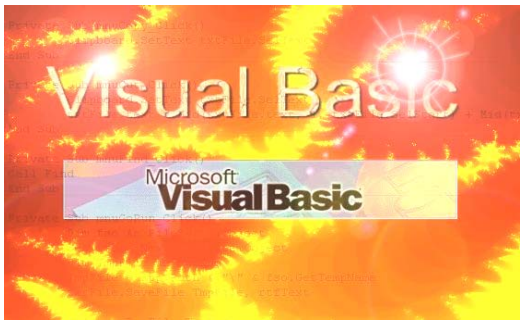
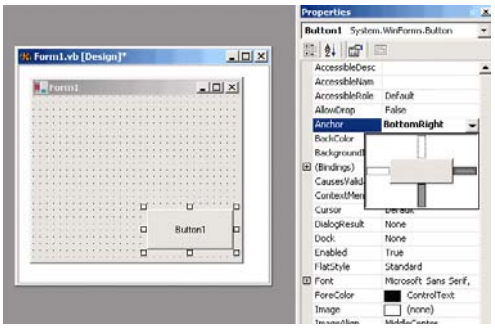
CURSO

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

3.11 FUNCIONES PARA MANEJO DE CADENAS DE CARACTERES

Existen varias funciones útiles para el manejo de *cadenas de caracteres* (***Strings***). Estas funciones se utilizan para la evaluación, manipulación o conversión de cadenas de caracteres. Algunas de ellas se muestran en la Tabla 3.4.

Utilidad	Función en Visual Basic 6.0	Comentarios
Número de caracteres de una cadena	Len(string | varname)	
Conversión a minúsculas o a mayúsculas	LCase(x), UCase(x)	
Conversión de cadenas a números y de números a cadenas	Str(n), CStr(n), Val(string)	
Extracción de un nº de caracteres en un rango, de la parte derecha o izquierda de una cadena	Mid(string, ini[, n]), Right(string, length), Left(string, length)	El parámetro ***n*** de Mid es opcional e indica el número de caracteres a extraer a partir de "***ini***"
Extracción de sub-cadenas	Split(string, [[delim], n])	Devuelve un array con las ***n*** (-1 para todas) subcadenas separadas por ***delim*** (por defecto, el espacio)
Unión de sub-cadenas	Join(string, [delim])	
Comparación de cadenas de caracteres	strComp(str1, str2)	Devuelve -1, 0, 1 según ***str1*** sea menor, igual o mayor que ***str2***
Hallar si una cadena es parte de otra (está contenida como sub-cadena)	InStr([n], str1, str2)	devuelve la posición de ***str2*** en ***str1*** buscando a partir del carácter ***n***
Hallar una cadena en otra a partir del final (reverse order)	InstrRev(str1, str2, [n])	devuelve la posición de ***str2*** en ***str1*** buscando a partir del carácter ***n***
Buscar y reemplazar una subcadena por otra en una cadena	Replace(string, substring, replacewith)	Reemplaza ***substring*** por ***replacewith***

Es necesario tener presente que cuando se quieren comparar dos cadenas de caracteres, dicha comparación se realiza por defecto en función del código ASCII asociado a cada letra (ver Anexo 8.1). Esto significa que por ejemplo *caña* es posterior a *casa* debido a que la letra ñ tiene un código ASCII asociado superior a la letra s (ñ es 164; s es 115). Esto mismo ocurre con las vocales acentuadas. Si se desea conseguir una comparación alfabética lógica es necesario incluir al comienzo del fichero de código la sentencia ***Option Compare Text*** (frente a ***Option Compare Binary*** establecida por defecto). La función ***strComp()*** admite un tercer argumento que permite especificar el tipo de comparación (constantes ***vbBinaryCompare*** o ***vbTextCompare***).

