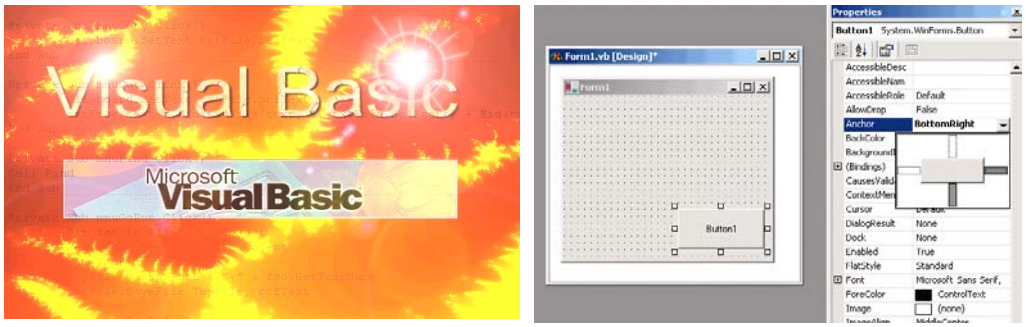
CURSO

Curso Completo de Visual Basic 6.0



Escuela Superior de Ingenieros Industriales
UNIVERSIDAD DE NAVARRA
Javier García de Jalón · José Ignacio Rodríguez
Alfonso Brazález · Patxi Funes · Eduardo Carrasco · Jesús Calleja

1.3 EL ENTORNO DE PROGRAMACIÓN VISUAL BASIC 6.0

Cuando se arranca ***Visual Basic 6.0*** aparece en la pantalla una configuración similar a la mostrada en la Figura 1.1. En ella se pueden distinguir los siguientes elementos:

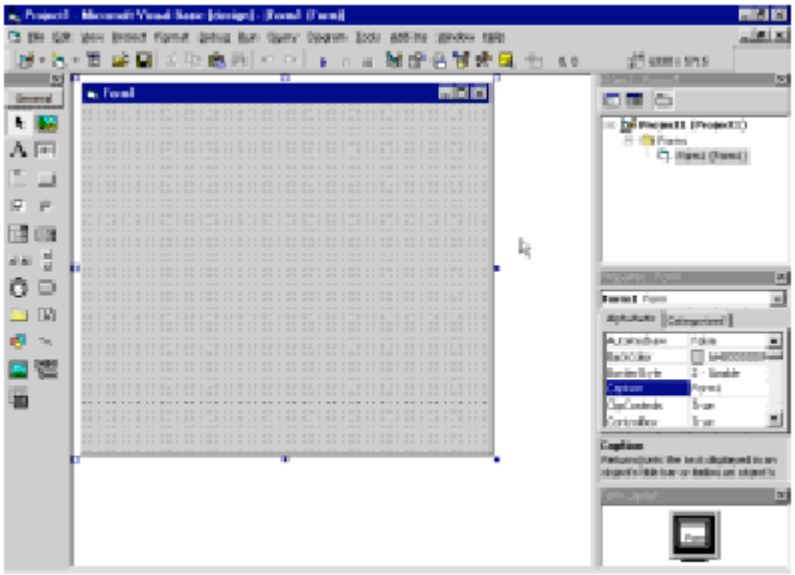


Figura 1.1. Entorno de programación de ***Visual Basic 6.0***.

1. La *barra de títulos*, la *barra de menús* y la *barra de herramientas* de ***Visual Basic 6.0*** en modo ***Diseño*** (parte superior de la pantalla).

2. *Caja de herramientas* (***toolbox***) con los controles disponibles (a la izquierda de la ventana).

3. *Formulario* (***form***) en gris, en que se pueden ir situando los controles (en el centro). Está dotado de una rejilla (***grid***) para facilitar la alineación de los controles.

4. Ventana de *proyecto*, que muestra los formularios y otros módulos de programas que forman parte de la aplicación (arriba a la derecha).

5. Ventana de *Propiedades*, en la que se pueden ver las propiedades del objeto seleccionado o del propio formulario (en el centro a la derecha). Si esta ventana no aparece, se puede hacer visible con la tecla F4.

6. Ventana *FormLayout*, que permite determinar la forma en que se abrirá la aplicación cuando comience a ejecutarse (abajo a la derecha).

Existen otras ventanas para edición de código (***Code Editor***) y para var variables en tiempo de ejecución con el *depurador* o ***Debugger*** (ventanas ***Inmediate***, ***Locals*** y ***Watch***). Todo este conjunto de herramientas y de ventanas es lo que se llama un *entorno integrado de desarrollo* o IDE (***Integrated Development Environment***).

