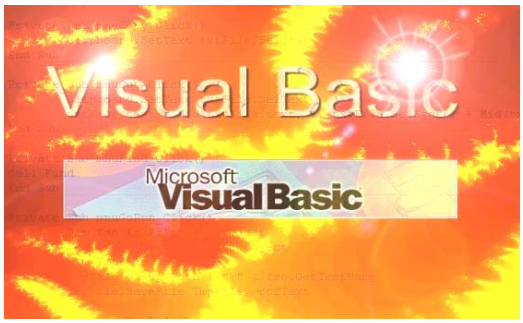
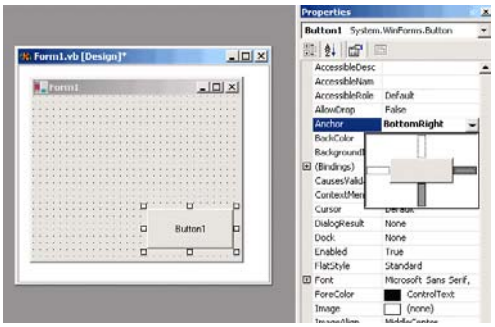
CURSO

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

2. ENTORNO DE PROGRAMACIÓN VISUAL BASIC 6.0

2.8 UTILIZACIÓN DEL CODE EDITOR

El *editor de código* o ***Code Editor*** de ***Visual Basic 6.0*** es la ventana en la cual se escriben las sentencias del programa. Esta ventana presenta algunas características muy interesantes que conviene conocer para sacar el máximo pertido a la aplicación.

Para abrir la ventana del editor de código se elige ***Code*** en el menú ***View***. También se abre clicando en el botón ***View Code*** de la ***Project Window***, o clicando dos veces en el formulario o en cualquiera de sus controles. Cada formulario, cada módulo de clase y cada módulo estándar tienen su propia ventana de código. La Figura 2.10 muestra un aspecto típico de la ventana de código.
Aunque el aspecto de dicha ventana no tiene nada de particular, el ***Code Editor*** de ***Visual Basic 6.0*** ofrece muchas ayudas al usuario que requieren una explicación más detenida.

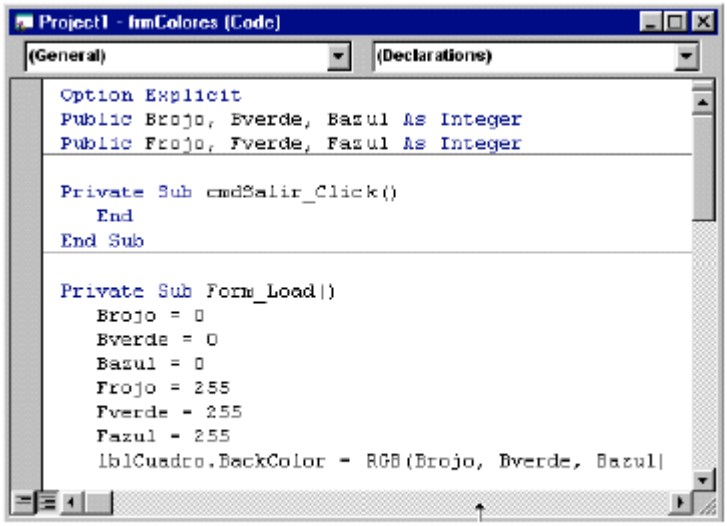


Figura 2.10. Ventana del ***Code Editor***.

En primer lugar, el ***Code Editor*** utiliza un ***código de colores*** (accesible y modificable en ***Tools/Options/Editor Format***) para destacar cada elemento del programa. Así, el código escrito por el usuario aparece en negro, las palabras clave de ***Basic*** en azul, los comentarios en verde, los errores en rojo, etc. Esta simple ayuda visual permite detectar y corregir problemas con más facilidad.

En la parte superor de esta ventana aparecen dos ***listas desplegables***. La de la izquierda corresponde a los distintos elementos del formulario (la parte ***General***, que es común a todo el formulario; el propio formulario y los distintos controles que están incluidos en él).

