FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

A LOS ASISTENTES A LOS CURSOS

Las autoridades de la Facultad de Ingeniería, por conducto del jefe de la División de Educación Continua, otorgan una constancia de asistencia a quienes cumplan con los requisitos establecidos para cada curso.

El control de asistencia se llevará a cabo a través de la persona que le entregó las notas. Las inasistencias serán computadas por las autoridades de la División, con el fin de entregarle constancia solamente a los alumnos que tengan un mínimo de 80% de asistencias.

Pedimos a los asistentes recoger su constancia el día de la clausura. Estas se retendrán por el periodo de un año, pasado este tiempo la DECFI no se hará responsable de este documento.

Se recomienda a los asistentes participar activamente con sus ideas y experiencias, pues los cursos que ofrece la División están planeados para que los profesores expongan una tesis, pero sobre todo, para que coordinen las opiniones de todos los interesados, constituyendo verdaderos seminarios.

Es muy importante que todos los asistentes llenen y entreguen su hoja de inscripción al inicio del curso, información que servirá para integrar un directorio de asistentes, que se entregará oportunamente.

Con el objeto de mejorar los servicios que la División de Educación Continua ofrece, al final del curso deberán entregar la evaluación a través de un cuestionario diseñado para emitir juicios anónimos.

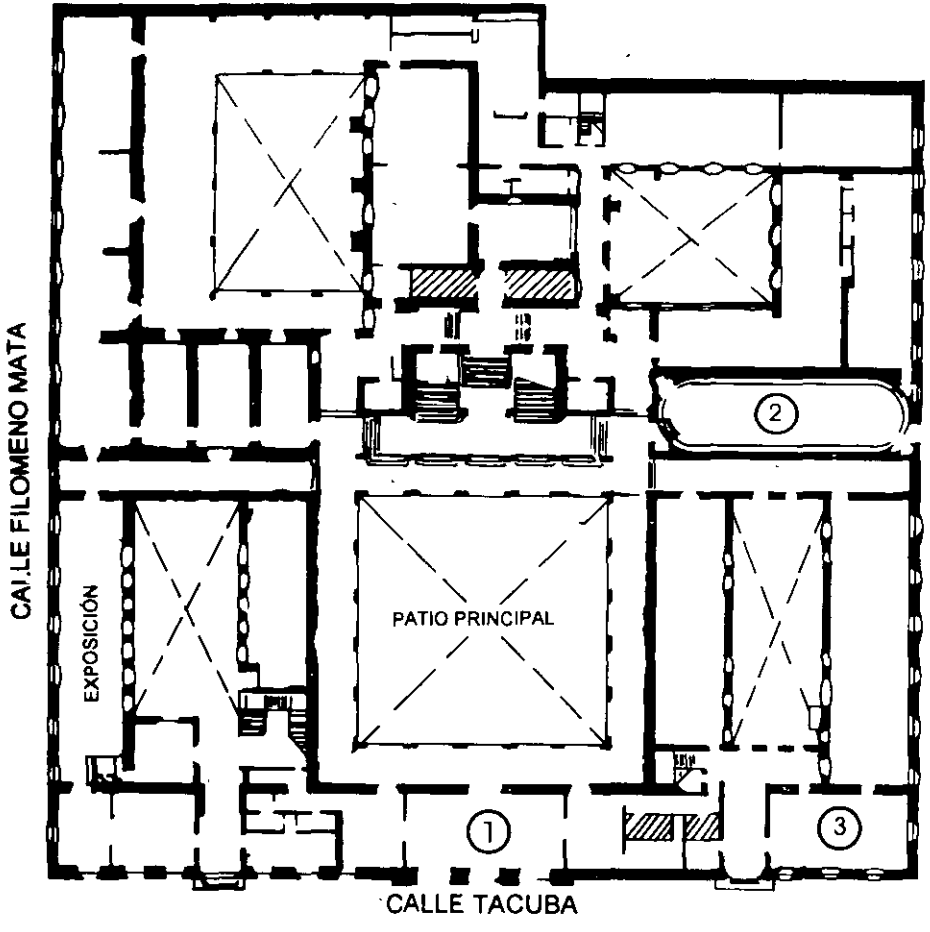
Se recomienda llenar dicha evaluación conforme los profesores impartan sus clases, a efecto de no llenar en la última sesión las evaluaciones y con esto sean más fehacientes sus apreciaciones.

Atentamente

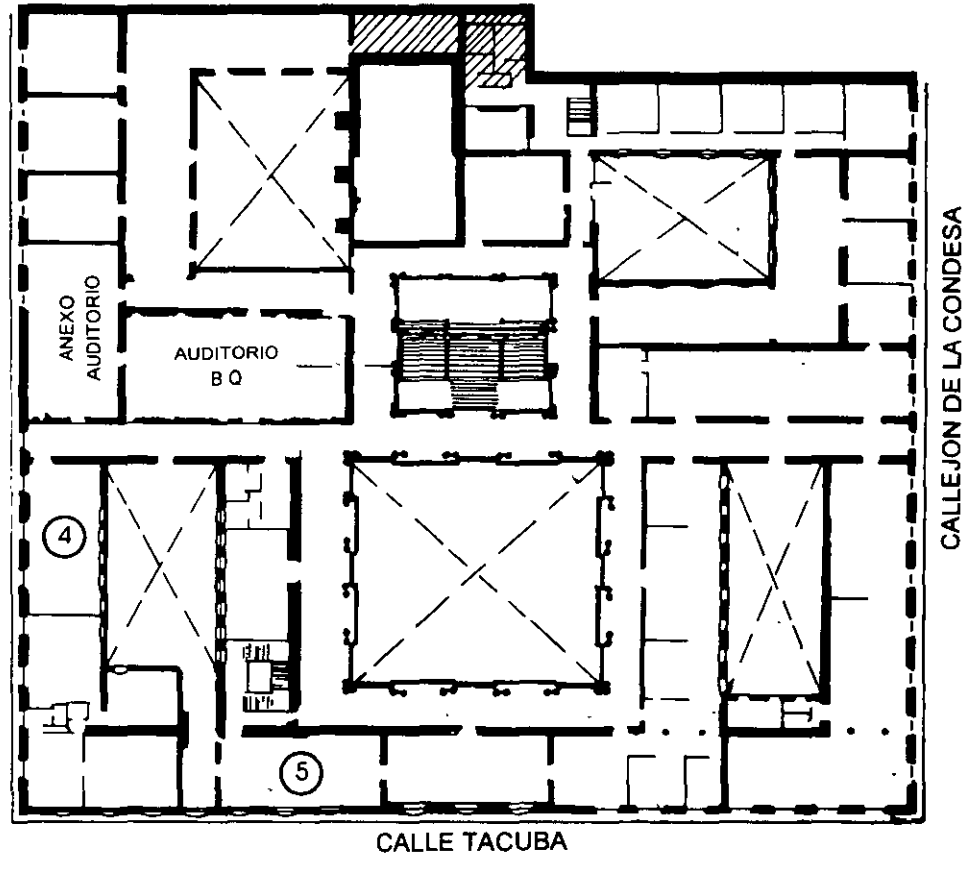
División de Educación Continua.