Construir aplicaciones con ***Visual Basic 6.0*** es muy sencillo: basta crear los controles en el formulario con ayuda de la ***toolbox*** y del ratón, establecer sus ***propiedades*** con ayuda de la ventana de propiedades y programar el ***código*** que realice las acciones adecuadas en respuesta a los ***eventos*** o acciones que realice el usuario. A continuación, tras explicar brevemente cómo se utiliza el ***Help*** de ***Visual Basic***, se presentan algunos ejemplos ilustrativos.

1.4 El Help de Visual Basic 6.0

El ***Help*** de ***Visual Basic 6.0*** es de los mejores que existen. Además de que se puede buscar cualquier tipo de información con la función ***Index***, basta seleccionar una propiedad cualquiera en la ventana de propiedades o un control cualquiera en el formulario (o el propio formulario), para que pulsando la tecla <F1> aparezca una ventana de ayuda muy completa. De cada control de muestran las propiedades, métodos y eventos que soporta, así como ejemplos de aplicación. También se muestra información similar o relacionada.

Existe además un breve pero interesante curso introductorio sobre ***Visual Basic 6.0*** que se activa con la opción ***Help/Contents***, seleccionando luego MSDN ***Contents/Visual Basic Documentation/Visual Basic Start Page/Getting Started***.

1.5 EJEMPLOS

El entorno de programación de ***Visual Basic 6.0*** ofrece muchas posibilidades de adaptación a los gustos, deseos y preferencias del usuario. Los usuarios expertos tienen siempre una forma propia de hacer las cosas, pero para los usuarios noveles conviene ofrecer unas ciertas orientaciones al respecto. Por eso, antes de realizar los ejemplos que siguen se recomienda modificar la configuración de ***Visual Basic 6.0*** de la siguiente forma:

1. En el menú ***Tools*** elegir el comando ***Options***; se abre un cuadro de diálogo con 6 solapas.

2. En la solapa ***Environment*** elegir "***Promp to Save Changes***" en "***When a Program Starts***" para que pregunte antes de cada ejecución si se desean guardar los cambios realizados. En la solada ***Editor*** elegir también "***Require Variable Declaration***" en "***Code Settings***" para evitar errores al teclear los nombres de las variables.

3. En la solapa ***Editor***, en ***Code Settings***, dar a "***Tab Width***" un valor de 4 y elegir la opción "***Auto Indent***" (para que ayude a mantener el código legible y ordenado). En ***Windows Settings*** elegir "***Default to Full Module View***" (para ver todo el código de un formulario en una misma ventana) y "***Procedure Separator***" (para que separe cada función de las demás mediante una línea horizontal).

1.5.1 Ejemplo 1.1: Sencillo programa de colores y posiciones

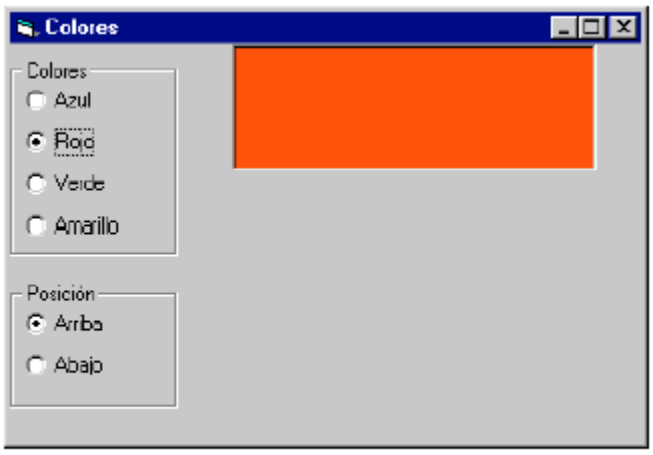


Figura 1.2. Formulario y controles del Ejemplo 1.1.

En la Figura 1.2 se muestra el formulario y los controles de un ejemplo muy sencillo que permite mover una caja de texto por la pantalla, permitiendo a su vez representarla con cuatro colores diferentes.
En la Tabla 1.2 se describen los controles utilizados, así como algunas de sus propiedades más importantes (sobre todo las que se separan de los valores por defecto). Los ficheros de este proyecto se llamarán ***Colores0.vbp*** y ***Colores0.frm***.