Ejemplos:

```
MyDouble = 437.324              ’ MyDouble es un Double.
MyString = CStr(MyDouble)       ’ MyString contiene "437.324".
MyValue = Val("2457")           ’ Devuelve 2457.
MyValue = Val(" 2 45 7")        ’ Devuelve 2457.
MyValue = Val("24 and 57")      ’ Devuelve 24.
AnyString = "Hello World"       ’ Se define el string.
MyStr = Right(AnyString, 6)     ’ Devuelve " World".
MyStr = Left(AnyString, 7)      ’ Devuelve "Hello W".
MyStr = Right(AnyString, 20)    ’ Devuelve "Hello World".
i = StrComp("casa", "caña")     ´ Devuelve –1 por defecto y 1 con Option
                                  ´Compare Text
MyString = "Mid Function Demo"   ' Se crea un nuevo string.
LastWord = Mid(MyString, 14, 4)  ' Devuelve "Demo".
MidWords = Mid(MyString, 5)      ' Devuelve "Function Demo".
```

El operador ***Like*** permite comparar dos cadenas de caracteres. Si son iguales devuelve ***True*** y si no lo son, ***False***.
Existe además el ***operador de concatenación &*** que puede ser utilizado con cadenas de caracteres. Se utiliza para poner una cadena a continuación de otra.

Por ejemplo:

```
str1 = "My first string" ’Se inicializan los strings
str2 = "My second string"
TextoFinal = str1 & str2 ’TextoFinal vale "My first stringMy second string"
```

El ***operador "+"*** opera de forma análoga, pero su uso se desaconseja pues en ciertas ocasiones convierte las cadenas en números y realiza la suma.
Para obtener más información sobre cada una de las funciones buscar ***Strings*** en el ***Help*** de ***Visual Basic 6.0***.

3.12 FUNCIONES MATEMÁTICAS

Al igual que las funciones vistas para el manejo de cadenas de caracteres, existe una serie de funciones matemáticas las cuales permiten realizar cálculos dentro de un programa de ***Visual Basic***.

Dichas funciones se muestran en la Tabla 3.5:

Función matemática	Función en Visual Basic	Función matemática	Función en Visual Basic
Valor absoluto	Abs(x)	Nº aleatorio	Rnd
Arco tangente	Atn(x)	Seno y coseno	Sin(x), Cos(x)
Exponencial	Exp(x)	Tangente	Tan(x)
Parte entera	Int(x), Fix(x)	Raíz cuadrada	Sqr(x)
Logaritmo	Log(x)	Signo (1, 0, -1)	Sgn(x)
Redondeo	Round(x, ndec)		

Tabla 3.5. Funciones matemáticas en ***Visual Basic 6.0***.

Ejemplos:

```
MyNumber = Abs(50.3)                          ’ Devuelve 50.3.
MyNumber = Abs(-50.3)                         ’ Devuelve 50.3.
MyAngle = 1.3                                 ' El ángulo debe estar en radianes.
MySecant = 1 / Cos(MyAngle)                   ' Calcula la secante.
MySqr = Sqr(4)                                ' Devuelve 2.
MySqr = Sqr(23)                               ' Devuelve 4.79583152331272.
MyVar1 = 12: MyVar2 = -2.4: MyVar3 = 0        'Declaración de las variables
MySign = Sgn(MyVar1)                          ' Devuelve 1.
MySign = Sgn(MyVar2)                          ' Devuelve -1.
MySign = Sgn(MyVar3)                          ' Devuelve 0.
```

Con el fin de completar estas funciones, se ofrece a continuación una relación de funciones que son derivadas de las anteriores. El alumno podría programar dichas funciones en un fichero ****.bas*** y así poderlas utilizar posteriormente en cualquier programa. Dichas funciones se muestran en la Tabla 3.6:

