VISUAL BASIC 6

NIVEL 01

GUIA 01

Introducción

Visual Basic es hoy el lenguaje de programación mas popular del mundo. Es el sueño del programador de aplicaciones. Es un producto con una interfaz grafica de usuario para crear aplicaciones para Windows basado en el lenguaje Basic y en la programación orientada a objetos

La palabra "Visual" hace referencia al método que se utiliza para crear la interfaz grafica de usuario. En lugar de escribir numerosas líneas de código para implementar una interfaz, se utiliza el ratón para arrastrar y colocar los objetos prefabricados al lugar deseado dentro de un formulario.

La palabra "Basic" hace referencia al BASIC (Beginners All-Purpose Simbolic Instruction Code), un lenguaje utilizado por más programadores que ningún otro lenguaje en la historia de la informática. Visual Basic ha evolucionado a partir del lenguaje BASIC original y ahora contiene centenares de instrucciones, funciones y palabras clave, muchas de las cuales están diferectamente relacionadas con la interfaz grafica de Windows.

Es importante saber tambien, que la inversión realizada en el aprendizaje de Visual Basic le ayudara a abarcar otras áreas, porque este lenguaje de programación no es exclusivo de la aplicación Visual Basic. Este lenguaje es utilizado también por Microsoft Excel, Microsoft Access y muchas otras aplicaciones Windows. El sistema de Visual Basic Sript para programar en Internet, también es subconjunto del lenguaje Visual Basic.

ESTILOS DE PROGRAMACIÓN

Se entiende por estilos de programación los métodos que existen para mejorar la calidad de los programas de computación. Y ¿cuales son las características de un buen programa?. Vamos a exponer algunas respuestas:

1. *El programa debe funcionar*. La característica mas simple e importante de un programa es que funcione

2. *El programa no debe tener dificultades*. Hay que anticiparse a las situaciones particulares en las cuales va a emplearse el programa con el fin de evitar errores. Es responsabilidad del programador asegurar que el programa este libre de errores.

3. *El programa debe estar bien documentado*. La documentación es necesaria para ayudar a comprender o a utilizar un programa. La documentación puede realizarse de dos formas: la documentación externa, que incluye diagramas de flujo, descripciones de los algoritmos, etc. Y la documentación interna, o comentarios en el propio prgrama. La documentación externa esta dirigida a los usuarios del programa, esencialmente. La documentación interna esta dirigida totalmente al programador.

FASES DEL PROCESO DE PROGRAMACIÓN

1. Análisis del problema: Esto es, conocer el problema antes de proceder a desarrollar la solución.

2. Desarrollo de la solución: Hay una desafortunada tendencia por parte de muchos programadores a sucumbir al engañoso atractivo de la maquina, iniciando la fase de construcción e instalación antes de que el problema haya sido resuelto realmente.

3. Construcción de la solución en forma de programa: Este proceso es completamente mecánico, ya que consiste en la construcción en forma de programa real de la solución desarrollada.

4. Prueba: Todo programador experto prueba mentalmente cada instrucción cuando la esta escribiendo, y simula, también mentalmente, la ejecución de cualquier modulo o seccion de su programa antes de proceder a realizar una prueba real de la etapa.

5. Documentación: Es imprescindible para la manipulación del programa, asi como para su mantenimiento.

PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos (POO) es una forma de programación que utiliza objetos, ligados mediante mensajes, para la solución de problemas. Puede considerarse como una extensión natural de la programación estructurada en un intento de potenciar los conceptos de modularidad y reutilización de código.

Mecanismos Básicos de la POO

Los mecanismos básicos de la programación orientada a objetos (POO) son: Objetos, Mensajes, Métodos , Propiedades y Eventos.

Objetos
Un programa tradicional se compone de procedimientos y de datos. Un programa orientado a objetos se compone solamente de objetos. Un objeto es una encapsulacion genérica de datos y de los procedimientos para manipularlos. Dicho de otra forma, un objeto es una entidad que tiene unos atributos particulares, las propiedades, y unas formas de operar sobre ellas, los métodos. Por lo tanto, un objeto contiene, por una parte, operaciones que definen su comportamiento, y por otra, variables manipuladas por esas operaciones que definen su estado.