Control	Propiedad	Valor	Control	Propiedad	Valor
frmColores0	Name	frmColores0	optVerde	Name	optVerde
	Caption	Colores		Caption	Verde
fraColores	Name	fraColor	fraPosicion	Name	fraPosicion
	Caption	Colores		Caption	Posición
optAzul	Name	optAzul	optArriba	Name	optArriba
	Caption	Azul		Caption	Arriba
optRojo	Name	optRojo	optAbajo	Name	optAbajo
	Caption	Rojo		Caption	Abajo
optAmarillo	Name	optAmarillo	txtCaja	Name	txtCaja
	Caption	Amarillo		Text	""

Tabla 1.2. Objetos y propiedades del ejemplo ***Colores0***.

```
Option Explicit
Private Sub Form_Load()
txtCaja.Top = 0
End Sub

Private Sub optArriba_Click()
txtCaja.Top = 0
End Sub

Private Sub optAbajo_Click()
'El valor de la propiedad siguiente puede tener que ser diferente
txtCaja.Top = 2300
End Sub

Private Sub optAzul_Click()
txtCaja.BackColor = &HFF0000
End Sub

Private Sub optRojo_Click()
txtCaja.BackColor = &HFF&
End Sub

Private Sub optVerde_Click()
txtCaja.BackColor = &HFF00&
End Sub

Private Sub optAmarillo_Click()
txtCaja.BackColor = &HFFFF&
End Sub
```

Sobre este primer programa en ***Visual Basic 6.0*** se pueden hacer algunos comentarios:

1. El comando ***Option Explicit*** sirve para obligar a ***declarar*** todas las variables que se utilicen.
   Esto impide el cometer errores en los nombres de las variables (confundir *masa* con *mesa*, por ejemplo). En este ejemplo esto no tiene ninguna importancia, pero es conveniente acostumbrarse a incluir esta opción. ***Declarar una variable*** es crearla con un nombre y de un tipo determinado antes de utilizarla.

2. Cada una de las partes de código que empieza con un ***Private Sub*** y termina con un ***End Sub*** es un ***procedimiento***, esto es, una parte de código independiente y reutilizable.

El nombre de uno de estos procedimientos, por ejemplo ***optAzul_Click()***, es típico de ***Visual Basic***. La primera parte es el nombre de un objeto (control); después va un separador que es el carácter de subrayado (_); a continuación el nombre de un evento -*click*, en este caso-, y finalmente unos paréntesis entre los que irían los argumentos, en caso de que los hubiera.

3. Es también interesante ver cómo se accede desde programa a la propiedad ***backColor*** de la caja de texto que se llama ***txtCaja***: se hace utilizando el punto en la forma ***txtCaja.BackColor***.
   Los colores se pueden introducir con notación hexadecinmal (comenzando con &H, Seguidos por dos dígitos entre 00 y FF (es decir, entre 0 y 255 en base 10) para los tres Colores fundamentales, es decir para el ***Red***, ***Green*** y ***Blue*** (RGB), de derecha a izquierda. También se pueden utilizar las constantes simbólicas predefinidas en ***Visual Basic 6.0***: ***vbRed***, ***vbGreen*** y ***vbBlue***.

4 Recuérdese que si se desea que el código de todos los eventos aparezca en una misma Ventana hay que activar la opción ***Default to Full Module View*** en la solapa ***Editor*** del comando ***Tools/Options***. Tampbién puede hacerse directamente en la ventana de código con uno de los botones que aparecen en la parte inferior izquierda.

5 ***Es muy importante*** crear primero el control ***frame*** y después colocar los ***botones de Opción*** en su interior. No sirve hacerlo a la inversa. ***Visual Basic*** supone que todos los botones de opción que están dentro del mismo ***frame*** forman parte del mismo grupo y sólo permite que uno esté seleccionado.

1.5.2 Ejemplo 1.2: Minicalculadora elemental

En este ejemplo se muestra una calculadora elemental que permite hacer las cuatro operaciones aritméticas (Figura 1.3). Los ficheros de este proyecto se pueden llamar ***minicalc.vbp*** y ***minicalc.frm***.

El usuario introduce los datos y clica sobre el botón correspondiente a la operación que desea realizar, apareciendo inmediatamente el resultado en la caja de texto de la derecha.

Figura 1.3. Minicalculadora elemental.

La Tabla 1.3 muestra los objetos y las propiedades más importantes de este ejemplo.