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

PALACIO DE MINERIA

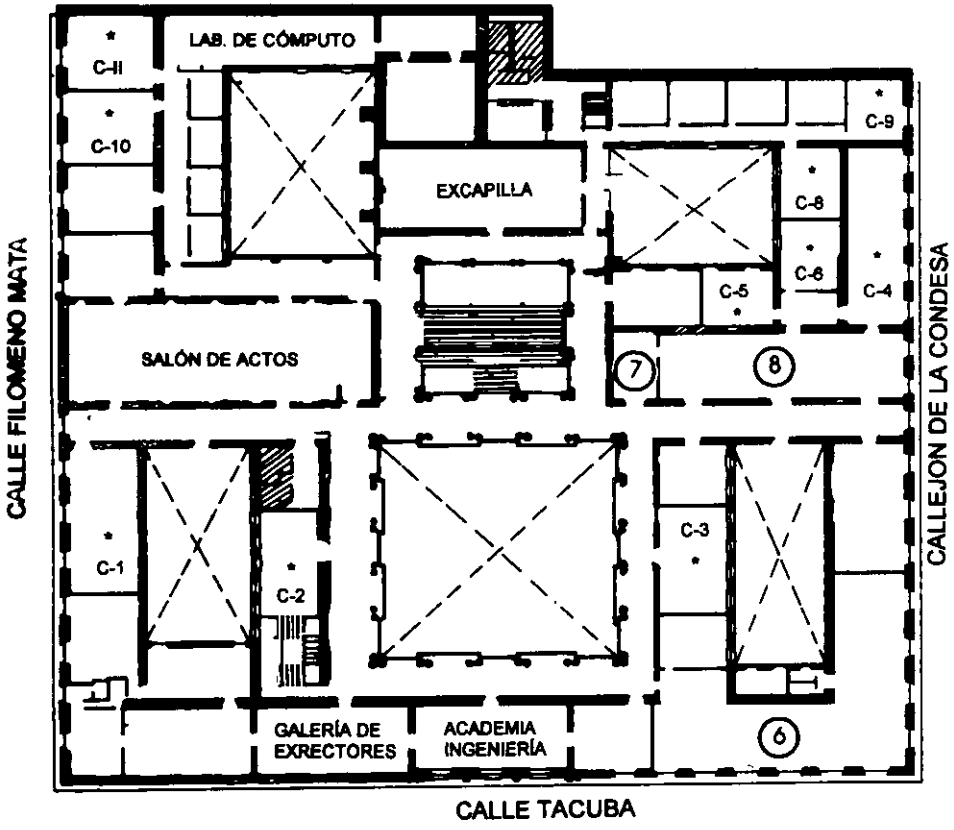


PLANTA BAJA



MEZZANINNE

PALACIO DE MINERIA



1er. PISO

GUÍA DE LOCALIZACIÓN

1. ACCESO
2. BIBLIOTECA HISTÓRICA
3. LIBRERÍA UNAM
4. CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN "ING. BRUNO MASCANZONI"
5. PROGRAMA DE APOYO A LA TITULACIÓN
6. OFICINAS GENERALES
7. ENTREGA DE MATERIAL Y CONTROL DE ASISTENCIA
8. SALA DE DESCANSO

SANITARIOS

* AULAS

DIVISIÓN DE EDUCACIÓN CONTINUA
FACULTAD DE INGENIERÍA U.N.A.M.
CURSOS ABIERTOS

DIVISIÓN DE EDUCACIÓN CONTINUA
DEC

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

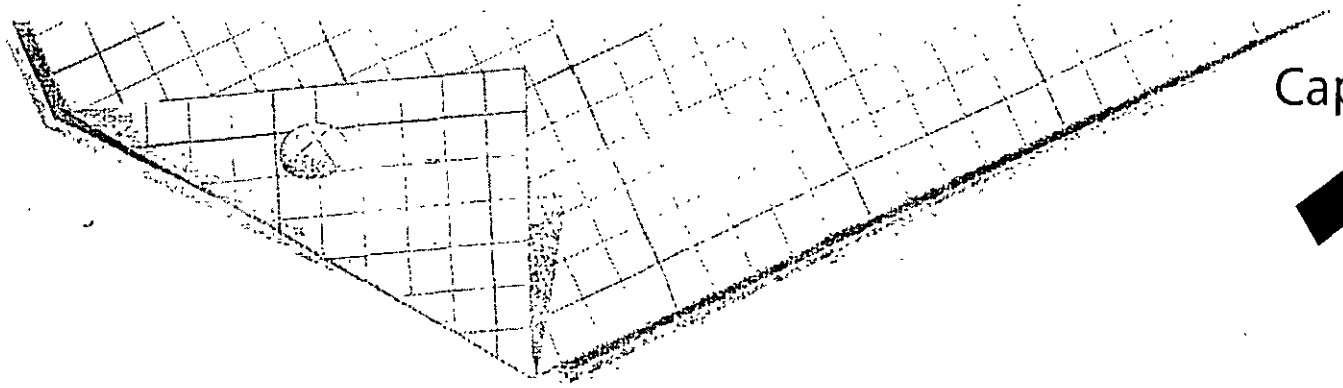
CURSOS INSTITUCIONALES

INTRODUCCIÓN A VISUAL BASIC VERSION 5.0

Del 16 al 27 de febrero de 1999.

Apuntes Generales

Ing. Víctor Caamaño Rosas
Palacio de Minería
1998.



Capítulo

1

Desarrollo del primer programa

Después de unos pocos capítulos, podrá comprobar que desarrollar potentes programas para Microsoft Windows utilizando Microsoft Visual Basic es una actividad muy sencilla. La capacidad de Visual Basic de ejecutar ciertas tareas complejas descansa en ciertos aspectos claves que analizaremos detalladamente en la primera parte de este libro. Incluso aunque nunca antes haya desarrollado un programa, podrá darse cuenta que la programación utiliza ciertas técnicas y razonamientos que usted pone en práctica día tras día. En este capítulo aprenderá a poner en marcha Visual Basic y a utilizar el Sistema de Programación de Visual Basic para escribir y ejecutar sus propios programas. Se familiarizará con las opciones básicas de los menús contenidos en Visual Basic y con sus procedimientos de programación. Para ello, ejecutará un programa sencillo denominado Saludo e irá afianzando sus conocimientos durante la elaboración de un nuevo programa llamado el Siete Afortunado. También aprenderá a modificar un programa, a almacenarlo en el disco duro de su computadora y a salir de Visual Basic.

EL ENTORNO DE PROGRAMACIÓN DE VISUAL BASIC

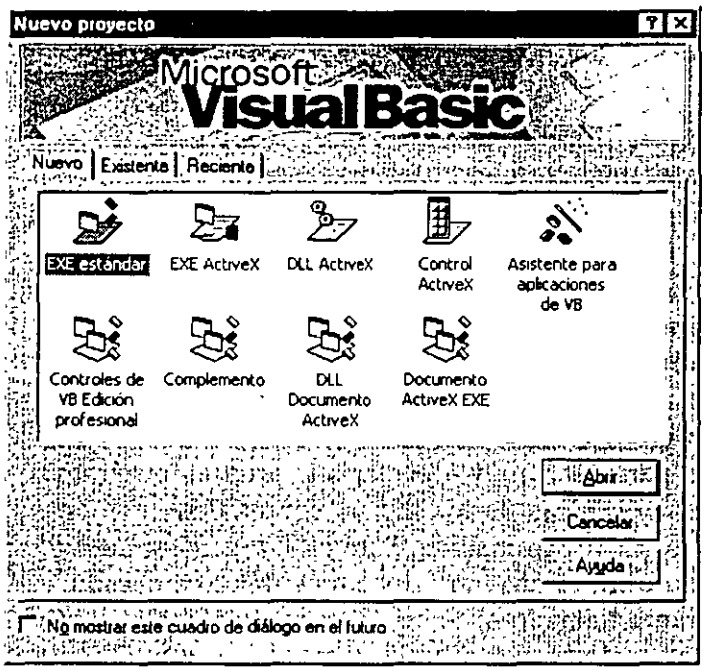
El entorno de programación de Visual Basic contiene todas las herramientas que usted necesita para construir programas para Windows, de una manera rápida y eficiente. Siga los pasos comentados en la página siguiente para poner en marcha Visual Basic.

Nota: Si todavía no ha instalado los archivos de prácticas que vienen con este libro, consulte los apartados «Búsqueda del mejor punto de partida» e «Instalación y empleo de los archivos de prácticas» contenidos al principio del libro. Finalmente, vuelva al presente capítulo.

Puesta en marcha de Visual Basic

1. En Windows 95, pulse el botón Inicio, despliegue el menú Programas y seleccione la carpeta denominada Microsoft Visual Basic 5.0. Los iconos contenidos en esta carpeta aparecerán en una lista.
2. Pulse el icono correspondiente al programa Visual Basic 5.0.

   El cuadro de diálogo Nuevo proyecto aparecerá en su pantalla tal y como se muestra en la siguiente figura. Este cuadro de diálogo le solicita que seleccione el tipo de proyecto de programación que desea crear.

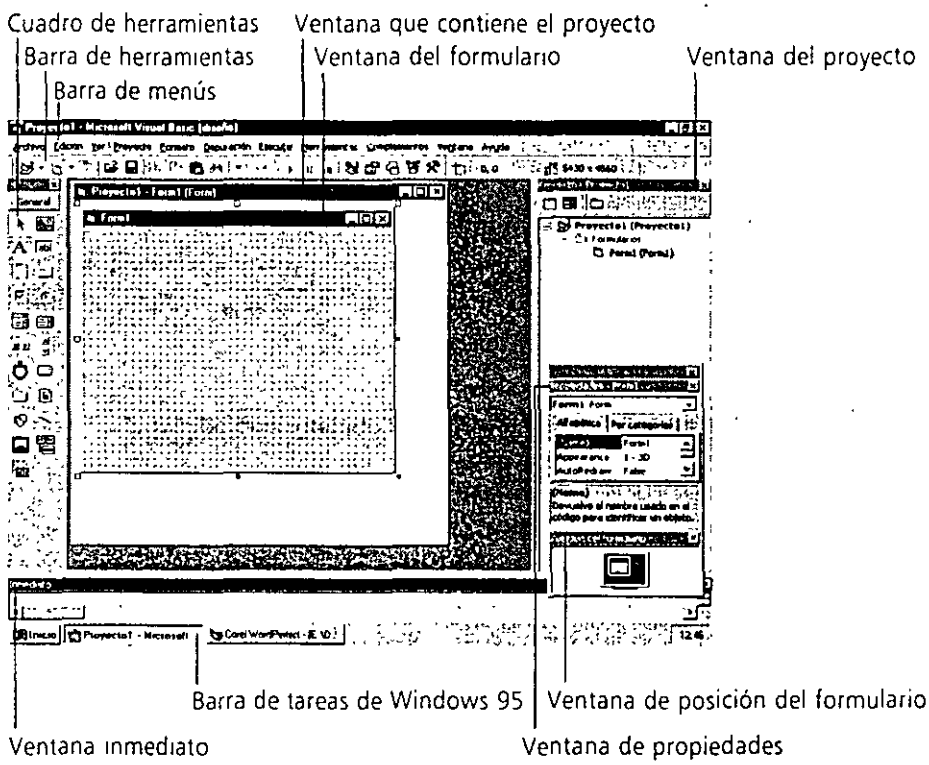


3. Pulse Abrir para aceptar el nuevo proyecto propuesto por defecto, una aplicación estándar de 32 bits para Visual Basic.

