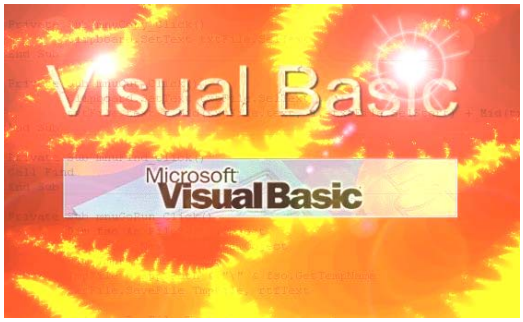
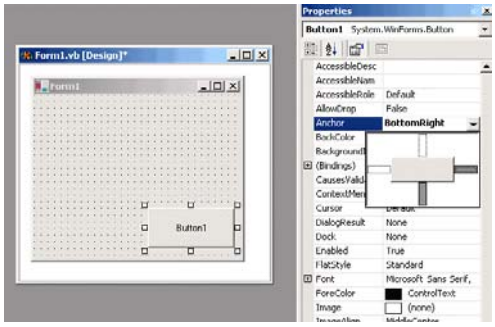
CURSO

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

7. ARCHIVOS Y ENTRADA/SALIDA DE DATOS

En este capítulo se van a describir varias formas de introducir información en el programa, así como de obtener resultados en forma impresa o mediante escritura en un fichero. Se va a presentar una nueva forma interactiva de comunicarse con el usuario, como son las cajas de diálogo ***MsgBox*** e ***InputBox***. Particular interés tiene la lectura y escritura de datos en el disco, lo cual es necesario tanto cuando el volumen de información es muy importante (la memoria RAM está siempre más limitada que el espacio en disco), como cuando se desea que los datos no desaparezcan al terminar la ejecución del programa. Los ficheros en disco resuelven ambos problemas.

También se verá en este capítulo cómo obtener resultados alfanuméricos y gráficos por la impresora.

7.1 CAJAS DE DIÁLOGO INPUTBOX Y MSGBOX

Estas cajas de diálogo son similares a las que se utilizan en muchas aplicaciones de ***Windows***. La caja de mensajes o ***MsgBox*** abre una ventana a través de la cual se envía un mensaje al usuario y se le pide una respuesta, por ejemplo en forma de clicar un botón ***O.K./Cancel***, o ***Yes/No***. Este tipo de mensajes son muy utilizados para confirmar acciones y para decisiones sencillas. La caja de diálogo ***InputBox*** pide al usuario que teclee una frase, por ejemplo su nombre, un título, etc.

La forma general de la función ***MsgBox*** es la siguiente:

```
respuesta = MsgBox("texto para el usuario", tiposBotones, "titulo")
```

donde ***respuesta*** es la variable donde se almacena el valor de retorno, que es un número indicativo del botón clicado por el usuario, de acuerdo con los valores de la **Tabla 7.1.**

La constante simbólica que representa el valor de retorno indica claramente el botón clicado. Los otros dos argumentos son opcionales. El parámetro ***tiposBotones*** es un entero que indica la combinación de botones deseada por el usuario; sus posibles valores se muestran en la Tabla 7.2. También en este caso la constante simbólica correspondiente es suficientemente explícita. Si este argumento se omite se muestra sólo el botón ***O.K.*** El parámetro ***titulo*** contiene un texto que aparece como título de la ventana; si se omite, se muestra en su lugar el nombre de la aplicación.

Valor de retorno	Constante simbólica
1	vbOK
2	vbCancel
3	vbAbort
4	vbRetry
5	vbIgnore
6	vbYes
7	vbNo

Tabla 7.1. Botón clicado por el usuario.

Valor tipos Botones	Constante simbólica
0	vbOKOnly
1	vbOKCancel
2	vbAbortRetryIgnore
3	vbYesNoCancel
4	vbYesNo
5	vbRetryCancel

Tabla 7.2. Botones mostrados en *sgBox*.

Se puede modificar el valor de ***tiposBotones*** de modo que el botón que se activa por defecto cuando se pulsa la tecla ***Intro*** (el botón que tiene el *focus*) sea cualquiera de los botones de la caja.
Para ello basta sumar a ***tiposBotones*** otra constante que puede tomar uno de los tres valores siguientes: ***0*** (*vbDefaulButton1*, que representa el primer botón), ***256*** (*vbDefaulButton2*, que representa el segundo botón) y ***512*** (*vbDefaulButton3*, que representa el tercer botón).

Figura 7.1. Ejemplo de caja ***MsgBox***.