Control	Propiedad	Valor	Control	Propiedad	Valor
frmMinicalc	Name	frmMinicalc	lblEqual	Name	lblEqual
	Caption	Minicalculadora		Caption	=
txtOper1	Name	txtOper1	cmdSuma	Name	cmdSuma
	Text			Caption	+
txtOper2	Name	txtOper2	cmdResta	Name	cmdResta
	Text			Caption	-
txtResult	Name	txtResult	cmdMulti	Name	cmdProd
	Text			Caption	*
lblOp	Name	lblOp	cmdDivi	Name	cmdDiv
	Caption			Caption	/

Tabla 1.3. Objetos y propiedades del ejemplo Minicalculadora.

A continuación se muestra el código correspondiente a los procedimientos que gestionan los eventos de este ejemplo.

```
Option Explicit

Private Sub cmdDiv_Click()
   txtResult.Text = Val(txtOper1.Text) / Val(txtOper2.Text)
   lblOp.Caption = "/"
End Sub

Private Sub cmdProd_Click()
   txtResult.Text = Val(txtOper1.Text) * Val(txtOper2.Text)
   lblOp.Caption = "*"
End Sub

Private Sub cmdResta_Click()
   txtResult.Text = Val(txtOper1.Text) - Val(txtOper2.Text)
   lblOp.Caption = "-"
End Sub

Private Sub cmdSuma_Click()
   txtResult.Text = Val(txtOper1.Text) + Val(txtOper2.Text)
   lblOp.Caption = "+"
End Sub
```

En este ejemplo se ha utilizado repetidamente la función ***Val()*** de ***Visual Basic***. Esta función convierte una serie de caracteres numéricos (un texto formado por cifras) en el número entero o de punto flotante correspondiente.

Sin la llamada a la función ***Val()*** el ***operador +*** aplicado a cedenas de caracteres las concatena, y como resultado, por ejemplo, “3+4” daría “34”. No es lo mismo los caracteres “1” y “2” formando la *cadena* o *string* “12” que el número 12; la función ***val()*** convierte cadenas de caracteres numéricos –con los que no se pueden realizar operaciones

aritméticas- en los números correspondientes –con los que sí se puede operar matemáticamente-. ***Visual Basic 6.0*** transforma de modo automático números en cadenas de caracteres y viceversa, pero este es un caso en el que dicha transformación no funciona porque el operador "+" tiene sentido tanto con números como con cadenas.

1.5.3 Ejemplo 1.3: Transformación de unidades de temperatura

La Figura 1.4 muestra un programa sencillo que permite ver la equivalencia entre las escalas de temperaturas en grados centígrados y grados Fahrenheit. Los ficheros de este proyecto se pueden llamar ***Temperat.vbp*** y ***Temperat.frm***.

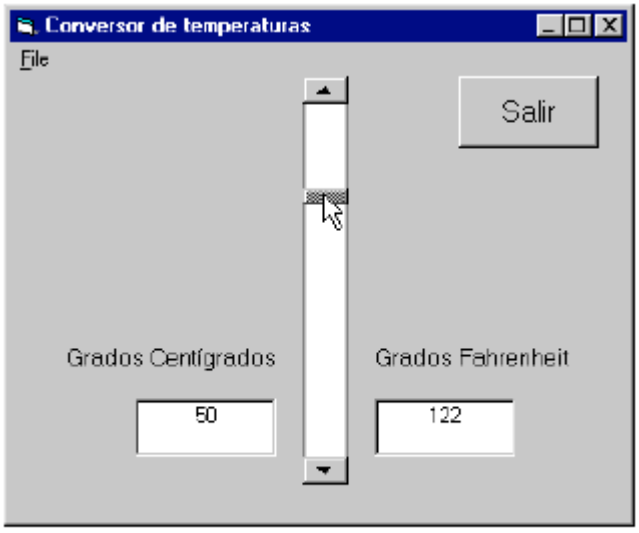


Figura 1.4. Equivalencia de temperaturas.

En el centro del formulario aparece una barra de desplazamiento vertical que permite desplazarse con incrementos pequeños de 1º C y grandes de 10º C. Como es habitual, también puede cambiarse el valor arrastrando con el ratón el cursor de la barra.
Los valores máximos y mínimo de la barra son 100º C y -100º C.

A ambos lados de la barra aparecen dos cuadros de texto (color de fondo blanco) donde aparecen los grados correspondientes a la barra en ambas escalas. Encima aparecen dos rótulos (*labels*) que indican la escala de temperaturas correspondiente. Completan la aplicación un botón ***Salir*** que termina la ejecución y un menú ***File*** con la única opción ***Exit***, que termina asimismo la ejecución del programa.