   En el entorno de programación de Visual Basic se abrirá un proyecto nuevo, junto con algunas de las ventanas y herramientas que se muestran en la figura contenida en la página siguiente.

El entorno de programación de Visual Basic contiene herramientas de programación que le ayudarán a desarrollar sus propios programas. La *barra de menús* le

permitirá acceder a la mayoría de las opciones que controlan el entorno de programación. Los menús y los mandatos trabajan según una serie de reglas estándar utilizadas en todos los programas basados en Windows; y podrá acceder a ellos utilizando el teclado o el ratón.



Para ver la función asociada a un botón de la barra de herramientas, sitúe el puntero del ratón sobre él durante unos cuantos segundos

Justo debajo de la barra de menús se encuentra la *barra de herramientas*, un conjunto de botones que funcionan como atajos para ejecutar opciones y controlar el entorno de programación de Visual Basic. Si anteriormente ha utilizado alguna vez Microsoft Excel o Microsoft Word, la barra de herramientas debe serle un elemento familiar. Para activar un botón de la barra de herramientas, deberá pulsar este icono utilizando el ratón. En la parte inferior de la pantalla se encuentra la *barra de tareas* de Windows 95. Podrá emplear esta barra para conmutar entre distintos componentes de Visual Basic y para activar otros programas de Windows.

En esta pantalla inicial también se encuentran disponibles el cuadro de herramientas, la barra de menús, la ventana que contiene al proyecto, la ventana del formulario, la ventana del proyecto, la ventana inmediata, la ventana de propiedades y la ventana de posición del formulario. El tamaño exacto y la forma de estas ventanas dependerá de la configuración que tenga su sistema. En Visual Basic 5 podrá alinear y fijar ventanas para que todos los elementos del entorno de programación permanezcan visibles y sean fácilmente accesibles. Más adelante, en este mismo capítulo, le mostraré cómo puede adaptar a sus propias necesidades el entorno de programación.

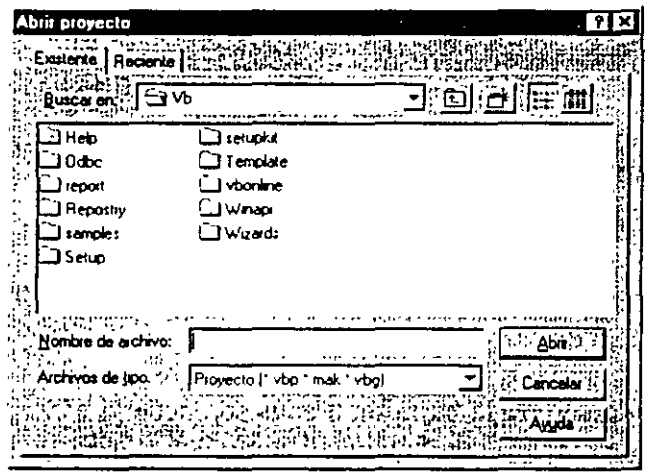
En el siguiente ejercicio practicaremos con la barra de menús y con la barra de herramientas para cargar y ejecutar un programa ejemplo desarrollado en Visual Basic denominado Saludo.

Empleo de la barra de menús para abrir un proyecto de programación previamente existente

1. Despliegue el menú Archivo utilizando el ratón.
   En su pantalla aparecerán las opciones contenidas en el menú Archivo.

Los archivos de proyecto de Visual Basic se distinguen por tener las extensiones .vbp, .mak o .vbg.

2. En el menú Archivo, seleccione la opción Abrir proyecto.
   En su pantalla aparecerá el cuadro de diálogo denominado Abrir proyecto. Este cuadro de diálogo le permitirá abrir cualquier programa desarrollado en Visual Basic existente en su disco duro, unidad de red, CD-ROM o disquete:



Botón Subir un nivel

3. Pulse tres veces el botón Subir un nivel, realice una doble pulsación sobre la carpeta Vb5Sbs contenida en el directorio raíz y, finalmente, realice una doble pulsación sobre la carpeta Less01.
   La carpeta \Vb5Sbs (así se denominará por defecto la carpeta creada por el programa de instalación de los archivos de prácticas contenidos en el CD-ROM de este libro) incluye todos los archivos de ejemplo y prácticas mencionados en el presente libro. Por ello, deberá utilizar la carpeta denominada Less*xx* cuando esté leyendo en el libro los conceptos contenidos en el capítulo mencionado.
4. Seleccione el proyecto Saludo y, posteriormente, pulse Abrir.
   El archivo de proyecto Saludo cargará el formulario de la interfaz de usuario, sus propiedades, el código asociado con el programa y el módulo estándar del programa Saludo.

5. Si en su pantalla no se muestra el formulario Saludo, deberá realizar una doble pulsación sobre la carpeta Formularios contenida en la ventana Proyecto; finalmente, deberá pulsar Form1 (Saludo.frm).

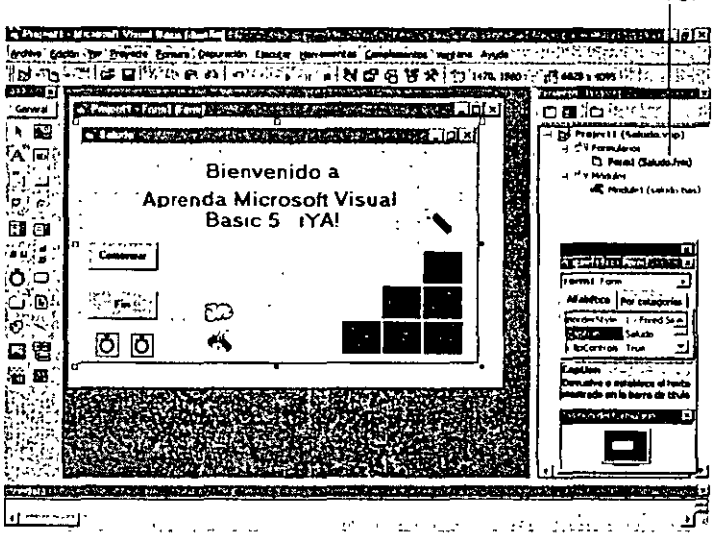
   Antes de que pueda trabajar con un componente de cualquier proyecto, deberá seleccionarlo en la ventana Proyecto.

Botón Ver objeto

6. Pulse el botón Ver objeto contenido en la ventana del Proyecto para contemplar el aspecto que tiene la interfaz del usuario del programa.

   En su pantalla aparecerá el formulario del programa, tal y como se muestra en la figura siguiente:

Formulario actual del programa



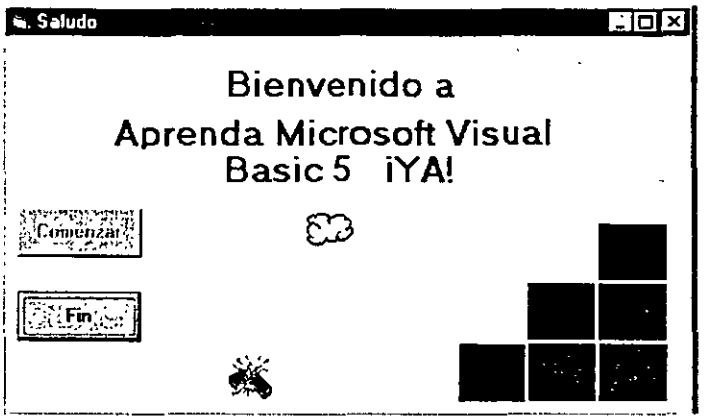
Si no ve la ventana contenedora del proyecto alrededor del formulario, el motivo será que dicha ventana se encuentra maximizada y tendrá que pulsar el botón Restaurar ventana situado en la barra de herramientas para ver el proyecto tal y como se muestra en la figura anterior. Saludo es, simplemente, un programa ejemplo desarrollado en Visual Basic pensado para que se familiarice con este entorno de programación. Como Saludo contiene varios de los elementos básicos que podrá encontrar con frecuencia en cualquier programa desarrollado con Visual Basic, podrá utilizarlo para explorar superficialmente el entorno de programación. Cuando ejecute Saludo verá por pantalla un mensaje de bienvenida y algunos efectos de animación.