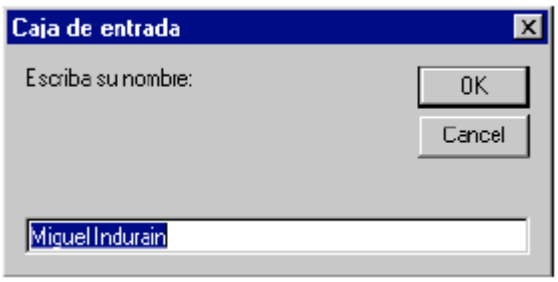


Figura 7.2. Ejemplo de caja de ***InputBox***.

Finalmente, se puede incluir en el mensaje un *icono ad-hoc* por el mismo procedimiento de sumarle al argumento ***tiposBotones*** una nueva constante numérica con los siguientes valores y significados definidos por la constante simbólica apropiada: 16 (*vbCritical*), 32 (*vbQuestion*), 48 (*vbExclamation*) y 64 (*vbInformation*). Es obvio que, por los propios valores considerados, al sumar estas constantes o las anteriores al argumento ***tiposBotones***, la información original descrita en la **Tabla 7.2** no se pierde. La Figura 7.1 muestra un ejemplo de caja ***MsgBox*** resultado de ejecutar el comando siguiente:

```
lblBox.Caption = MsgBox("Pulse un botón: ", 2 + 256 + 48, _
                        "Caja de mensajes")
```

donde el "2" indica que deben aparecer los botones ***Abort***, ***Retry*** y ***Cancel***, el "256" indica que el *botón por defecto* es el segundo (***Retry***) y el "48" indica que debe aparecer el *icono de exclamación*.

Por otra parte, la forma general de la función ***InputBox*** es la siguiente:

```
texto = InputBox("texto para el usuario", "titulo", "default", left, top)
```

donde ***texto*** es la variable donde se almacena el valor de retorno, que es el texto tecleado por el usuario. Los parámetros "*texto para el usuario*" y ***titulo*** tienen el mismo significado que en ***MsgBox***. El parámetro ***default*** es un texto por defecto que aparece en la caja de texto y que el usuario puede aceptar, modificar o sustituir; el contenido de esta caja es lo que en definitiva esta función devuelve como valor de retorno. Finalmente, ***left*** y ***top*** son las coordenadas de la esquina superior izquierda de la ***InputBox***; si se omiten, ***Visual Basic 6.0*** dibuja esta caja centrada en horizontal y algo por encima de la mitad de la pantalla en vertical. La **Figura 7.2** muestra un ejemplo de caja ***InputBox*** resultado de ejecutar el comando siguiente:

```
lblBox.Caption = InputBox("Escriba su nombre: ", _
                          "Caja de entrada", "Miguel Indurain")
```

donde el nombre que aparece por defecto es el del mejor ciclista de los últimos tiempos. Este nombre aparece seleccionado y puede ser sustituido por otro que teclee el usuario.

7.2 MÉTODO PRINT

Este método permite escribir texto en ***formularios***, cajas ***pictureBox*** y en un objeto llamado ***Printer*** que se verá un poco más adelante.

7.2.1 Características generales

La forma general del método ***Print*** se explica mejor con algunos ejemplos como los siguientes:

```
pctBox.Print "La distancia es: "; Dist; " km."
pctBox.Print 123; 456; "San"; "Sebastián"
pctBox.Print 123, 456, "San", "Sebastián"
```

cuyo resultado se puede ver en la Figura 7.3 (puede variar dependiendo del tipo y tamaño de las letras):

De estos ejemplos se pueden ya sacar algunas conclusiones:

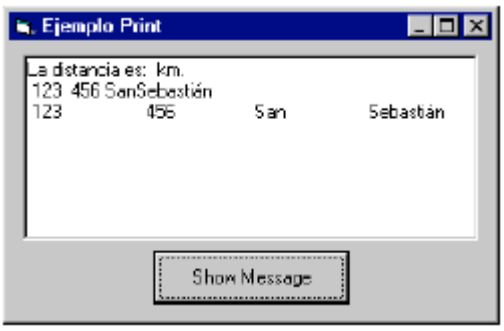


Figura 7.3: Ejemplo del método ***Print***.

1. El método ***Print*** recibe como datos una *lista de variables y/o cadenas de caracteres*. Las cadenas son impresas y las variables se sustituyen por su valor.

2. Hay dos tipos básicos de ***separadores*** para los elementos de la lista.
   El carácter ***punto y coma*** (;) hace que se escriba inmediatamente a continuación de lo anterior. La ***coma*** (,) hace que se vaya al comienzo de la siguiente ***área de salida***. Con letra de paso constante como la Courier las áreas de salida empiezan cada 14 caracteres, es decir en las columnas 1, 15, 29, etc. Con letras de paso variable esto se hace sólo de modo aproximado.