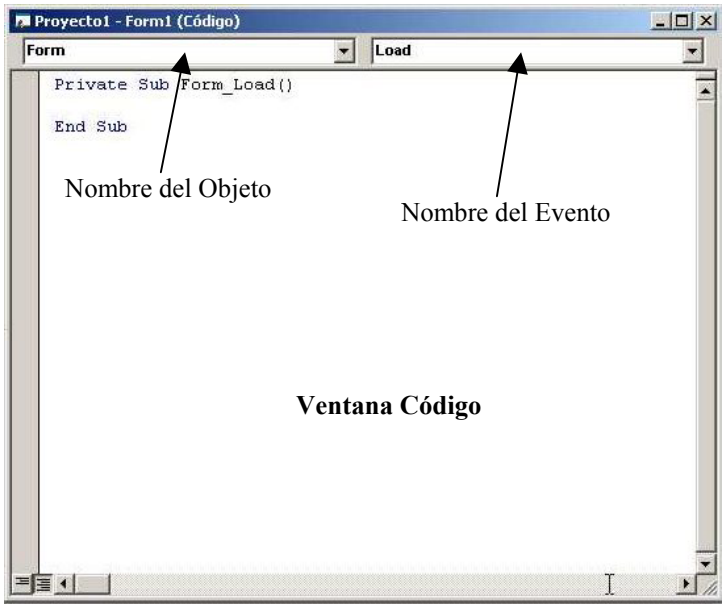
Mensajes
Cuando se ejecuta un programa orientado a objetos, los objetos estan recibiendo, interpretando y respondiendo a mensajes de otros objetos. Esto marca una clara diferencia con respecto a los elementos de datos pasivos de los sitemas tradicionales. Por ejemplo, en Visual Basic un mensaje esta asociado con un procedimiento, de tal forma que cuando un objeto recibe un mensaje la respuesta a ese mensaje es ejecutar el procedimiento asociado. Este procedimiento recibe el nombre de metodo.

Métodos
Un metodo se implementa en una clase de objetos y determina como tiene que actuar el objeto cuando recibe un mensaje. En adicion, las propiedades permitiran almacenar información para dicho objeto. Un metodo puede también enviar mensajes a otros objetos solicitando una accion o información.

Propiedades
Las propiedades de un objeto definen la manera en que dicho objeto se ve y se comporta

Eventos

Visual Basic es un lenguaje de programación controlado por eventos. Esto significa que el código se ejecutara en respuesta a algo que ocurre. Por ejemplo, si hace clic en un botón durante la ejecución del programa, se generara un evento Clic y se ejecutara automáticamente el código que le corresponde.



Procedimientos

La primera línea del código, la cual Visual Basic escribió por usted, comienza con las palabras ***Private Sub***. Sub es una palabra clave que indica que ahí comienza un procedimiento. Un procedimiento es código dedicado a una acción en particular. La ultima línea del código (también escrita por Visual Basic) es ***End Sub***, que indica el final del procedimiento.