Botón Iniciar

7. Pulse el botón Iniciar contenido en la barra de herramientas de Visual Basic para poner en marcha el programa Saludo.

   A partir de ese instante, el cuadro de herramientas y la ventana Propiedades desaparecerán de la pantalla, a la vez que el programa Saludo comienza a ejecutarse.

8. Pulse el botón Comenzar para ver algún ejemplo de animación. Su pantalla tendrá el aspecto mostrado en la figura siguiente.



9. Pulse el botón Fin para finalizar el programa y volver al entorno de programación. Esto es todo por el momento. ¡Acaba de ejecutar su primer programa en Visual Basic!

Las herramientas de desplazamiento, fijación y modificación de tamaño

Debido a que contiene siete herramientas de programación en una única pantalla, el entorno de desarrollo de Visual Basic puede convertirse en un lugar demasiado poblado. Visual Basic 5 cuenta con la capacidad de mover, fijar y modificar el tamaño de cada una de estas herramientas de programación. Gracias a ello podrá tener un control completo sobre la forma y el tamaño de los elementos presentes en el entorno de desarrollo.

En Visual Basic 5 podrá utilizar una nueva función denominada «fijar o acoplar» para organizar sus herramientas de programación.

Para mover una ventana, el cuadro de herramientas o la barra de herramientas, sólo tendrá que pulsar con el ratón sobre la barra de títulos y arrastrar el objeto hacia su nueva ubicación. Si alinea una ventana con el borde de otra ventana, ambas se acoplarán o fijarán. Las ventanas de este tipo suponen una ventaja porque siempre estarán visibles (no permanecerán ocultas detrás de otras ventanas).

Si desea ver una parte mayor de la ventana fijada, sólo tendrá que arrastrar uno de sus bordes para ver más contenido. Si está cansado de las ventanas acopladas y desea que sus herramientas se vuelvan a superponer entre sí como ocurría en las versiones anteriores de Visual Basic, seleccione el mandato Opciones contenido en el menú Herramientas, pulse la etiqueta denominada Acople y elimine las marcas de verificación correspondientes a aquellas herramientas que desee visualizar de forma independiente.

A medida que avance en la lectura de los siguientes apartados irá practicando con el desplazamiento, acople y la modificación del tamaño de las diferentes herramientas contenidas en el entorno de programación de Visual Basic hasta que, finalmente, se sienta a gusto con ellas.

El formulario de la interfaz de usuario

Cada formulario es una ventana en su interfaz de usuario.

En Visual Basic, un *formulario* es una ventana que podrá adaptar a sus gustos o necesidades personales para crear la interfaz de usuario de su programa. En el programa que acabamos de ejecutar, el formulario es la ventana que se mostró en pantalla durante la ejecución del programa. Un formulario puede contener menús, botones, cuadros de lista, barras de desplazamiento y cualquier otro elemento que se pueden contemplar en los típicos programas desarrollados para Windows. Cuando ponga en marcha el Entorno de Programación de Visual Basic aparecerá sobre una rejilla estándar (una «rejilla» es un grupo de puntos espaciados de forma regular) un formulario por defecto denominado Form1. Podrá utilizarlo para crear la interfaz de usuario de su programa y para alinear los elementos presentes en la interfaz. Podrá ajustar el tamaño del formulario utilizando el ratón; el formulario puede ocupar toda o parte de la pantalla. Podrá añadir nuevos formularios ejecutando el mandato Agregar formulario contenido en el menú Proyecto.

Si parte del formulario se encuentra cubierto por las herramientas de programación, podrá cerrar o modificar el tamaño de éstas para que ocupen menos espacio, o podrá pulsar sobre la barra de títulos del formulario y arrastrar el formulario hasta que pueda verlo por completo. A diferencia de lo que ocurría en las versiones anteriores de Visual Basic, en Visual Basic 5 mover el formulario por la pantalla no afectará al lugar donde posteriormente se mostrará el formulario cuando ejecute el programa. Esta característica del programa se controlará desde la ventana denominada Posición del formulario. Si desea definir la posición inicial que ocupará el nuevo formulario, sólo tendrá que arrastrar con el ratón el pequeño formulario que se muestra en la ventana de Posición del formulario hasta que ocupe la posición que usted desee.

Cuadro de herramientas

Botón Cuadro de herramientas

Podrá desplazar el cuadro de herramientas a cualquier lugar de la pantalla sin más que situar el puntero del ratón sobre su barra de títulos, pulsar el botón izquierdo de este dispositivo y arrastrar el cuadro de herramientas al lugar deseado.

Podrá añadir los elementos deseados a la interfaz de usuario de su programa utilizando las herramientas, o *controles*, contenidas en el cuadro de herramientas. Para abrir el cuadro de herramientas deberá pulsar el botón Cuadro de herramientas situado en la barra de herramientas. El cuadro de herramientas suele estar localizado junto al borde izquierdo de la pantalla. Una vez que los controles han sido introducidos en el formulario se convierten en *objetos*, o elementos programables de la interfaz de usuario, del programa. El cuadro de herramientas contiene controles con los que podrá añadir en la interfaz del usuario dibujos artísticos, etiquetas, botones, cuadros de lista, barras de desplazamiento, rejillas, menús y formas geométricas. Estos elementos serán visibles para el usuario cuando éste ejecute su programa y funcionarán en la misma forma en que lo hacen los demás elementos que aparecen en las típicas aplicaciones desarrolladas para Windows.

El cuadro de herramientas contiene también controles que podrá utilizar para crear objetos que lleven a cabo operaciones especiales «detrás del telón». Estos potentes objetos realizarán las tareas que usted desee pero no se mostrarán al usua-

rio cuando éste ejecute el programa. Entre este tipo de objetos se incluyen aquellos que sirven para manipular información en las bases de datos, permiten trabajar con aplicaciones basadas en Windows o llevan un control del tiempo transcurrido durante la ejecución de sus programas.

Podrá mostrar el nombre asociado a cualquier control contenido en el cuadro de herramientas situando durante unos instantes el puntero del ratón sobre dicho elemento. Más adelante, en este mismo capítulo, le mostraré cómo hacer uso de estos controles.

La ventana de propiedades

La ventana de propiedades le permitirá modificar las características, o *propiedades de configuración,* asociadas con cada uno de los elementos presentes en la interfaz de usuario. Una propiedad es una cualidad de uno de los objetos de la interfaz de usuario. Por ejemplo, podrá modificar el mensaje de bienvenida mostrado por el programa Salida para que aparezca en otro tipo o tamaño de letra o con una alineación diferente (con Visual Basic podrá mostrar el texto en cualquier tipo de letra que tenga instalada en su computadora, al igual que sucede con Excel o con Word). Podrá modificar las propiedades asociadas utilizando la ventana Propiedades durante la creación de la interfaz de usuario o introduciendo nuevo código durante la ejecución del programa.

La ventana de Propiedades contiene un cuadro de lista desplegable que muestra todos los objetos o elementos disponibles para las interfaces de usuario; la ventana de Propiedades también lista las propiedades que podrá modificar para cada objeto. Practicaremos ahora modificando la propiedad Caption (rótulo) asociada al botón Fin mostrado en el programa Saludo.

Modificación de una propiedad

1. Compruebe que el programa Saludo no se está ejecutando (verá la palabra *diseño* en la barra de títulos cuando el programa no se esté ejecutando) y pulse el objeto Fin contenido en el formulario.

   El objeto Fin (un botón de orden) estará ahora rodeado por pequeños cuadrados de color oscuro que indicarán que este objeto se encuentra *seleccionado.* Para poder trabajar con un objeto contenido en un formulario de Visual Basic deberá seleccionarlo primero.