La lista desplegable de la derecha muestra los distintos procedimientos que se corresponden con el elemento seleccionado en la lista de la izquierda. Por ejemplo, si en la izquierda está seleccionado un botón de comando, en la lista de la derecha aparecerá la lista de todos los posibles procedimientos Sub que pueden generar sus posibles eventos. Estas dos listas permiten localizar fácilmente el código que se desee programar o modificar.

El código mostrado en la Figura 2.10 contiene en la parte superior una serie de declaraciones de variables y la opción de no permitir utilizar variables no declaradas (***Option Explicit***). Ésta es la parte ***General*** de código del formulario. En esta parte también se pueden definir ***funciones*** y ***procedimientos Sub*** no relacionados con ningún evento o control en particular. A continuación aparecen dos ***procedimientos Sub*** (el segundo de ellos incompleto) que se corresponden con el evento ***Click*** del botón ***cmdSalir*** y con el evento ***Load*** del formulario. Estos procedimientos están separados por una línea, que se activa con ***Procedure Separator*** en ***Tools/Options/Editor***.

Para ver todos los procedimientos del formulario y de sus controles simultáneamente en la misma ventana (con o sin separador) o ver sólo un procedimiento (el seleccionado en las listas desplegables) se pueden utilizar los dos pequeños botones que aparecen en la parte inferior izquierda de la ventana. El primero de ellos es el ***Procedure View*** y el segundo el ***Full Module View***. Esta opción está también accesible en ***Tools/Options/Editor***.

Otra opción muy interesante es la de completar automáticamente el ***código (Automatric Completion Code***). La Figura 2.11 muestra cómo al teclear el punto (o alguna letra inicial de una propiedad después del punto) detrás del nombre de un objeto, automáticamente se abre una lista con las propiedades de ese objeto. Pulsando la tecla ***Tab*** se introduce el nombre completo de la propiedad seleccionada. A esta característica se le conoce como ***AutoListMembers***.
Por otra parte, la opción ***AutoQuickInfo*** hace que al comenzar a teclear el nombre de una función aparezca información sobre esa función: nombre, argumentos y valor de retorno (ver Figura 2.12). Tanto la opción ***AutoListMembers*** como la opción ***AutoQuickInfo*** se activan en el cuadro de diálogo que se abre con ***Tools/Options/Editor.***

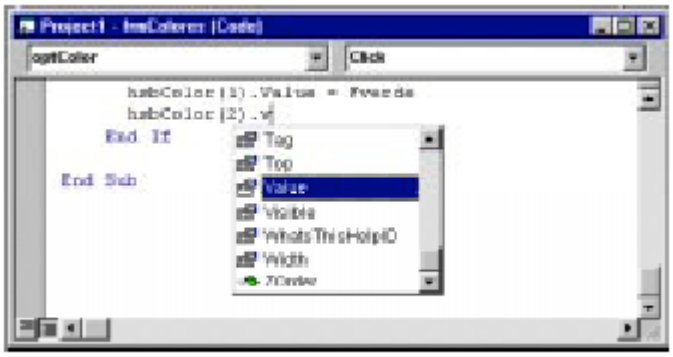


Figura 2.11. Inserción automática de propiedades.

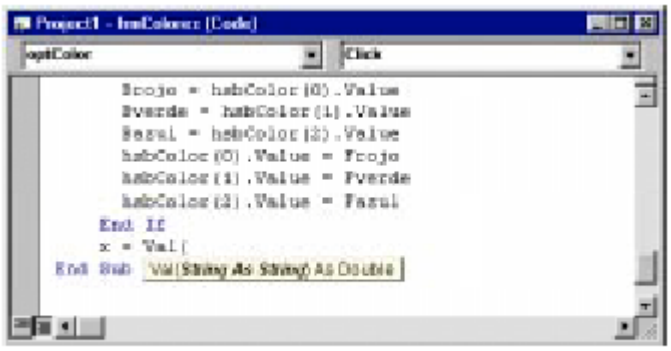


Figura 2.12. Ayuda para inserción de funciones.