3. Las constantes numéricas positivas van precedidas por un espacio en blanco y separadas entre sí por otro espacio en blanco. Si son negativas el segundo espacio es ocupado por el signo menos (-).

4. El tipo y tamaño de letra que se utiliza depende de la propiedad ***Font*** del formulario, Objeto ***PictureBox*** u objeto ***Printer*** en que se esté escribiendo.

Existen otros separadores tales como ***Tab(n)*** y ***Spc(n)***. El primero de ellos lleva el punto de inserción de texto a la columna ***n***, mientras que el segundo deja ***n*** espacios en blanco antes de seguir escribiendo. ***Tab*** sin argumento equivale a la coma (,). Estos espaciadores se utilizan en combinación con el punto y coma (;), para separarlos de los demás argumentos.

Por defecto la salida de cada método ***Print*** se escribe en una nueva línea, pero si se coloca un punto y coma al final de un método ***Print***, el resultado del siguiente ***Print*** se escribe en la misma línea.

Puede controlarse el lugar del formulario o control donde se imprime la salida del método ***Print***. Esta salida se imprime en el lugar indicado por las propiedades ***CurrentX*** y ***CurrentY*** del formulario o control donde se imprime. Cambiando estas propiedades se modifica el lugar de impresión, que por defecto es la esquina superior izquierda. Existen unas funciones llamadas ***TextWidth(string)*** y ***TextHeight(string)*** que devuelven la anchura y la altura de una cadena de caracteres pasada como argumento.
Estas funciones pueden ayudar a calcular los valores más adecuados para las propiedades ***CurrentX*** y ***CurrentY***.

La función ***str(valor_numérico)*** convierte un número en cadena de caracteres para facilitar su impresión. En realidad, es lo que ***Visual Basic 6.0*** ha hecho de modo implícito en los ejemplos anteriores. En versiones anteriores del programa era necesario que el usuario realizase la conversión de modo explícito.

7.2.2 Función Format

La función ***Format*** realiza las conversiones necesarias para que ciertos datos numéricos o de otro tipo puedan ser impresos con ***Print***. Como se ha visto, en el caso de las variables numéricas esto no es imprescindible, pero la función ***Format*** permite controlar el número de espacios, el número de decimales, etc. En el caso de su aplicación a objetos tipo *fecha* (***date***) y *hora* (***time***) la aplicación de ***Format*** es imprescindible, pues ***Print*** no los escribe directamente. La forma general de esta función es la siguiente:

```
Format(expresion, formato)
```

donde ***expresion*** es una variable o expresión y ***formato*** -que es opcional- describe el formato deseado para el resultado. El valor de retorno es una cadena de caracteres directamente utilizable en ***Print***. Para *fechas* existen formatos predefinidos tales como *"General Date"*, *"Long Date"*, *"Medium Date"* y *"Short Date"*; para la hora los formatos predefinidos son *"Long Time"*, *"Medium Time"* y *"Short Time"*. Además existe la posibilidad de que el usuario defina sus propios formatos (ver ***User-Defined Date/Time Formats (Format Function)***, en el ***Help*** del programa).
El usuario también puede definir sus propios formatos numéricos y de cadenas de caracteres.

A diferencia de la función ***Str***, la función ***Format*** no deja espacio en blanco para el signo de los números positivos. Para una información más detallada sobre la función ***Format*** consultar el ***Help***.

7.3 UTILIZACIÓN DE IMPRESORAS

Visual Basic 6.0 permite obtener por la impresora gráficos y texto similares a los que se pueden obtener por la pantalla, aunque con algunas diferencias de cierta importancia. Existen dos formas de imprimir: la primera mediante el método ***PrintForm***, y la segunda utilizando el objeto ***Printer***, que es un objeto similar al objeto ***PictureBox***. Ambos métodos tienen puntos fuertes y débiles que se comentarán a continuación.

7.3.1 Método PrintForm

El método ***PrintForm*** permite imprimir un formulario con sus controles y con los resultados de los métodos gráficos (***pSet***, ***line*** y ***circle***) y del método ***print***. Para ello la propiedad ***AutoRedraw*** del formulario tiene que estar puesta a ***true***, y los métodos citados tienen que estar llamados desde un evento distinto del ***Paint***. Lo único que no se dibuja del formulario es la *barra de títul*o.