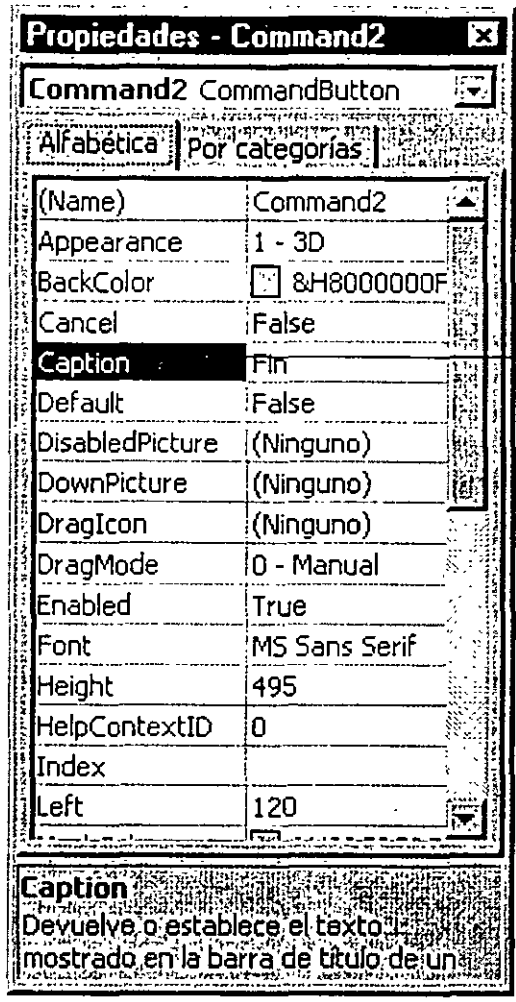
   Pulse el botón Ventana Propiedades contenido en la barra de herramientas.

Botón Ventana Propiedades

   En el entorno de programación se resaltará la ventana Propiedades (si la ventana de propiedades no se encontraba abierta, al pulsar el botón anterior esta ventana aparecerá en su pantalla).

2. Realice una doble pulsación sobre la barra de títulos de la ventana de Propiedades para mostrarla como una ventana flotante (no acoplada).

   Verá una ventana similar a la que se muestra en la figura siguiente:



La ventana de Propiedades lista todos los parámetros de configuración asociados con el segundo botón de orden contenido en el formulario (en total, los botones de órdenes tienen asociadas 32 propiedades). Los nombres de las propiedades del botón de orden se listan en la columna izquierda de la ventana, mientras que el valor asignado a cada una de estas propiedades se muestra en la columna de la derecha. En la etiqueta Alfabética se listan las propiedades en orden alfabético.

3. Desplace la lista de propiedades hasta que se haga visible la propiedad Caption (rótulo).

   Podrá desplazar el contenido de la ventana de Propiedades en la misma forma que lo hace en cualquier otro cuadro de lista.

4. Realice una doble pulsación sobre la propiedad Caption (busque el nombre de la propiedad en la columna de la izquierda).

   El actual valor para Caption se encuentra resaltado en la columna de la derecha («Fin») y un cursor parpadea a su derecha.

5. Pulse la tecla SUPR, escriba **Terminar** y pulse INTRO.

   El valor de la propiedad Caption habrá variado de «Fin» a «Terminar». El rótulo que se muestre ahora en el formulario será distinto, y en la siguiente ocasión en que ejecute el programa, el botón de orden encerrará la palabra Terminar en su interior.

6. Retorne la ventana de Propiedades a su posición acoplada por debajo de la ventana de Proyecto.

   Es un buen momento para que practique con la tarea de fijar o acoplar ventanas. Realizar una doble pulsación sobre la barra de títulos es el método más rápido. También podrá realizar esta operación arrastrando la ventana de Propiedades hasta situar su barra de títulos justo debajo del borde inferior de la ventana de Proyecto. Como existen tantas ventanas próximas, el «acoplamiento manual» exige cierta práctica y puede resultar bastante frustrante en un principio. Pero cuando utilice más adelante estas herramientas de programación, podrá beneficiarse si su espacio de trabajo se encuentra bien organizado.

Discusión sobre propiedades

En Visual Basic cada uno de los elementos de la interfaz de usuario de un programa (incluyendo el propio formulario) cuenta con un conjunto de propiedades redefinibles. Podrá definir las propiedades durante el proceso de diseño utilizando la ventana de Propiedades. Además, si introduce propiedades en el código podrá modificar el valor asociado a dichas propiedades durante la ejecución del programa (los elementos de la interfaz de usuario que reciben datos por parte del usuario suelen utilizar propiedades para transmitir la información). Sin embargo, en un principio, las propiedades pueden ser un concepto difícil de digerir. Expresar esta idea en términos de algo familiar en nuestra vida diaria puede ayudar a comprenderlo mejor.

Piense en la siguiente analogía con una bicicleta: Una bicicleta es un objeto que podrá utilizar para desplazarse con cierta rapidez de un lugar a otro. Como la bicicleta es un objeto físico, tiene varias características que le son propias. Tiene una marca, color, ruedas, frenos, volante, etc., y habrá sido construida siguiendo un estilo determinado (puede ser una bicicleta de paseo, de montaña o un tándem). Siguiendo la terminología de Visual Basic, estas características son las *propiedades* del objeto bicicleta. El molde con el que se creó el modelo de la bicicleta se puede denominar control «bicicleta». La mayor parte de las características de la bicicleta se definirán en el momento de la fabricación de ésta, pero algunas propiedades especiales (tales como neumáticos, velocidad y edad) se irán modificando a medida que se le dé uso a la bicicleta. A medida que trabaje con Visual Basic se irá encontrando con propiedades de los dos tipos.

La ventana de Proyecto

Cualquier programa desarrollado en Visual Basic está formado por varios archivos enlazados entre sí o *compilados*. Para facilitarle la tarea de pasar de uno a otro componente cuando esté trabajando en un proyecto, los diseñadores de Visual Basic han incluido una *ventana de Proyecto* en el entorno de programación. La ventana Proyecto lista todos los archivos utilizados en el proceso de programación y le permitirá acceder a su contenido utilizando dos botones especiales: Ver Código y Ver Objeto. Podrá añadir, eliminar o guardar archivos individuales de un proyecto utilizando las opciones contenidas en los menús Archivo y Proyecto. Si se efectúa algún cambio en un proyecto, éste se reflejará en la ventana Proyecto.

El archivo que almacena la información relativa a todos los demás elementos que forman parte de un proyecto se denomina *archivo de proyecto de Visual Basic* y llevará la extensión *.vbp*. En Visual Basic 5 se podrá cargar simultáneamente en la ventana de Proyecto más de un archivo de proyecto, y podrá pasar de uno a otro sin más que pulsar sobre el nombre del proyecto. Debajo del nombre del proyecto, la ventana Proyecto muestra los componentes que existen en cada proyecto en una estructura en árbol, de forma similar a las vistas presentadas por el Explorador de Windows. Si pulsa con el ratón los signos más y menos situados a la izquierda de las carpetas; podrá expandir y contraer estas «ramas», incluyendo Formularios, Módulos y otras categorías.

En el siguiente ejercicio se mostrará la ventana Proyecto correspondiente al programa Saludo.

Cómo visualizar la ventana Proyecto

Botón Explorador de Proyecto

1. Pulse el botón Explorador de Proyecto situado en la barra de herramientas.

   En el entorno de programación se resaltará la ventana del Proyecto (si la ventana no estaba abierta, aparecerá ahora).

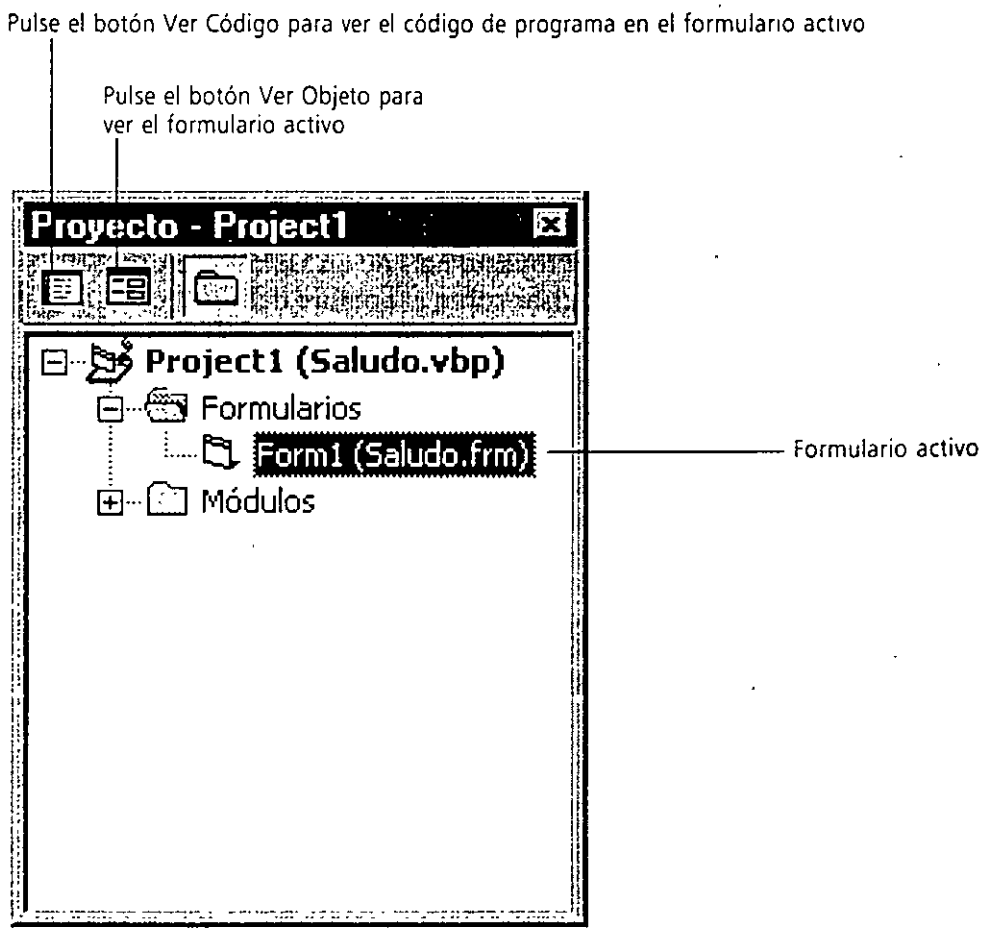
2. Realice una doble pulsación sobre la barra de títulos de la ventana del Proyecto para mostrarla como una ventana flotante (no fijada).