La Tabla 1.4 indica los controles utilizados en este ejemplo junto con las propiedades y los valores correspondientes.

Control	Propiedad	Valor	Control	Propiedad	Valor
frmTemp	Name	frmTemp	vsbTemp	Name	vsbTemp
	Caption	Conversor de temperaturas		Min	100
mnuFile	Name	mnuFile		Max	-100
	Caption	&File		SmallChange	1
mnuFileExit	Name	mnuFileExit		LargeChange	10
	Caption	E&xit		Value	0
cmdSalir	Name	cmdSalir	lblCent	Name	lblCent
	Caption	Salir		Caption	Grados Centigrados
	Font	MS Sans Serif, Bold, 14		Font	MS Sans Serif, 10
txtCent	Name	txtCent	lblFahr	Name	lblFahr
	text	0		Caption	Grados Fahrenheit
txtFahr	Name	txtFahr		Font	MS Sans Serif, 10
	text	32			

Tabla 1.4. Controles y propiedades del Ejemplo 1.3.

Por otra parte, el código con el que este programa responde a los eventos es el contenido en los siguientes procedimientos:

```
Option Explicit

Private Sub cmbSalir_Click()
      Beep
      End
End Sub

Private Sub mnuFileExit_Click()
      End
      End Sub

Private Sub vsbTemp_Change()
      txtCent.Text = vsbTemp.Value
      txtFahr.Text = 32 + 1.8 * vsbTemp.Value
End Sub
```

Sobre este tercer ejemplo se puede comentar lo siguiente:

1. Se ha utilizado la propiedad ***Value*** de la barra de desplazamiento, la cual da el valor actual de la misma con respecto a los límites inferior y superior, previamente establecidos (-100 y 100).

2. Mediante el procedimiento ***cmdSalir_Click***, se cierra el programa, gracias a la instrucción ***End***. El cometido de ***Beep*** no es otro que el de emitir un pitido a través del altavoz del ordenador, que nos indicará que en efecto se ha salido del programa.

3. La función ***mnuFileExit_Click()*** y activa desde el menú y termina la ejecución sin emitir ningún sonido.

4. Finalmente, la función ***vsbTemp_Change()*** se activa al cambiar el valor de la barra de desplazamiento; su efecto es modificar el valor de la propiedad ***text*** en las cajas de texto

que muestran la temperatura en cada una de las dos escalas.

1.5.4 Ejemplo 1.4: Colores RGB

La Figura 1.5 muestra el formulario y los controles del proyecto ***Colores***. Los ficheros de este proyecto se pueden llamar ***Colores.vbp*** y ***olores.frm***.

Figura 1.5. Colores de fondo y de texto.

En este ejemplo se dispone de tres barras de desplazamiento con las que pueden controlarse las componentes RGB del color del fondo y del color del texto de un control ***label***. Dos botones de opción permiten determinar si los valores de las barras se aplican al fondo o al texto. Cuando se cambia del texto al fondo o viceversa los valores de las barras de desplazamiento (y la posición de los cursores) cambian de modo acorde.

A la derecha. de las barras de desplazamiento tres cajas de texto contienen los valores numéricos de los tres colores (entre 0 y 255). A la izda. Tres ***labels*** indican los colores de las tres barras. La Tabla 1.5 muestra los controles y las propiedades utilizadas en el este ejemplo.

Control	Propiedad	Valor	Control	Propiedad	Valor
frmColores	Name	frmColores	hsbColor	Name	hsbColor
	Caption	Colores		Min	0
lblCuadro	Name	lblCuadro		Max	255
	Caption	TEXTO		SmallChange	1
	Font	MS Sans Serif, Bold, 24		LargeChange	16
cmdSalir	Name	cmdSalir		Index	0,1,2
	Caption	Salir		Value	0
	Font	MS Sans Serif, Bold, 10	txtColor	Name	txtColor
optColor	Name	optColor		Text	0
	Index	0,1		Locked	true
	Caption	Fondo, Texto		Index	0,1,2
	Font	MS Sans Serif, Bold, 10	lblColor	Name	lblColor
				Caption	Rojo,Verde,Azul
				Index	0,1,2
				Font	MS Sans Serif, 10

Tabla 1.5. Objetos y propiedades del ejemplo ***Colores***.