VISUAL BASIC 6.0

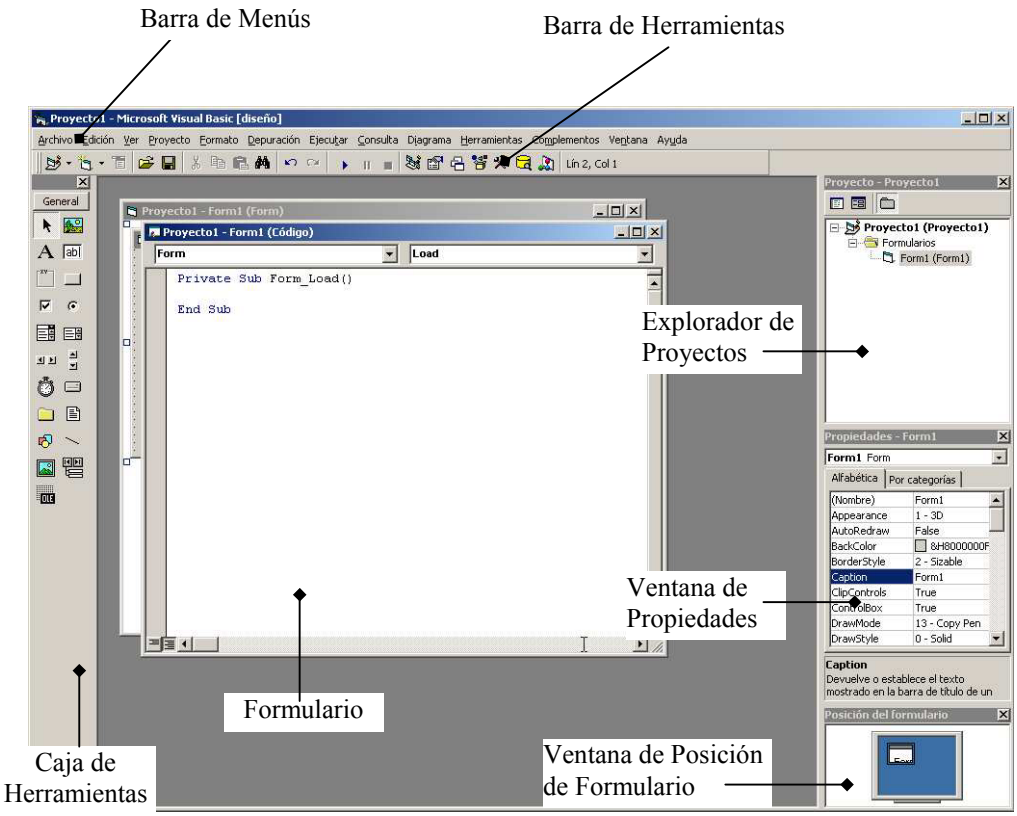
- Los objetos de Visual Basic estan encapsulados, es decir, contienen su propio código y sus propios datos.
- Los objetos de Visual Basic tienen propiedades, métodos y eventos.
- Al conjunto de propiedades y métodos se les llama Interfaz. Ademas de su interfaz predeterminada, los objetos pueden implementar interfaces adicionales para proporcionar polimorfismo.

Requerimientos Mínimos De Instalación

- Microprocesador Pentium 100 Mhz o superior
- Disco duro con un espacio mínimo disponible de 100Mb
- Unidad de CD-ROM
- Un ratón
- 32 Mb de memoria RAM o mas
- Microsoft Windows 95 o posterior

ENTORNO DE DESARROLLO DE VISUAL BASIC

Cuando se arranca Visual Basic 6, usted podrá ver una interfaz similar a la de la figura que se muestra a continuación:



- ✓ Barra de Menús: Visualiza las ordenes que usted utiliza para desarrollar una aplicación.

- ✓ Menus Contextuales: Un menu contextual es un menu flotante que presenta ordenes especificas realtivas a un determinado objeto.

- ✓ Barra de Herramientas: Facilita un acceso rapido a las ordenes mas comúnmente utilizadas. Las barras de herramientas se pueden acoplar debajo de la barra de menus o pueden "flotar" si selecciona el tirador vertical del borde izquierdo y la arrastra fuera del lugar debajo de la barra de menus.

- ✓ Explorador de Proyectos: Esta ventana contiene la lista de los ficheros que componen el proyecto actual.

- ✓ Ventana de Propiedades: Cada objeto lleva asociado un conjunto de propiedades (nombre, posición, tamaño, color, etc). Para ver o especificar los valores de las propiedades de un objeto, utilizaremos la ventana de propiedades .

- ✓ Caja o cuadro de herramientas: Proporciona un conjunto de herramientas que permiten colocar los controles en el formulario durante el diseño de la interfaz grafica del usuario. Observe que hay una ficha, denominada General, predeterminada. Esto significa que una caja de herramientas puede tener varias fichas. Puede crear su propia ficha si ejecuta

la orden la orden agregar ficha del menu contextual a añade los controles a la ficha resultante.

- ✓ Formulario: Es la ventana sobre la que colocaremos los controles de la interfaz grafica que el usuario utilizara para comunicarse con la aplicación. Cada formulario de la aplicación tiene su propia ventana diseñador de formulario.

- ✓ Ventana de posición de formulario: Esta ventana le permite colocar los formularios de su aplicación utilizando una pequeña representación grafica de la pantalla. Si visualiza el menu contextual de esta ventana y ejecuta la orden "Guias de Resolución", observara que se pintan unas lineas que delimitan la pantalla para cuando la resolución sea de 640x480 pixeles.