3. Pulse el signo más situado a la izquierda de las carpetas Formularios y Módulos (si no lo ha hecho anteriormente) para ver todos los componentes del proyecto.

   El archivo de proyecto en este caso tiene el nombre de Saludo. En el proyecto Saludo se listan los archivos Saludo.frm y Saludo.bas. Saludo.frm contiene el formulario correspondiente a la interfaz de usuario y todo el código asociado con los objetos que forman parte de este formulario.

Saludo.bas contiene el código compartido por todas las partes del programa. Cuando el programa se compila en un archivo ejecutable, o cuando se prepara para poder ejecutarse bajo Windows, estos archivos se combinarán para formar un único archivo .exe.

4. Realice una doble pulsación sobre la barra de títulos de la ventana Proyecto para volver a su posición de acoplado.



Obtención de ayuda

Visual Basic incluye un sistema de referencia en línea que podrá utilizar cuando así lo desee para aprender más sobre el entorno de programación, las herramientas de desarrollo y el lenguaje de programación en el Sistema de Programación de Visual Basic. Dedique unos instantes a explorar los recursos de ayuda antes de construir el primer programa.

Podrá acceder al sistema de ayuda de diversas formas.

Para obtener información	Haga lo siguiente
Por tema o actividad	En el menú de Ayuda de Visual Basic, pulse el tema denominado Temas de Ayuda de Microsoft Visual Basic.
Cuando se encuentre trabajando en una ventana o en un cuadro de diálogo	Pulse F1 o pulse el botón Ayuda contenido en el cuadro de diálogo.
Sobre una herramienta de programación, propiedad o elemento del lenguaje específicos	En el menú Ayuda, seleccione la opción Temas de Ayuda de Microsoft Visual Basic, seleccione la etiqueta Índice y en el cuadro de texto escriba la palabra sobre la que desea obtener información.
Para poder ponerse en contacto con Microsoft para obtener soporte técnico	En el menú Ayuda seleccione la opción Obtener Soporte Técnico. Si puede acceder a Internet podrá desplegar el submenú Microsoft en el Web y conectarse con uno de los servidores Web que aparecen en la lista.

Podrá utilizar los siguientes pasos para obtener ayuda sobre un tema específico en Visual Basic. Este ejercicio práctico le mostrará cómo buscar información relacionada con la ventana de Proyecto, pero podrá modificar el ejercicio a su gusto para buscar información sobre el tema que le interese.

Obtención de ayuda sobre un tema específico

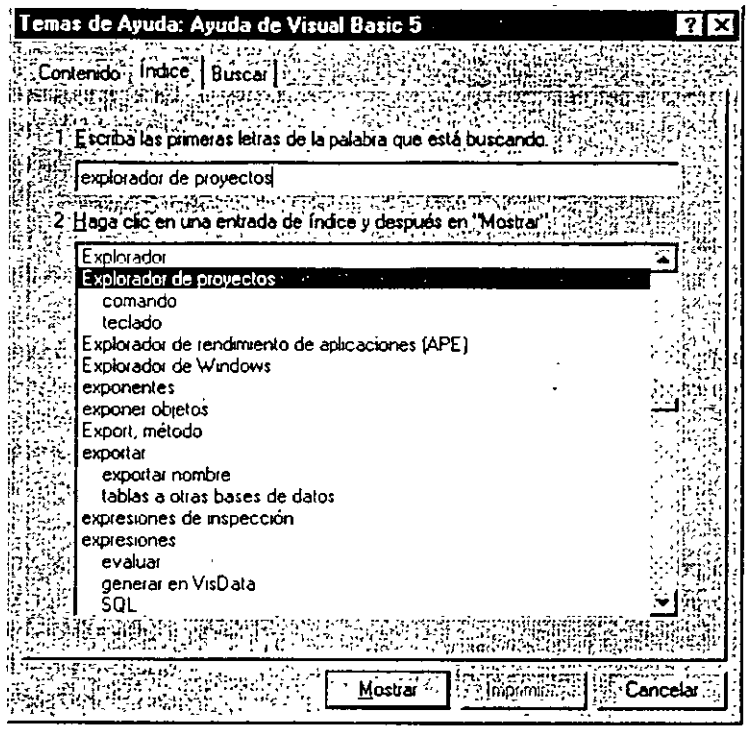
El menú Ayuda es su puerta de entrada al sistema de Ayuda de Visual Basic.

1. Pulse el menú Ayuda contenido en la barra de menús. En su pantalla aparecerá el contenido del menú Ayuda.

2. En el menú Ayuda, seleccione la opción Temas de Ayuda de Microsoft Visual Basic.

3. Seleccione la etiqueta Índice y en el cuadro de texto escriba la frase **explorador de proyectos** (o cualquier otro tema de búsqueda).

   A medida que vaya escribiendo las palabras *explorador de proyectos*, irán apareciendo en el cuadro de lista los temas de ayuda que empiecen por «e», luego por «ex», etc., hasta que se muestre un contenido similar al de la siguiente figura:



4. Realice una doble pulsación sobre el tema Explorador de Proyecto contenido en la lista.

   La ventana de Ayuda muestra información sobre la ventana de proyecto de Visual Basic. Si todo el texto mostrado no cabe en una única ventana aparecerán unas barras de desplazamiento que podrá emplear para desplazarse por el mismo.

5. Pulse Ver También, que está situado cerca de la parte superior de la ventana y está subrayado con una línea verde.

   Se abrirá una ventana denominada Temas encontrados. Realice una doble pulsación sobre los temas listados para conocer otras funciones relacionadas con la ventana del Proyecto y por las que esté interesado.

6. Pulse ESC para cerrar la ventana Temas encontrados.

7. Pulse el botón Cerrar situado en la barra de títulos de la ventana de Ayuda para salir del sistema de Ayuda.

El sistema de Ayuda de Visual Basic es un recurso de gran utilidad para conocer en mayor profundidad el entorno de programación o cualquier tema relacionado con la programación en Visual Basic. No dude en utilizarlo en cualquier momento que le surjan dudas.

SIETE AFORTUNADO: SU PRIMER PROGRAMA EN VISUAL BASIC

Ahora que ya cuenta con cierta experiencia a la hora de utilizar el entorno de programación de Visual Basic, es el momento de generar su primer programa en él. La aplicación para Windows que está a punto de desarrollar se denomina «Siete afortunado» y se trata de un juego que simula una máquina tragaperras. Este juego tiene una interfaz de usuario bastante simple y puede ser creado y compilado en sólo unos minutos utilizando Visual Basic. El aspecto del programa que va a crear será similar al mostrado en la figura siguiente (si desea ejecutar este programa antes de comenzar, podrá encontrarlo en la carpeta denominada \Vb5Sbs\Less01 contenida en su disco duro). Así verá su programa cuando esté terminado:

Nota: Si desea obtener más información sobre qué es un programa y cómo se crean los programas en Visual Basic, consulte el apéndice denominado «Cómo pensar como un programador».