Una característica importante de este ejemplo es que se han utilizado ***vectores (arrays) de controles***. Las tres barras se llaman ***hsbColor*** y se diferencian por la propiedad ***Index***, que varía entre 0, 1 y 2. También las tres cajas de texto, las tres ***labels*** y los dos botones de opción son ***arrays de controles***. Para crear un array de controles basta crear el primero de ellos y luego hacer ***Copy*** y ***Paste*** tantas veces como se desee, respondiendo afirmativamente a la pregunta de si desea crear un array.

El ***procedimiento Sub*** que contiene el código que gestiona un ***evento*** de un array es único para todo el array, y recibe como argumento la propiedad ***Index***. De este modo que se puede saber exactamente en qué control del array se ha producido el evento. Así pues, una ventaja de los *arrays* de controles es que pueden compartir el código de los eventos y permitir un tratamiento conjunto por medio de bucles ***for***. A continuación se muestra el código correspondiente a los procedimientos que tratan los eventos de este ejemplo.

```
Option Explicit

Public Brojo, Bverde, Bazul As Integer
Public Frojo, Fverde, Fazul As Integer

Private Sub cmdSalir_Click()
      End
      End Sub

Private Sub Form_Load()
      Brojo = 0
      Bverde = 0
      Bazul = 0
      Frojo = 255
      Fverde = 255
      Fazul = 255
      lblCuadro.BackColor = RGB(Brojo, Bverde, Bazul)
      blCuadro.ForeColor = RGB(Frojo, Fverde, Fazul)
End Sub
Private Sub hsbColor_Change(Index As Integer)
      If optColor(0).Value = True Then
        lblCuadro.BackColor = RGB(hsbColor(0).Value, hsbColor(1).Value, _
```

```
                              hsbColor(2).Value)
        Dim i As Integer
        For i = 0 To 2
          txtColor(i).Text = hsbColor(i).Value
        Next i
      Else
        lblCuadro.ForeColor = RGB(hsbColor(0).Value, hsbColor(1).Value, _
                              hsbColor(2).Value)
        For i = 0 To 2
          txtColor(i).Text = hsbColor(i).Value
        Next i
      End If
End Sub

Private Sub optColor_Click(Index As Integer)
     If Index = 0 Then 'Se pasa a cambiar el fondo
       Frojo = hsbColor(0).Value
       Fverde = hsbColor(1).Value
       Fazul = hsbColor(2).Value
       hsbColor(0).Value = Brojo
       hsbColor(1).Value = Bverde
       hsbColor(2).Value = Bazul
    Else 'Se pasa a cambiar el texto
       Brojo = hsbColor(0).Value
       Bverde = hsbColor(1).Value
       Bazul = hsbColor(2).Value
       hsbColor(0).Value = Frojo
       hsbColor(1).Value = Fverde
       hsbColor(2).Value = Fazul
    End If
End Sub
```

El código de este ejemplo es un poco más complicado que el de los ejemplos anteriores y requiere unas ciertas explicaciones adicionales adelantando cuestiones que se verán posteriormente:

1. La función ***RGB()*** crea un ***código de color*** a partir de sus argumentos: las componentes RGB (*Red*, *Green* and *Blue*). Estas componentes, cuyo valor se almacena en un byte y puede oscilar entre 0 y 255, se determinan por medio de las tres barras de desplazamiento.

2. El color ***blanco*** se obtiene con los tres colores fundamentales a su máxima intensidad. El color ***negro*** se obtiene con los tres colores RGB a cero. También se pueden introducir con las constantes predefinidas ***vbWhite*** y ***vbBlack***, respectivamente.

3. Es importante disponer de unas ***variables globales*** que almacenen los colores del fondo y del texto, y que permitan tanto guardar los valores anteriores de las barras como cambiar estas a sus nuevos valores cuando se clica en los botones de opción. Las variables globales, definidas en la parte de definiciones generales del código, fuera de cualquier procedimiento, son visibles desde cualquier parte del programa. Las variables definidas

dentro de una función o procedimiento sólo son visibles desde dentro de dicha función o procedimiento (*variables locales*).

4. La función ***hsbColor_Change(Index As Integer)*** se activa cada vez que se cambia el valor en una cualquiera de las barras de desplazamiento. El argumento ***Index***, que ***Visual Basic*** define automáticamente, indica cuál de las barras del array es la que ha cambiado de valor (la 0, la 1 ó la 2). En este ejemplo dicho argumento no se ha utilizado, pero está disponible por si se hubiera querido utilizar en el código.

Continuará.....