Función matemática	Expresión equivalente
Secante	Sec(X) = 1 / Cos(X)
Cosecante	Cosec(X) = 1 / Sin(X)
Cotangente	Cotan(X) = 1 / Tan(X)
Arcoseno	Arcsin(X) = Atn(X / Sqr(-X * X + 1))
Arcocoseno	Arccos(X) = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
Arcosecante	Arcsec(X) = Atn(X / Sqr(X * X – 1)) + Sgn((X) -1) * (2 * Atn(1))
Arcocosecante	Arccosec(X) = Atn(X / Sqr(X * X - 1)) + (Sgn(X) - 1) * (2 * Atn(1))
Arcocotangente	Arccotan(X) = Atn(X) + 2 * Atn(1)
Seno Hiperbólico	HSin(X) = (Exp(X) - Exp(-X)) / 2
Coseno Hiperbólico	Hcos(X) = (Exp(X) + Exp(-X)) / 2
Tangente Hiperbólica	Htan(X) = (Exp(X) - Exp(-X)) / (Exp(X) + Exp(-X))
Secante Hiperbólica	HSec(X) = 2 / (Exp(X) + Exp(-X))
Cosecante Hiperbólica	Hcosec(X) = 2 / (Exp(X) - Exp(-X))
Cotangente Hiperbólica	Hcotan(X) = (Exp(X) + Exp(-X)) / (Exp(X) - Exp(-X))
Arcoseno Hiperbólico	Harcsin(X) = Log(X + Sqr(X * X + 1))
Arcocoseno Hiperbólico	Harccos(X) = Log(X + Sqr(X * X - 1))
Arcotangente Hiperbólica	Harctan(X) = Log((1 + X) / (1 - X)) / 2
Arcosecante Hiperbólica	Harcsec(X) = Log((Sqr(-X * X + 1) + 1) / X)
Arcocosecante Hiperbólica	Harccosec(X) = Log((Sgn(X) * Sqr(X * X + 1) +1) / X)
Arcocotangente Hiperbólica	Harccotan(X) = Log((X + 1) / (X – 1)) / 2
Logaritmo en base N	LogN(X) = Log(X) / Log(N)

Tabla 3.6. Funciones auxiliares matemáticas (no las tiene ***Visual Basic 6.0***).

4. EVENTOS, PROPIEDADES Y CONTROLES

En este capítulo se pretende recoger de una manera más sistemática y general los eventos y controles más habituales de ***Visual Basic 6.0***. Hay que señalar que en ningún momento se pretende abandonar el carácter introductorio de este manual, y que ***Visual Basic 6.0*** tiene muchas más posibilidades de las que aquí se muestran. Por ejemplo, muchos de los controles y eventos de ***Visual Basic 6.0*** están relacionados con el acceso a bases de datos.

Estos aspectos no se citarán en estos apuntes. Para una información más detallada se puede acudir a un buen libro de referencia o al ***Help*** del programa.

La programación en ***Visual Basic 6.0*** (al menos para ejemplos sencillos) suele proceder del siguiente modo:

1. Se definen interactivamente sobre el formulario los controles que van a constituir la aplicación.

2. Se define para cada control el código con el que se va a responder a cada uno de los eventos. Para ello basta clicar dos veces sobre el control y se abre una ventana de código como la mostrada en la Figura 4.1. En ella ***Visual Basic 6.0*** ha preparado ya el inicio y el final de la función con la que se va a responder al evento. El nombre del evento forma parte del nombre de la función, junto al nombre del control. En el ejemplo del la Figura 4.1 está preparada la función para escribir el código que se ejecutará al producirse el evento ***Click*** sobre el control ***cmbSalir***.