2.9 UTILIZACIÓN DEL DEBUGGER

Cualquier programador con un mínimo de experiencia sabe que una parte muy importante (muchas veces la mayor parte) del tiempo destinado a la elaboración de un programa se destina a la ***detección y corrección de errores***. Casi todos los entornos de desarrollo disponen hoy en día de potentes herramientas que facilitan la depuración de los programas realizados. La herramienta más utilizada para ello es el *Depurador* o ***Debugger***.

La característica principal del ***Debugger*** es que permite ejecutar parcialmente el programa, deteniendo la ejecución en el punto deseado y estudiando en cada momento el valor de cada una de las variables. De esta manera se facilita enormemente el descubrimiento de las fuentes de errores.

2.9.1 Ejecución controlada de un programa

Para ejecutar ***parcialmente*** un programa se pueden utilizar varias formas. Una de ellas consiste en incluir ***breakpoints*** (puntos de parada de la ejecución) en determinadas líneas del código. Los breakpoints se indican con un punto grueso en el margen y un cambio de color de la línea, tal como se ve en la Figura 2.13.
En esta figura se muestra también la barra de herramientas ***Debug***. El colocar un ***breakpoint*** en una línea de código implica que la ejecución del programa se detendrá al llegar a esa línea. Para insertar un ***breakpoint*** en una línea del código se utiliza la opción ***Toggle Breakpoint*** del menú ***Debug***, con el botón del mismo nombre ( ) o pulsando la tecla <F9>, estando el cursor posicionado sobre la línea en cuestión. Para borrarlo se repite esa operación.

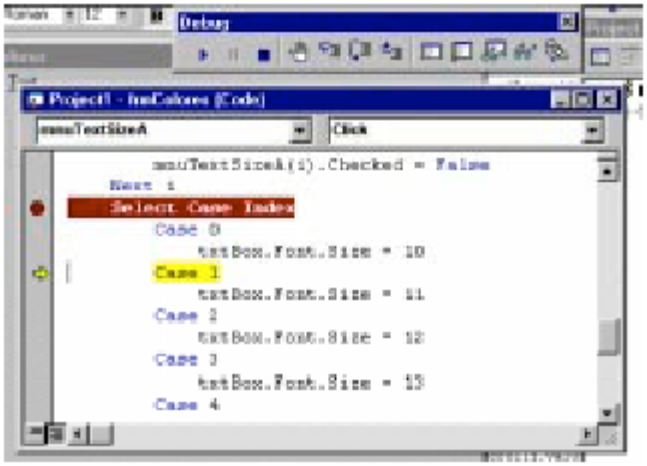


Figura 2.13. Utilización del ***Debugger***.

Cuando la ejecución está detenida en una línea aparece una flecha en el margen izquierdo, tal como puede verse también en la Figura 2.13. En ese momento se puede consultar el valor de cualquier variable que sea accesible desde ese punto en la ventana de depuración (***Debug Window***).

Un poco más adelante se verán varias formas de hacer esto.

En la Figura 2.13 se puede observar como la ejecución del programa está detenida en la línea coloreada o recuadrada, con una flecha en el margen izquierdo. Se puede observar también la variación del color de fondo de la línea anterior debido a que en ella hay un ***breakpoint***.

De todos modos no es estrictamente necesaria la utilización de ***breakpoints*** para la ejecución parcial de un programa. Esto se puede hacer también ejecutando el programa ***paso a paso*** (o *línea a línea*). Para hacer esto hay varias opciones: pulsando la tecla <F8>, seleccionando la opción ***Step Into*** del menú ***Run*** o clicando en el botón correspondiente ().

Esta instrucción hace que se ejecute una línea del código. En el caso de que ésta se trate de la llamada a un procedimiento o función, la ejecución se trasladará a la primera línea de ese procedimiento o función. En el caso de que se desee ejecutar toda la función en un único paso (por ejemplo porque se tiene constancia de que esa función funciona correctamente) se puede hacer mediante la opción ***Step Over***, pulsando las teclas <mayúsculas> y <F8> simultáneamente, o clicando en el botón correspondiente ().

En este caso la ejecución se traslada a la línea inmediatamente posterior a la llamada a la función. En el caso de que la línea a ejecutar no sea la llamada a una función ambas opciones (***Step Into*** y ***Step Over***) operan idénticamente. El comando y botón ***Step Out*** () hace que se salga de la función o procedimiento que se está ejecutando y que la ejecución se detenga en la sentencia inmediatamente siguiente a la llamada a dicha función o procedimiento.