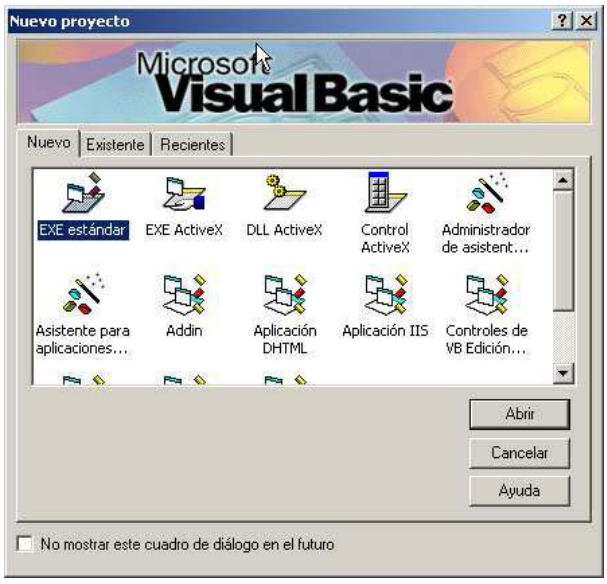
MI PRIMERA APLICACIÓN

Suponiendo que ya hemos arrancado Visual Basic, ¿Cuál es el siguiente paso para desarrollar una aplicación Windows? En general, para construir una aplicación siga los pasos indicados a continuación:

1. Cree una nueva aplicación (nuevo proyecto)
2. Ajuste el tamaño por defecto del formulario
3. Dibuje los controles
4. Defina las propiedades del formulario
5. Escriba el código para cada uno de los objetos
6. Guarde la aplicación
7. Verifique la aplicación
8. Cree un archivo ejecutable

Crear una nueva aplicación

Cuando arranca Visual Basic, aparece una ventana como la siguiente:



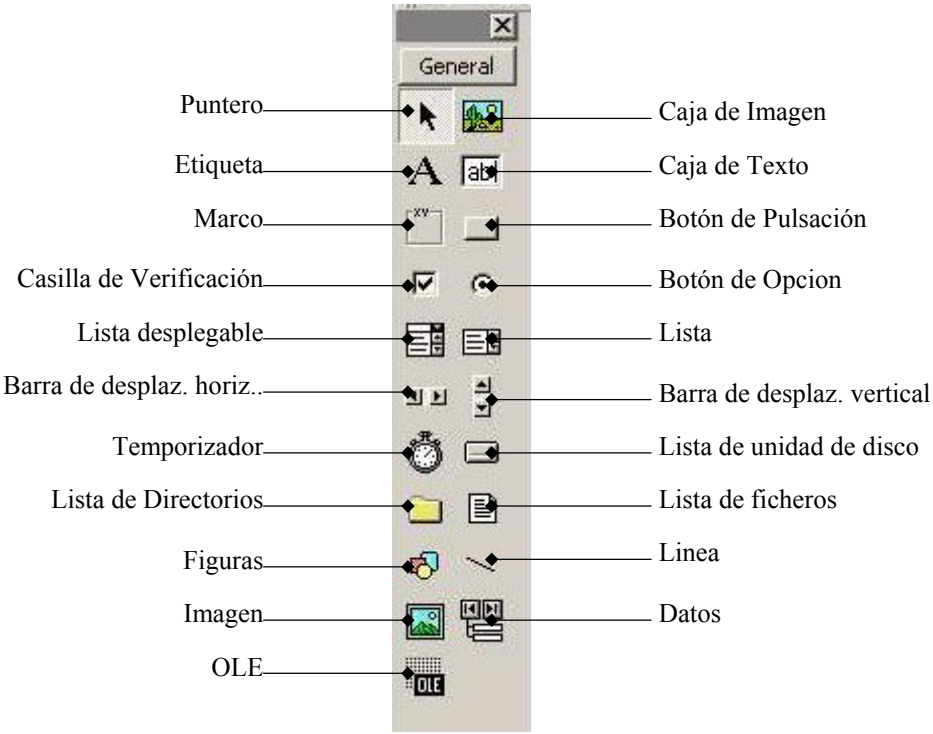
Esta ventana tiene tres pestañas que hacen referencia al proyecto: Nuevo, Existente y reciente que vamos a abrir. La pagina Nuevo tiene un grupo de iconos, que se corresponden con plantillas de proyectos. Elija la planilla seleccionada por omisión, EXE estándar, y pulse el botón Aceptar.