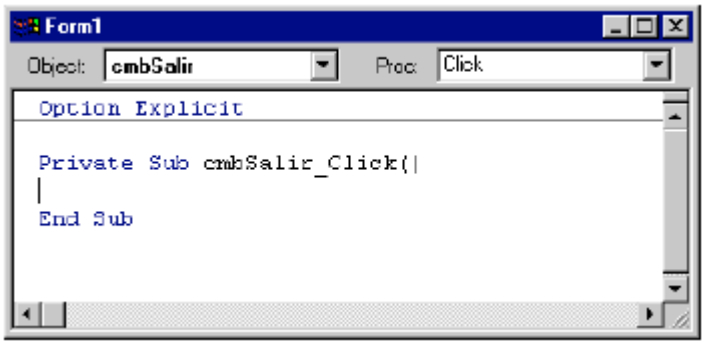


Figura 4.1. Código que gestionará el evento ***Click*** sobre el control de nombre ***cmbSalir***.

En el resto de este capítulo se verán con un cierto detalle los eventos, controles y propiedades más habituales en ***Visual Basic 6.0***.

4.1 EVENTOS

A continuación se presentan brevemente los eventos más normales que reconoce ***Visual Basic 6.0***. Es importante tener una visión general de los eventos que existen en ***Windows 9x/NT/XP/Vista*** porque cada control de los que se verán más adelante tiene su propio conjunto de eventos que reconoce, y otros que no reconoce. Cualquier usuario de las aplicaciones escritas para ***Windows 9x/NT/XP/Vista*** hace uso continuo e intuitivo de los eventos, pero es posible que nunca se haya detenido a pensar en ello.

Para saber qué eventos puede recibir un control determinado basta seleccionarlo y pulsar ***F1***.
De esta forma se abre una ventana del ***Help*** que explica el control y permite acceder a los eventos que soporta.

4.1.1 Eventos generales

4.1.1.1 Carga y descarga de formularios

Cuando se arranca una aplicación, o más en concreto cuando se visualiza por primera vez un formulario se producen varios eventos consecutivos: ***Initialize***, ***Load***, ***Activate*** y ***Paint***.

Cada uno de estos eventos se puede aprovechar para realizar ciertas operaciones por medio de la función correspondiente.

Para inicializar las variables definidas a nivel de módulo se suele utilizar el evento ***Initialize***, que tiene lugar antes que el ***Load***. El evento ***Load*** se activa al cargar un formulario. Con el formulario principal esto sucede al arrancar la ejecución de un programa; con el resto de los formularios al mandarlos cargar desde cualquier procedimiento o al hacer referencia a alguna propiedad o control de un formulario que no esté cargado. Al descargar un formulario se produce el evento ***Unload***. Si se detiene el programa desde el botón ***Stop*** de ***Visual Basic 6.0*** (o del menú correspondiente) o con un ***End,*** no se pasa por el evento ***Unload***. Para pasar por el evento ***Unload*** es necesario cerrar la ventana con el botón de cerrar o llamarlo explícitamente. El evento ***QueryUnload*** se produce antes del evento ***Unload*** y permite por ejemplo enviar un mensaje de confirmación.

El evento ***Load*** de un formulario se suele utilizar para ejecutar una función que dé valor a sus propiedades y a las de los controles que dependen de dicho formulario. No se puede utilizar para dibujar o imprimir sobre el formulario, pues en el momento en que se produce este evento el formulario todavía no está disponible para dichas operaciones. Por ejemplo, si en el formulario debe aparecer la salida del método ***Print*** o de los métodos gráficos ***Pset***, ***Line*** y ***Circle*** (que se estudian en el Capítulo 6 de este manual) puede utilizarse el evento ***Paint*** u otro posterior (por ejemplo, el evento ***GotFocus*** del primer control) pero no puede utilizarse el evento ***Load***.

Se puede ocultar un formulario sin descargarlo con el método ***Hide*** o haciendo la propiedad ***Visible = False***. Esto hace que el formulario desaparezca de la ventana, aunque sus variables y propiedades sigan estando accesibles y conservando sus valores. Para hacer visible un formulario oculto pero ya cargado se utiliza el método ***Show***, que equivale a hacer la propiedad ***Visible = True***, y que genera los eventos ***Activate*** y ***Paint***. Si el formulario no había sido cargado previamente, el método ***Show*** genera los cuatro eventos mencionados.