Este sistema de impresión es muy sencillo de utilizar, pero tiene el inconveniente de que el resultado se imprime con la misma resolución de la pantalla (entre 50 y 100 puntos por pulgada), no aprovechando por tanto la mayor resolución que suelen tener las impresoras (300 ó 600 puntos por pulgada).

7.3.2 Objeto Printer

Este segundo sistema tiene la ventaja de que permite aprovechar plenamente la resolución de la impresora, pero ***no permite dibujar controles*** sino sólo los métodos gráficos habituales (***pSet, line*** y ***circle***), el método ***print*** y un método no visto hasta ahora que es ***PaintPicture***.

Para ***Visual Basic 6.0*** la impresora es un objeto gráfico más, similar a los formularios y a las cajas gráficas ***PictureBox***. Como tal objeto gráfico tiene sus propiedades generales (*DrawStyle*, *BackColor*, *ForeColor*, etc.), además de otras propiedades específicas de la impresora, como *DeviceName*, *DriverName*, *Orientation*, *Copies*, etc. Para más información puede utilizarse el ***Help***, buscando ***Printer object***. En principio se utiliza la impresora por defecto del PC, pero ***Visual Basic*** mantiene una ***Printers Collection***, que es algo así como un array de impresoras disponibles. A partir de esta ***Printers Collection*** se puede cambiar a la impresora que se desee.

El objeto ***Printer*** tiene un método llamado ***EndDoc*** para enviar realmente a la impresora el resultado de los métodos gráficos. El método ***PaintPicture*** permite incorporar el contenido de ***ficheros gráficos*** a un *formulario*, ***PictureBox*** o ***Printer***. Su forma general es:

```
object.PaintPicture pictProp X, Y, Width, Height
```

donde ***pictProp*** indica el gráfico (coincide con la propiedad ***picture*** de ***PictureBox***), ***X*** e ***Y*** indican las coordenadas de inserción y los dos últimos parámetros las dimensiones (opcionales).

7.4 CONTROLES FILELIST, DIRLIST Y DRIVELIST

Uno de los problemas que hay que resolver para leer o escribir en ficheros de disco es ser capaces de localizar interactivamente los correspondientes ficheros, de modo análogo a como se realiza con los comandos ***File/Open*** o ***File/Save As*** de ***Word***, ***Excel*** o de cualquier otra aplicación. Este tipo de operaciones se pueden hacer mucho más fácilmente con los ***Common Dialog Controls*** vistos en el apartado 4.4, ***aconsejando por lo tanto su uso***.

A pesar de ello, aquí se van a explicar los controles específicos de que dispone

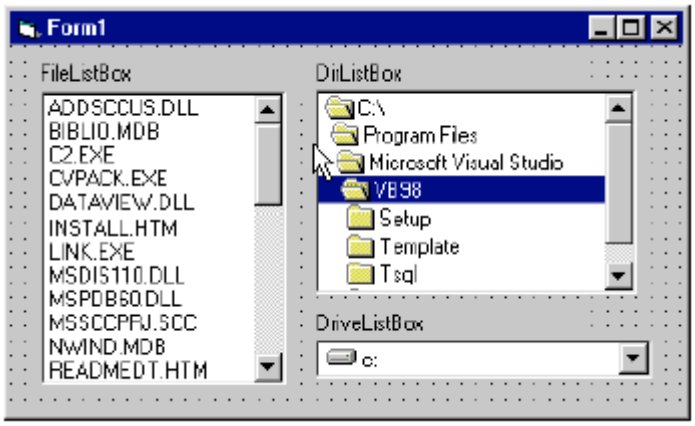


Figura 7.4. Cajas ***DriveListBox***, ***DirListBox*** y ***FileListBox***.

Visual Basic 6.0 dispone de tres controles que facilitan el recorrer el árbol de ficheros y de directorios, localizando o creando interactivamente un fichero determinado. Estos controles son el ***FileListBox*** (para ficheros), el ***DirListBox*** (para directorios) y el ***DriveListBox*** (para unidades de disco). La **Figura 7.4** muestra estos tres controles, junto con unas etiquetas que los identifican. Los dos primeros son ***listas***, mientras que el tercero es una caja de tipo ***combo***.

En principio estos controles, cuando se colocan en un formulario tal como se muestra en la Figura 7.4, están desconectados. Quiere esto decir que al cambiar la unidad de disco (***drive***) no se muestran en la caja ***dirListBox*** los directorios correspondientes a la nueva unidad de disco. Por otra parte, al cambiar de directorio tendrán que cambiar de modo acorde los ficheros en la caja ***fileListBox***. La dificultad de conectar estas cajas no es grande, pero sí hay que saber cómo se hace pues depende de propiedades de estas cajas que no aparecen en la ventana de propiedades (ventana ***Properties***)en modo de diseño, y que sólo están accesibles en modo de ejecución. De entre estas propiedades las más importantes son las siguientes:

1. La ***DriveListBox*** tiene una propiedad llamada ***drive*** que recoge la unidad seleccionada por el usuario (puede ser una unidad física como el disco ***c:***\ o una unidad lógica asignada por el usuario a otro disco o directorio en un servidor o en otro ordenador de la red).