Una aplicación Visual Basic esta formada generalmente por los siguientes tipos de ficheros: módulos de formularios (.frm), módulos estándar (.bas), módulos de clases (.cls), archivos binarios (.bin) y ficheros de recursos (.frx). la lista de archivos que componen la aplicación junto con las opciones (el conjunto de opciones lo puede ver ejecutando la orden Opciones del Menú Herramientas) es guardada en un archivo de proyecto con extensión .vbp cada vez que se guarda la aplicación o proyecto. Así mismo, el espacio de trabajo del proyecto Visual Basic se guarda en un fichero con extensión .vbw

Creación o dibujo de los controles de un formulario

En Visual Basic disponemos fundamentalmente de dos tipos de objetos: ventanas y controles. Un formulario es una ventana sobre la que nosotros dibujamos los elementos que el usuario tiene que utilizar para comunicarse con la aplicación. Los elementos son los controles; esto es, objetos gráficos que permiten entrar o salir datos: por ejemplo, cajas de texto, botones, etiquetas, marcos, listas y temporizadores. El formularios mas los controles forman la interfaz o medio de comunicación.

Para añadir un control a un formulario, utilizaremos la caja de herramientas que se muestra en la figura siguiente. Cada herramienta de la caja crea un único control.



El puntero se utiliza para manipular los controles existentes sobre el formulario.

Una caja de imagen se utiliza normalmente para presentar gráficos, para que actué como contenedor de otros controles y para presentar texto mediante el método print.

Utilizaremos una etiqueta cuando queramos un texto, de una o mas lineas que no pueda ser modificado por el usuario.

Una caja de texto es un área dentro del formulario en la que el usuario puede escribir o visualizar texto.

Un marco se utiliza para realzar el aspecto del formulario.

Un botón de pulsación tiene asociada una orden con el. Esta orden se ejecuta cuando el usuario hace clic sobre el.

Una casilla de verificación se utiliza para seleccionar una opción. De esta forma se pueden seleccionar varias opciones de un grupo

El control botón de opción se utiliza para seleccionar una sola opción de entre varias. De esta forma solo se puede seleccionar una sola opción de un grupo de ellas.

La barra de desplazamiento horizontal y vertical, permiten seleccionar un valor dentro de un rango de valores.

El temporizador permite activar procesos a intervalos regulares

La lista de unidades de disco se utiliza para visualizar la lista de unidades disponibles con el fin de seleccionar una

La lista de directorios se utiliza para visualizar los directorios a los que el usuario puede moverse

El control figuras se utiliza para dibujar rectángulos, cuadrados, elipses o circulos.

El control línea se utiliza para dibujar líneas rectas en un formulario

El control imagen se utiliza para presentar gráficos en los siguientes formatos: bmp, iconos, meta-archivos, JPEG o GIF.

El control de datos le permite conectarse a una base de datos existente y visualizar su información en el formulario

Borrar un control
Para borrar un control, primero se selecciona haciendo clic sobre el, y a continuación se pulsa la tecla Supr (Del).

Bloquear la posición de todos los controles
Una vez que se haya ajustado el tamaño de los objetos y haya situado los controles en su posición definitiva, puede seleccionar el formulario y bloquear sus controles para que no puedan ser movidos accidentalmente. Para ello, ejecute la orden Bloquear controles del Menu Formato.

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 Herramientas elegir el comando Opciones; se abre un cuadro de diálogo con 6 solapas.
2. En la solapa Entorno elegir "Preguntar si se guardan los cambios" en "Cuando se inicie un Programa" para que pregunte antes de cada ejecución si se desean guardar los cambios realizados. En la solada Editor elegir también "Requerir declaración de variables" en "Opciones de Código" para evitar errores al teclear los nombres de las variables.

3. En la solapa Editor, en Opciones de Código, dar a "Ancho de Tabulación" un valor de 4 y elegir la opción "Sangría Automática" (para que ayude a mantener el código legible y ordenado). En Opciones de Ventana elegir "Ver modulo completo de forma predeterminada" (para ver todo el código de un formulario en una misma ventana) y "Separador de Procedimientos" (para que separe cada función de las demás mediante una línea horizontal).

Ejemplo desarrollado 1.1: Sencillo programa de colores y posiciones

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.