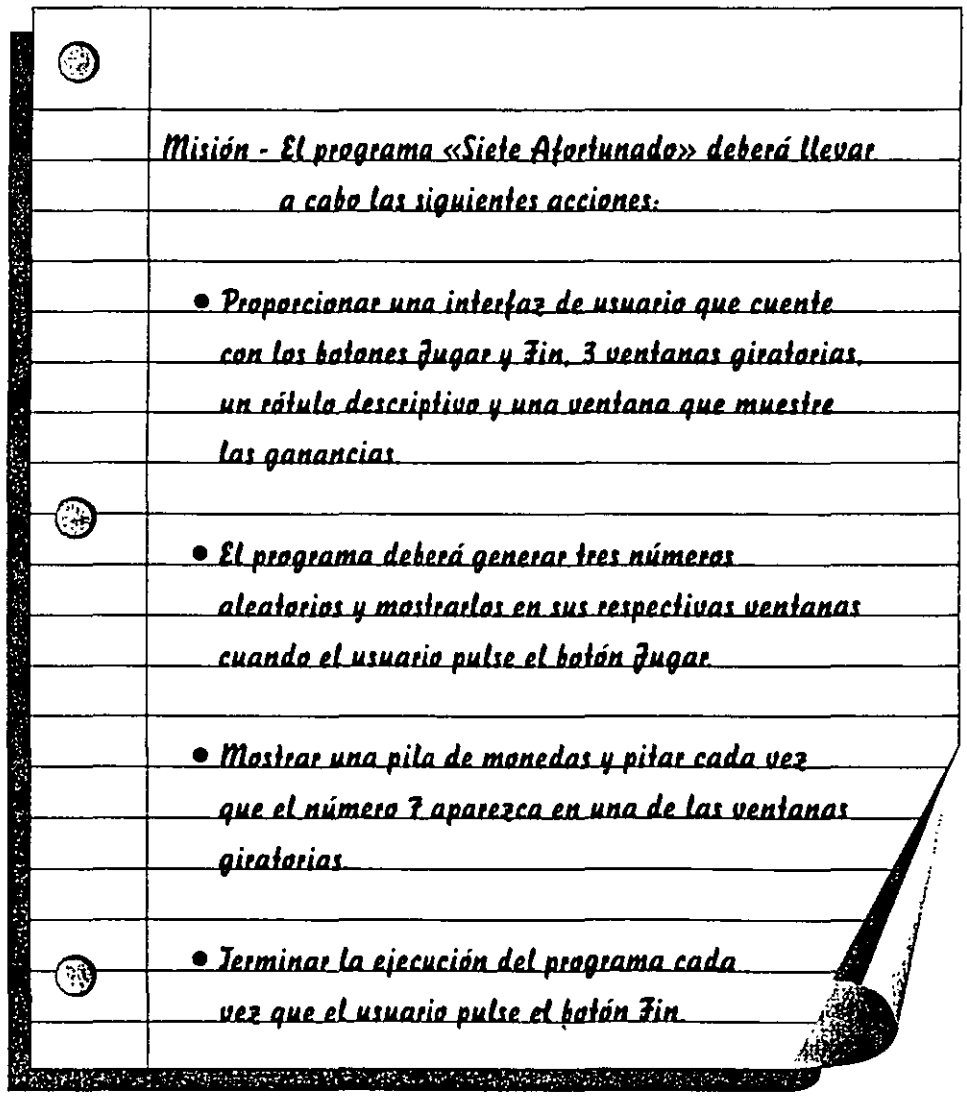
Pasos de programación

La interfaz de usuario de «Siete afortunado» contiene dos botones de orden, tres ventanas de números, un gráfico que muestra una pila de monedas y el rótulo Siete Afortunado. Estos elementos han sido introducidos en el programa creando siete objetos dentro del formulario Siete Afortunado y, finalmente, modificando ciertas propiedades asociadas a cada uno de estos objetos. Una vez que la interfaz ha sido diseñada, se añadió el código asociado a los botones Jugar y Fin para hacer que cada vez que se pulsara el botón adecuado la computadora mostrara tres números aleatorios. Para generar el programa Siete Afortunado deberá seguir los tres pasos comentados en el Apéndice «Cómo pensar como un programador»: crear la interfaz de usuario, defi-

nir las propiedades y añadir el código del programa. El proceso seguido para crear el programa Siete Afortunado se muestra esquemáticamente en la siguiente tabla:

Pasos de programación	Número de elementos
1. Crear la interfaz de usuario	7 objetos
2. Definir las propiedades	10 propiedades
3. Escribir el código del programa	2 objetos

Otra forma de concebir este programa es utilizando el siguiente algoritmo o lista que contiene los pasos de programación. Crear un algoritmo puede ser un útil punto de partida siempre que desee desarrollar un programa.



Creación de la interfaz de usuario

Comenzaremos a construir el programa creando un nuevo proyecto y generando la interfaz de usuario utilizando los controles contenidos en el cuadro de herramientas.

Cree la interfaz de usuario

Siempre que desee generar un nuevo proyecto deberá escoger la opción Nuevo Proyecto contenida en el menú Archivo.

1. Despliegue el menú Archivo y seleccione la opción Nuevo Proyecto.

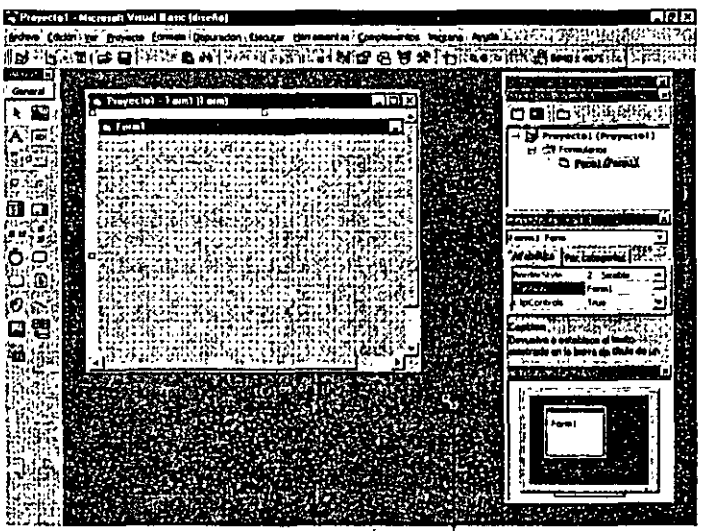
   Pulse el botón No en el caso de que Visual Basic le pregunte si desea almacenar los cambios realizados en el programa Saludo. Al hacerlo, el programa Saludo será eliminado de la memoria del ordenador.

2. Pulse el botón Aceptar para crear una aplicación estándar de 32 bits con Visual Basic.

   Visual Basic mostrará un formulario vacío en el centro de la pantalla que le servirá para construir su interfaz de usuario. En primer lugar, aumentaremos el tamaño del formulario y crearemos los dos botones de órdenes de esta interfaz:

3. Sitúe el puntero del ratón sobre la esquina inferior derecha de la ventana del Formulario (no de la ventana que contiene al Proyecto) hasta que el aspecto del puntero cambie y se convierta en una doble flecha. A partir de este momento, aumente el tamaño del formulario para tener más espacio para los objetos que desea introducir en el programa.

   Cuando modifique el tamaño del formulario aparecerán barras de desplazamiento en la ventana del Proyecto, tal y como se muestra en la siguiente figura:



Arrastre para aumentar el tamaño del formulario

Para ver el formulario completo sin interferencias, aumente el tamaño de la ventana que contiene el proyecto para eliminar las barras de desplazamiento y desplace o cierre las ventanas de Propiedades, de Proyecto y la de Posición del Formulario. En el siguiente paso creará un botón de orden y lo introducirá en el formulario.

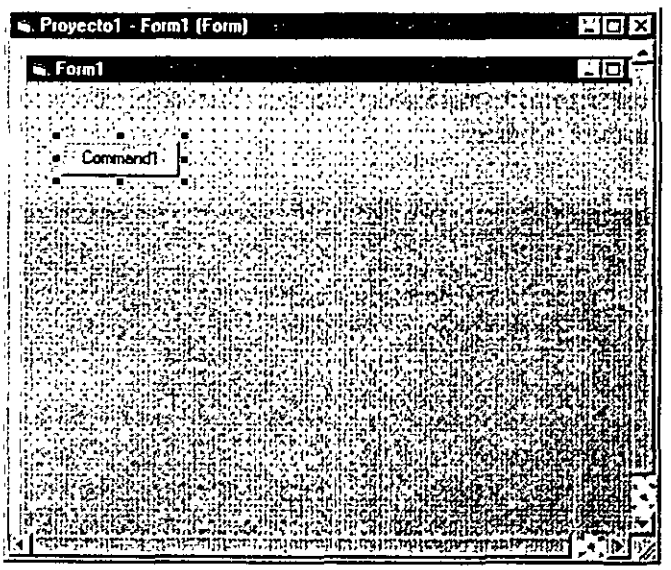
Control CommandButton

4. Pulse el control CommandButton contenido en el cuadro de herramientas y, posteriormente, sitúe el puntero del ratón sobre el formulario.

   Se seleccionará el control CommandButton y el puntero del ratón se convertirá en una cruz cuando se encuentre situado sobre el formulario. La cruz tiene como objetivo ayudar al programador en la generación de la forma rectangular de los botones de órdenes. Si pulsa el botón izquierdo del ratón y lo mantiene pulsado mientras arrastra a este dispositivo sobre la mesa, el objeto botón de orden tomará forma en la pantalla e irá adoptando un tamaño que se ajustará a los puntos contenidos en la rejilla que muestra el formulario. Intente crear ahora su primer botón de orden.

