Observa como después de unos segundos nuestro formulario ha pasado a estar dentro del **Explorador de proyectos**.

Lo único que nos queda hacer es introducir el código, necesario para que el usuario pueda visualizar el formulario introducido desde alguna de las opciones de nuestra aplicación.

La utilización de formulario es de gran ayuda, pero como en todo no podemos abusar ya que el hecho de que aparezcan y desaparezcan ventanas durante la ejecución de una aplicación le puede dar la sensación, a un usuario poco experimentado, de mareo y de no saber lo que se está haciendo exactamente.

Las ventanas debemos utilizarlas en su justa medida y para los casos en los que consideremos que pueden aclarar la utilización de la aplicación.

En siguientes lecciones terminaremos de trabajar con la aplicación que hemos creado en esta lección y veremos como podemos pasar datos de un a otro formulario.

Fin Lección 9