Cuando un formulario pasa a ser la ventana activa se produce el evento ***Activate*** y al dejar de serlo el evento ***Deactivate***. En el caso de que el formulario que va a ser activo no estuviera cargado ya, primero sucederían los eventos ***Initialize***, ***Load*** y luego los eventos ***Activate*** y ***Paint***.

Todo esto se puede ver y entender con un simple ejemplo, mostrado en la Figura 4.2. Se han de crear dos formularios (***frmPrincipal*** y ***frmSecundario***). El primero de ellos contendrá dos botones (***cmdVerSec*** y ***cmdSalir***) y el segundo tres (***cmdHide***, ***cmdUnload*** y ***cmdTerminate***). El formulario principal será el primero que aparece, y sólo se verá el segundo si se clica en el botón ***Cargar Formulario***. Cuando así se haga, a medida que los eventos antes mencionados se vayan sucediendo, irán apareciendo en pantalla unas cajas de mensajes que tendrán como texto el nombre del evento que se acaba de producir. Según con cual de los tres botones se haga desaparecer el segundo formulario, al volverlo a ver se producirán unos eventos u otros, según se puede ver por los mensajes que van apareciendo con cada evento.

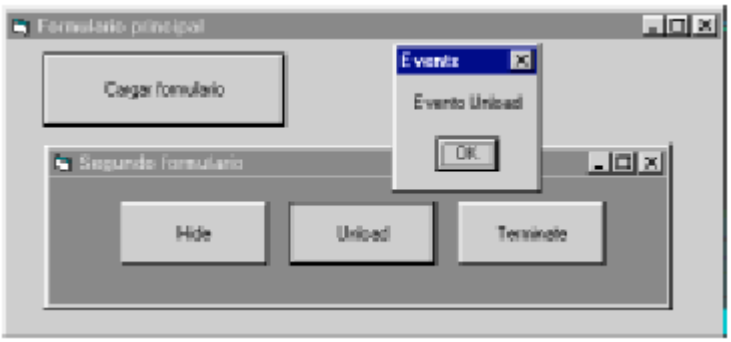


Figura 4.2. Resultado del ejemplo de carga de formularios.

```
' código del form. principal
Private Sub cmdCargar_Click()
      frmSecundario.Show
End Sub

' código del form. secundario
Private Sub cmdHide_Click()
      Hide
End Sub

Private Sub cmdUnload_Click()
      Unload Me
End Sub

Private Sub cmdTerminate_Click()
      Hide
      Set Form2 = Nothing
End Sub

Private Sub Form_Activate()
      MsgBox ("Evento Activate")
End Sub

Private Sub Form_Deactivate()
      MsgBox ("Evento Deactivate")
End Sub
Private Sub Form_Initialize()
      MsgBox ("Evento Initialize")
End Sub
```

```
Private Sub Form_Load()
      MsgBox ("Evento Load")
End Sub

Private Sub Form_Paint()
      MsgBox ("Evento Paint")
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
      MsgBox ("Evento QueryUnload")
End Sub

Private Sub Form_Terminate()
      MsgBox ("Evento Terminate")
End Sub

Private Sub Form_Unload(Cancel As Integer)
      MsgBox ("Evento Unload")
End Sub
```

Es muy interesante realizar este ejemplo y seguir la secuencia de eventos que se producen al hacer aparecer y desaparecer los formularios.

4.1.1.2 Paint

El evento ***Paint*** sucede cuando hay que redibujar un formulario o ***PictureBox.*** Esto sucede cuando esos objetos se hacen visibles después de haber estado tapados por otros, tras haber sido movidos o tras haber sido modificados de tamaño.