2. La propiedad ***path*** de la caja ***DirListBox*** determina el ***drive*** seleccionado y por tanto que directorios se muestran en dicha caja.

3. Finalmente, una propiedad también llamada ***path*** de la caja ***FileListBox*** determina el directorio que contiene los ficheros mostrados.

Para enlazar correctamente las cajas de discos, directorios y ficheros se puede utilizar el

evento ***change***, de tal forma que cada vez que el usuario cambia la unidad de disco se cambia el ***path*** del directorio y cada vez que se cambia el directorio se cambia el ***path*** de los ficheros. Esto puede hacerse con el código siguiente:

```
Private Sub dirPrueba_Change()
      filPrueba.Path = dirPrueba.Path
End Sub

Private Sub drvPrueba_Change()
      dirPrueba.Path = drvPrueba.Drive
End Sub
```

La caja ***FileListBox*** tiene una propiedad llamada ***FileName*** que contiene el nombre del fichero seleccionado por el usuario. Para tener el ***path*** completo del fichero basta anteponerle la propiedad ***path*** de la ***fileListBox***, que incluye el directorio y el drive, y la barra invertida (\). Si el usuario introduce ***FileName*** incluyendo el ***path***, ***Visual Basic*** actualiza también de modo automático la propiedad ***Path*** de ***FileListBox***. El usuario se debe preocupar de utilizar el evento ***Change*** para actualizar el ***Path*** de la caja ***DirListBox*** y la propiedad ***Drive*** de ***DriveListBox***.

Otra propiedad importante es la propiedad ***Pattern***, que indica los tipos de ficheros que se mostrarán en la caja. El valor por defecto es ***"*.*"***, lo cual hace que se muestren todos los ficheros.
Si su valor fuese ***"*.doc"*** sólo se mostrarían los ficheros con esta extensión. La propiedad ***Pattern*** admite varias opciones separadas por untos y coma (***"*.doc; *.dot"***).

7.5 TIPOS DE FICHEROS

Tanto en ***Windows*** como en ***Visual Basic 6.0*** existen, principalmente, dos tipos de archivos:

1. ***Ficheros ASCII*** o ficheros de texto. Contienen caracteres codificados según el código ASCII y se pueden leer con cualquier editor de texto como ***Notepad***. Suelen tener extensión ****.txt*** o ****.bat***, pero también otras como ****.m*** para los programas de ***Matlab***, ****.c*** para los ficheros fuente de C o ****.cpp*** para los ficheros fuente de C++.

2. ***Ficheros binarios***: Son ficheros imagen de los datos o programas tal como están en La memoria del ordenador. No son legibles directamente por el usuario. Tienen la ventaja de que ocupan menos espacio en disco y que no se pierde tiempo y precisión cambiándolos a formato ASCII al escribirlos y al leerlos en el disco.

Con ***Visual Basic 6.0*** se pueden leer tanto ficheros ASCII como ficheros binarios. Además el acceso a un fichero puede ser de tres formas principales.

1. ***Acceso secuencial***. Se leen y escriben los datos como si se tratara de un libro: siempre a continuación del anterior y sin posibilidad de volver atrás o saltar datos. Si se quiere acceder a un dato que está hacia la mitad de un fichero, habrá que pasar primero por todos los datos anteriores. Los ficheros de texto tienen acceso secuencial.

2. ***Acceso aleatorio*** *(random)*: Permiten acceder directamente a un dato sin tener que pasar por todos los demás, y pueden acceder a la información en cualquier orden. Tienen la limitación de que los datos están almacenados en unas unidades o bloques que se llaman ***registros***, y que todos los registros que se almacenan en un fichero deben ser del mismo tamaño. Los ficheros de acceso aleatorio son ficheros binarios.

3. ***Acceso binario***. Son como los de acceso aleatorio, pero el acceso no se hace por *registros* sino por ***bytes***.

Antes de poder leer o escribir en un fichero hay que abrirlo por medio de la sentencia ***Open***.

En esta sentencia hay que especificar qué tipo de acceso se desea tener, distinguiendo también si es para lectura (***input***), escritura (***output***) o escritura añadida (***append***).

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)**