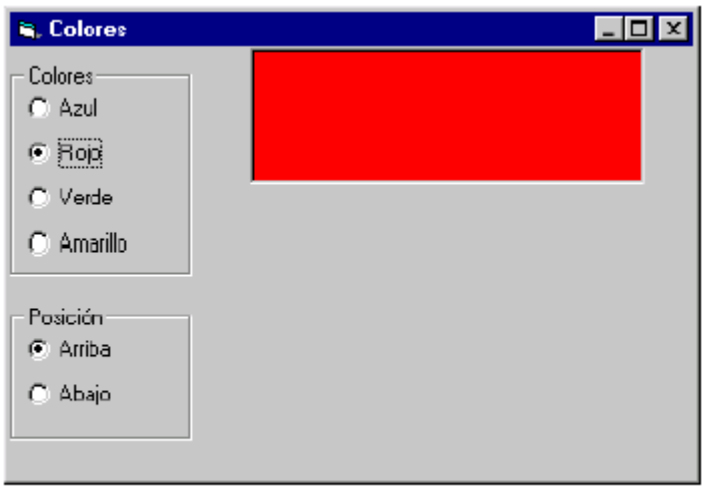


Figura 1.2. Formulario y controles del Ejemplo 1.1.

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***.

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

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

```
txtCaja.Top = frmColores0.ScaleHeight - txtCaja.Height
End Sub
Private Sub optAzul_Click()
txtCaja.BackColor = vbBlue
End Sub
Private Sub optRojo_Click()
txtCaja.BackColor = vbRed
End Sub
Private Sub optVerde_Click()
txtCaja.BackColor = vbGreen
End Sub
Private Sub optAmarillo_Click()
txtCaja.BackColor = vbYellow
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 podrían también introducir con notación hexadecimal (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. Aquí se han utilizado 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 Ver Modulo completo en forma predeterminada en la solapa Editor del Menu Herramientas / Opciones. También 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, estando seleccionado, 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.

Nombre	Código HEX	Color
vbBlack	&H000000	Negro
vbRed	&H0000FF	Rojo.
vbGreen	&H00FF00	Verde.
vbYellow	&H00FFFF	Amarillo.
vbBlue	&HFF0000	Azul.
vbMagenta	&HFF00FF	Magenta.
vbCyan	&HFFFF00	Cyan.
vbWhite	&HFFFFFF	Blanco.

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

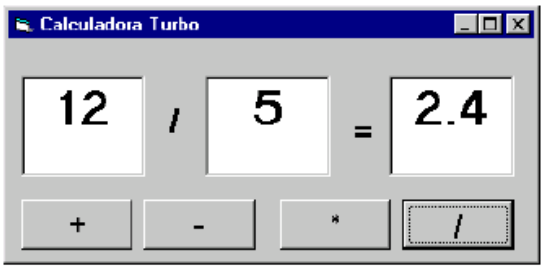


Figura 1.3. Minicalculadora elemental.

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 cadenas 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.

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***. 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.

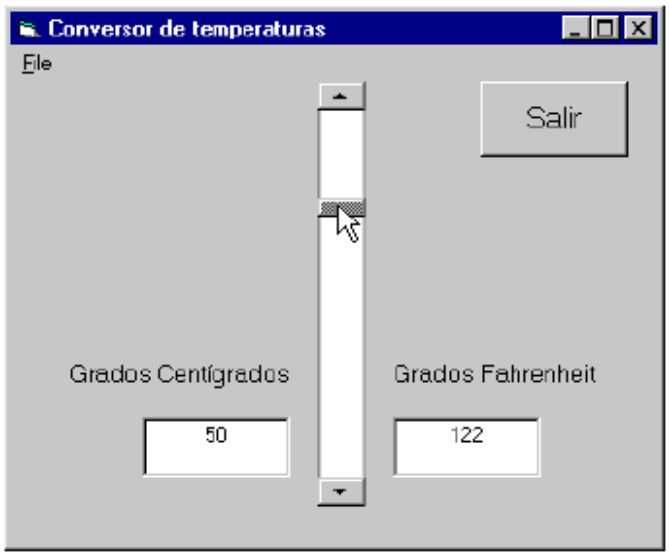


Figura 1.4. Equivalencia de temperaturas.

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 Centígrados
	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 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.

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 Colores.frm. 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.

Figura 1.5. Colores de fondo y de texto.

Control	Propiedad	Valor	Control	Propiedad	Valor
frmColores	Name	frmColores	hsbColor	Name	hsbColor
	Caption	Colores		Min	0
lblCuadro	Name	lblCuadro		Max	255
	Caption	INFORMÁTICA 1		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 toma los valores 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 al cuadro de diálogo que pregunta 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)
lblCuadro.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 éstas 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.