4.1.1.3 El foco (focus)

En todas las aplicaciones de ***Windows***, en cualquiera de sus versiones, siempre hay un único control, formulario o ventana que puede recibir clicks del ratón o entradas desde teclado. En cada momento ese control, ventana o formulario es el que dispone del "foco" (***focus***). El objeto que posee el foco está caracterizado por estar ***resaltado*** con letra negrita, con un contorno más vivo o teniendo parpadeando el cursor en él. Este foco puede ser trasladado de un objeto a otro por código o por interacciones del usuario, como por ejemplo clicando con el ratón en distintos puntos de la pantalla o pulsando la tecla ***Tab***. Cada vez que un objeto pierde el foco se produce su evento ***LostFocus*** y, posteriormente, el evento ***GotFocus*** del objeto que ha recibido el foco.

Dos propiedades de muchos controles relacionadas con el foco son ***TabIndex*** y ***TabStop***. ***TabStop*** determina si el foco se va o no a posar en el objeto al pulsar la tecla ***Tab*** (si ***TabStop*** está a***False*** no se puede obtener el foco mediante el tabulador) y ***TabIndex*** determina el orden en el que esto va a suceder. Así al cargar un formulario, el foco estará en aquel objeto cuyo ***TabIndex*** sea 0.

Al pulsar la tecla ***Tab*** el foco irá al objeto que tenga ***TabIndex*** = 1 y así sucesivamente.

Para retroceder en esta lista se pulsa ***Mayúsculas***+***Tab***. La propiedad ***TabIndex*** se puede determinar en tiempo de diseño por medio de la caja de propiedades, del modo habitual.

Cuando a un control se le asigna un determinado valor de ***TabIndex***, ***Visual Basic*** ajusta automáticamente los valores de los demás controles (si tiene que desplazarlos hacia arriba o hacia abajo, lo hace de modo que siempre tengan números consecutivos). Para que un ***formulario*** reciba el foco es necesario que no haya en él ningún control que sea capaz de recibirlo. Un grupo de botones de opción tiene un único ***TabIndex***, es decir, se comporta como un único control. Para elegir una u otra de las opciones se pueden utilizar las flechas del teclado (↑ y ↓).

4.1.1.4 KeyPress, KeyUp y KeyDown

El evento ***KeyPress*** sucede cuando el usuario pulsa y suelta determinada tecla. En este evento el único argumento ***KeyAscii*** es necesario para conocer cuál es el código ASCII de la tecla pulsada. El evento ***KeyDown*** se produce cuando el usuario pulsa determinada tecla y el evento ***KeyUp*** al soltar una tecla.
Los eventos ***KeyUp*** y ***KeyDown*** tienen un segundo argumento llamado ***Shift*** que permiten determinar si esa tecla se ha pulsado estando pulsadas a la vez cualquier combinación de las teclas ***Shift***, ***Alt*** y ***Ctrl***. En un apartado próximo se explica cómo se identifican las teclas pulsadas a partir del argumento ***Shift***.

4.1.2 Eventos relacionados con el ratón

4.1.2.1 Click y DblClick

El evento ***Click*** se activa cuando el usuario pulsa y suelta rápidamente uno de los botones del ratón.

También puede activarse desde código (sin tocar el ratón) variando la propiedad ***Value*** de uno de los controles. En el caso de un formulario este evento se activa cuando el usuario clica sobre una zona del formulario en la que no haya ningún control o sobre un control que en ese momento esté inhabilitado (propiedad ***Enabled*** = *False*). En el caso de un control, el evento se activa cuando el usuario realiza una de las siguientes operaciones:

- Clicar sobre un control con el botón derecho o izquierdo del ratón. En el caso de un botón de comando, de un botón de selección o de un botón de opción, el evento sucede solamente al clicar con el botón izquierdo.

- Seleccionar un registro de alguno de los varios tipos listas desplegables que dispone

Visual Basic.

- Pulsar la ***barra espaciadora*** cuando el foco está en un botón de comando, en un botón de selección o en un botón de opción.

- Pulsar la tecla ***Return*** cuando en un formulario hay un botón que tiene su propiedad ***Default*** = ***True***.