La utilización del ***Debugger*** permite también otras opciones muy interesantes como la de ejecutar el programa hasta la línea en la que se encuentre posicionado el cursor (con ***Step To Cursor*** o ***Ctrl+<F8>***); la de continuar con la ejecución del programa hasta el siguiente ***breakpoint*** en el caso de que lo haya o hasta el final del mismo si no hay ninguno (con ***Continue***, botón  o <F5>); y la posibilidad de volver a comenzar la ejecución (con ***Restart*** o ***Mayúsculas+ <F5>***). Además de las ya mencionadas, también existe la posibilidad de detener momentáneamente la ejecución del programa mediante el botón ***Pause*** () o la combinación de teclas ***Ctrl+Pausa***.

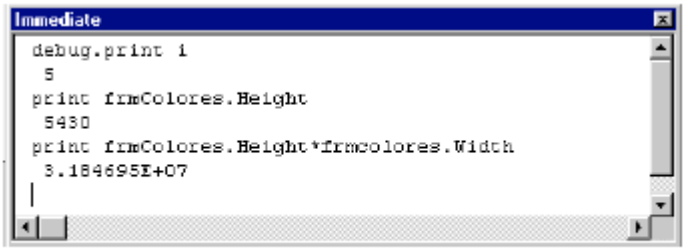


Figura 2.14. Ventana ***Immediate***.

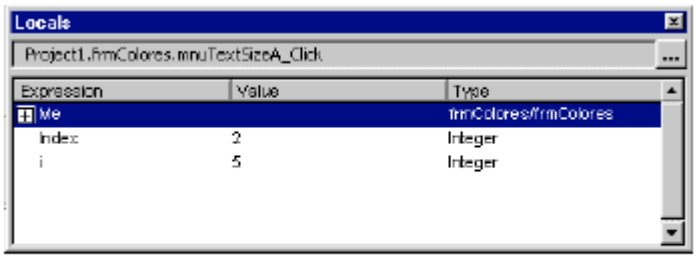


Figura 2.15. Ventana ***Locals***.

2.9.2 Ventanas Immediate, Locals y Watches

El ***Debugger*** de ***Visual Basic 6.0*** dispone de varias formas para consultar el valor de variables y propiedades, así como para ejecutar funciones y procedimientos comprobando su correcto funcionamiento. En ello juegan un papel importante tres tipos de ventanas: ***Immmediate***, ***Locals*** y ***Watch***.

La ventana ***Immediate*** (ver Figura 2.14) permite realizar diversas acciones:

1. Imprimir el valor de cualquier variable y/o propiedad accesible la función o Procedimiento que se está ejecutando. Esto se puede hacer utilizando el método ***Print VarName*** (o su equivalente ***?VarName***) directamente en dicha ventana o introduciendo en el código del programa sentencias del tipo ***Debug.Print VarName***. En este último caso el valor de la variable o propiedad se escribe en la ventana ***Immediate*** sin necesidad de parar la ejecución del programa. Además esas sentencias se guardan con el formulario y no hay que volver a escribirlas para una nueva ejecución. Cuando se compila el programa para producir un ejecutable las sentencias ***Debug.Print*** son ignoradas. La utilización del método ***Print*** se explica en el Apartado 7.2.

2. Asignar valores a variables y propiedades cuando la ejecución está detenida y proseguir la ejecución con los nuevos valores. Sin embargo, no se pueden crear nuevas variables.

3. Ejecutar expresiones y probar funciones y procedimientos incluyendo en la ventana ***Immediate*** la llamada correspondiente.

La ventana ***Locals***, mostrada en la Figura 2.15, muestra el valor de todas las variables visibles en el procedimiento en el que está detenida la ejecución.

Otra opción que puede resultar útil es la de conocer permanentemente el valor de una variable sin tener que consultarlo cada vez. Para conocer inmediatamente el valor de una variable se puede utilizar la ventana ***Quick Watch***, mostrada en la Figura 2.16. Para observar continuamente el valor de una variable, o expresión hay que añadirla a la ventana ***Watches***. Esto se hace con la opción ***Add Watch…*** del menú ***Debug***.