- Pulsar la tecla ***Esc*** cuando en un formulario hay un botón que tiene su propiedad ***Cancel*** = ***True.***

- Pulsar una combinación de teclas aceleradoras (*Alt* + otra tecla, como por ejemplo cuando se despliega el menú ***File*** de ***Word*** con *Alt+F*) definidas para activar un determinado control de un formulario.

También se puede activar el evento ***Click*** desde código realizando una de las siguientes operaciones:

- Hacer que la propiedad ***Value*** de un botón de comando valga ***True***.

- Hacer que la propiedad ***Value*** de un botón de opción valga ***True***

- Modificar la propiedad ***Value*** de un botón de selección.

El evento ***DblClick*** sucede al clicar dos veces seguidas sobre un control o formulario con el botón izquierdo del ratón.

4.1.2.2 MouseDown, MouseUp y MouseMove

El evento ***MouseDown*** sucede cuando el usuario pulsa cualquiera de los botones del ratón, mientras que el evento ***MouseUp*** sucede al soltar un botón que había sido pulsado. El evento ***MouseMove*** sucede al mover el ratón sobre un control o formulario.

Los eventos ***MouseUp*** y ***MouseDown*** tienen algunos argumentos que merecen ser comentados. El argumento ***Button*** indica cuál de los botones del ratón ha sido pulsado o soltado, y el argumento ***Shift*** indica si además alguna de las teclas ***alt***, ***shift*** o ***ctrl*** está también pulsada. La lista con todos los posibles valores de estos argumentos se muestra en la Tabla 4.1:

Cte simbólica	Valor	Acción	Cte simbólica	Valor	Acción
vbLeftButton	1	Botón izdo pulsado o soltado	vbShiftMask	1	Tecla SHIFT pulsada
vbRightButton	2	Botón dcho pulsado o soltado	vbCtrlMask	2	Tecla CTRL pulsada
vbMiddleButton	4	Botón central pulsado o soltado	vbAltMask	4	Tecla ALT pulsada

Tabla 4.1. Valores de los argumentos de los eventos ***MouseUp*** y ***MouseDown***

Con estos valores se aplica la aritmética booleana, lo cual quiere decir que si se pulsan simultáneamente los botones izquierdo y derecho del ratón el argumento ***Button*** valdrá 3 (1+2) y si se pulsan las tres teclas ***shift***, ***ctrl*** y ***alt*** simultáneamente el argumento ***Shift*** valdrá 7 (1+2+4).

4.1.2.3 DragOver y DragDrop

El evento ***DragOver*** sucede mientras se está arrastrando un objeto sobre un control. Suele utilizarse para variar la forma del cursor que se mueve con el ratón dependiendo de si el objeto sobre el que se encuentra el cursor en ese momento es válido para soltar o no. El evento ***DragDrop*** sucede al concluir una operación de arrastrar y soltar. El evento ***DragOver*** requiere de los argumentos que se muestran a continuación:

```
Private Sub Text1_DragOver(Source As Control, _
                           X As Single, Y As Single, State As Integer)
    ...
End Sub
```

Los argumentos de este evento son ***Source*** que contiene el objeto que está siendo arrastrado, ***X*** e ***Y*** que indican la posición del objeto arrastrado dentro del sistema de coordenadas del objeto sobre el que se está arrastrando y ***State*** (que es propio del ***DragOver***, pero no aparece en el ***DragDrop***) que vale 0, 1 ó 2 según se esté *entrando*, *saliendo* o *permaneciendo dentro* del mismo objeto, respectivamente. Es importante señalar que el evento ***DragOver*** es propio del objeto sobre el que se arrastra, no del objeto que es arrastrado.

Continuará.....

Nota de Radacción: El lector puede descargar este capítulo y capítulos anteriores del curso desde la sección ***"Artículos Técnicos"*** en el sitio web de **EduDevices** **(**www.edudevices.com.ar **)**