El valor de las variables incluidas en la ventana ***Watches*** (ver Figura 2.18) se actualiza automáticamente, indicándose también cuando no son accesibles desde el procedimiento que se esté ejecutando (***Out of Context***).

La ventana ***Add Watch*** mostrada en la Figura 2.17 permite introducir ***Breaks*** o paradas del programa condicionales, cuando se cumple cierta condición o cuando el valor de la variable cambia.

Las capacidades de ***Visual Basic 6.0*** para *vigilar* el valor de las variables pueden activarse desde el menú ***Debug*** o con algunos botones en la barra de herramientas ***Debug*** ().

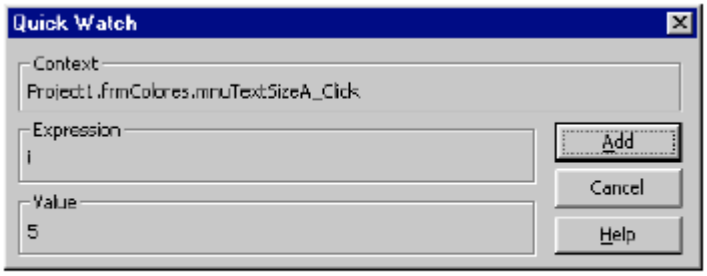


Figura 2.16. Ventana ***Quick Watch***.

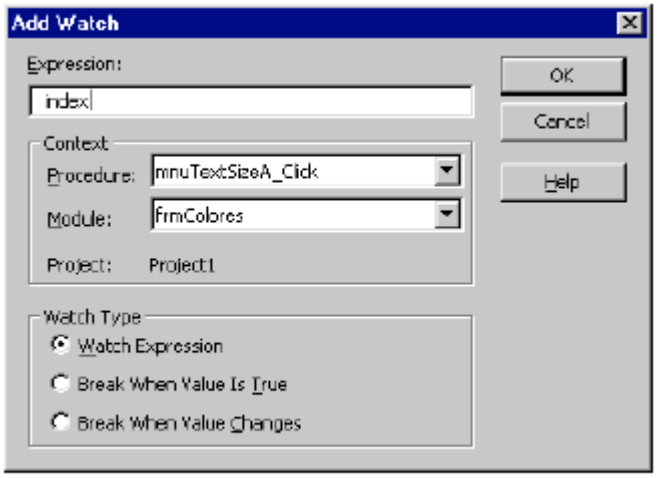


Figura 2.17. Ventana ***Add Watch***.

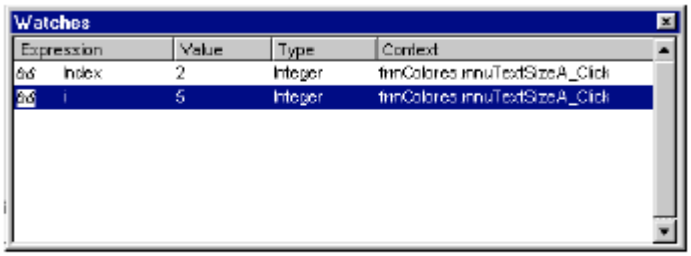


Figura 2.18. Ventana ***Watches***.

2.9.3 Otras posibilidades del Debugger

El ***Debugger*** de ***Visual Basic 6.0*** permite no sólo saber qué sentencia va a ser la próxima en ejecutarse (con ***Debug/Show Next Statement***), sino también decidir cuál va aser dicha sentencia (con ***Debug/Set Next Statement***), pudiendo cambiar de esta forma el curso habitual de la ejecución: saltando sentencias, volviendo a una sentencia ya ejecutada, etc. ***Visual Basic 6.0*** puede dar también información sobre las llamadas a funciones y procedimientos. Esto se hace con el comando ***View/Call Stack*** o con el botón corresponndiente de la barra ***Debug*** ( ).

De esta manera puede conocerse qué función ha llamado a qué función hasta la sentencia donde la ejecución está detenida.

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)**

WWW.EDUDEVICES.COM.AR