5. Desplace el puntero del ratón a una zona cercana a la esquina superior izquierda del formulario, pulse y mantenga pulsado el botón izquierdo del ratón mientras arrastra el puntero hacia la derecha y hacia abajo. Suelte el botón izquierdo del ratón cuando en su pantalla se muestre un botón de orden similar al contenido en la siguiente figura:

El nombre contenido en el botón de orden será Command1.



En el formulario aparecerá un botón de orden que se encontrará seleccionado. El nombre asignado a dicho botón será Command1, siendo el primer botón de orden del programa que está generando (puede que sea conveniente que recuerde el nombre asignado a este botón, lo necesitará más tarde cuando comience a escribir el código del programa).

Podrá mover los botones de orden utilizando el ratón y modificar su tamaño pinchando con el ratón los manejadores de selección, siempre y cuando Visual Basic se encuentre en *modo diseño* (se estará en este modo cuando el entorno de programación de Visual Basic esté activado). Sin embargo, cuando se esté ejecutando un programa el usuario no podrá mover los elementos de la interfaz a menos que se haya modificado una propiedad especial en el programa para permitir este tipo de acciones. Practique ahora moviendo y modificando el tamaño del botón de orden que acaba de crear.

Cómo mover y modificar el tamaño de un botón de orden

La rejilla le facilitará el diseño de la interfaz de usuario.

1. Arrastre el botón de orden hacia la derecha utilizando el ratón.

   El botón de orden se ajustará a la rejilla cuando suelte el botón del ratón. El objetivo de la rejilla del formulario es ayudarle a editar y alinear los diferentes elementos que aparezcan en la interfaz del usuario. Podrá modificar el tamaño de la rejilla utilizando el mandato Opciones contenido en el menú Herramientas y pulsando la etiqueta General.

2. Sitúe el puntero del ratón sobre la esquina inferior derecha del botón de orden.

   El puntero del ratón adoptará la forma de una doble flecha cuando se encuentre situado sobre una esquina o sobre un lado del objeto seleccionado. Podrá utilizar este puntero para modificar el tamaño del objeto seleccionado.

3. Aumente el tamaño del objeto pulsando y manteniendo pulsado el botón izquierdo del ratón y arrastrando el puntero hacia la derecha y hacia abajo.

   Cuando suelte el botón izquierdo del ratón, el botón de orden cambiará de tamaño y se ajustará a la rejilla.

4. Utilice el puntero del ratón en forma de doble flecha para restaurar el tamaño original del botón de orden; finalmente, vuelva a situar a este botón en su posición original.

A continuación, añadiremos un segundo botón de orden en el formulario, justo debajo del primero.

Cómo añadir un segundo botón de orden

Control CommandButton

1. Pulse el control denominado CommandButton contenido en el cuadro de herramientas.
2. Introduzca un segundo botón de orden justo debajo del primero (para obtener un aspecto pulcro y cuidado, dele al segundo botón el mismo tamaño que al primero).

Podrá borrar un objeto seleccionándolo con el ratón y pulsando la tecla SUPR.

3. Si fuera necesario, mueva o modifique el tamaño del nuevo botón hasta que consiga el efecto deseado. Si comete algún error, no dude en borrar el botón de orden y vuelva a comenzar esta operación.

Cómo añadir las etiquetas de número

Ahora añadiremos las etiquetas utilizadas para mostrar los números. Cuando el usuario pulse el botón Jugar deberán aparecer tres números aleatorios en sendos cuadros de etiqueta. Si uno de esos números resulta ser un siete, el usuario obtendrá un premio.

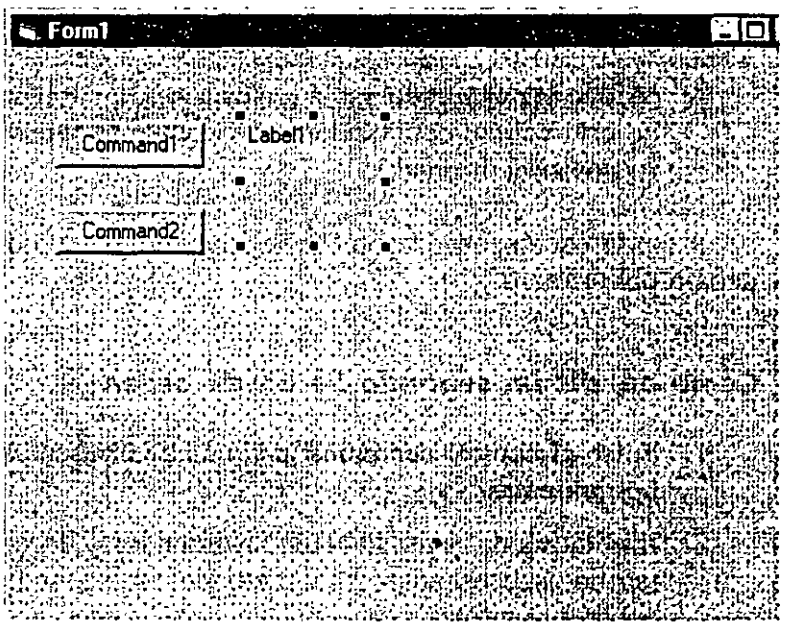
Control Label

1. Pulse el control denominado Label contenido en el cuadro de herramientas y, posteriormente, sitúe el puntero del ratón sobre el formulario.

   El control Label permanecerá seleccionado y el puntero del ratón adoptará la forma de una cruz cuando se sitúe sobre el formulario.

2. Cree una pequeña caja rectangular a la derecha de los botones de orden, similar a la que se muestra en la figura siguiente.

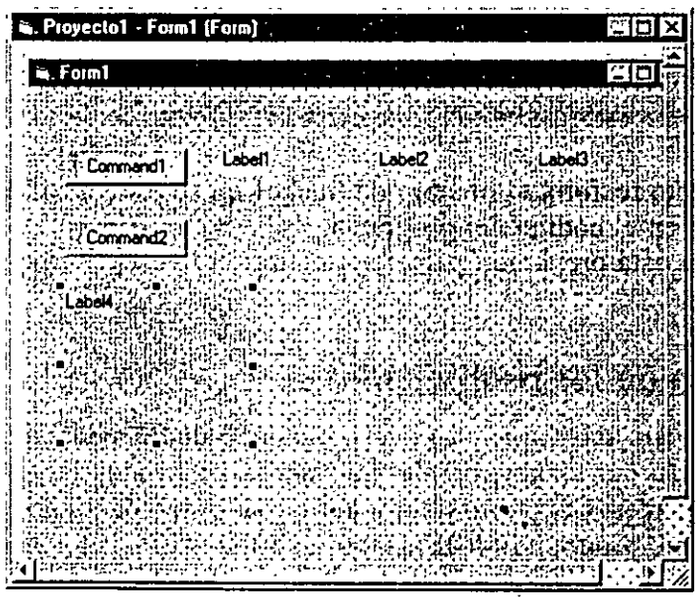
   El objeto etiqueta que acaba de crear llevará por nombre Label1 y es la primera etiqueta del programa. Una etiqueta es un elemento especial de la interfaz de usuario diseñada para mostrar texto, números o símbolos cuando se ejecute el programa. A continuación, creará en el formulario dos etiquetas más, denominadas Label2 y Label3.



Nota: Cuando añada etiquetas en el presente ejercicio observe los cuadros que aparecen cerca de las etiquetas cuando las dibuja. Estos cuadros, que contienen medidas verticales y horizontales, reciben el nombre de cuadros de tamaño. *El número de la izquierda proporciona la localización de la esquina superior izquierda del objeto seleccionado, mientras que el número de la derecha proporciona las dimensiones del objeto. Estos números están dados en una unidad de medida denominada «twips»; un twip es la vigésima parte de un punto (un punto equivale a 1/72 pulgadas, por lo que un twip es igual a 1/1440 pulgadas; recuerde que una pulgada equivale, aproximadamente, a 2,5 centímetros). Podrá utilizar los cuadros de tamaño para situar con exactitud los objetos en el formulario. De esta forma podrá comparar los tamaños relativos de los objetos que vaya creando.*

3. Pulse el control Label y dibuje un cuadro de etiqueta a la derecha del primero.
   Cree una etiqueta que tenga el mismo tamaño que la primera. El rótulo «Label2» aparecerá en la etiqueta.
4. Pulse de nuevo el control Label y añada una tercera etiqueta en el formulario, a la derecha de la que acaba de introducir.
   El rótulo «Label3» aparecerá dentro del cuadro. A continuación, podrá utilizar el control Label para añadir un rótulo descriptivo en su formulario. Se tratará de la cuarta y última etiqueta que introducirá en el programa.
5. Pulse el control Label contenido en el cuadro de herramientas.
6. Cree un gran rectángulo justo debajo de los dos botones de orden.

Cuando acabe de introducir las cuatro etiquetas, el aspecto del formulario deberá ser similar al mostrado en la figura siguiente (modifique el tamaño de los rótulos si el aspecto difiere bastante al mostrado).



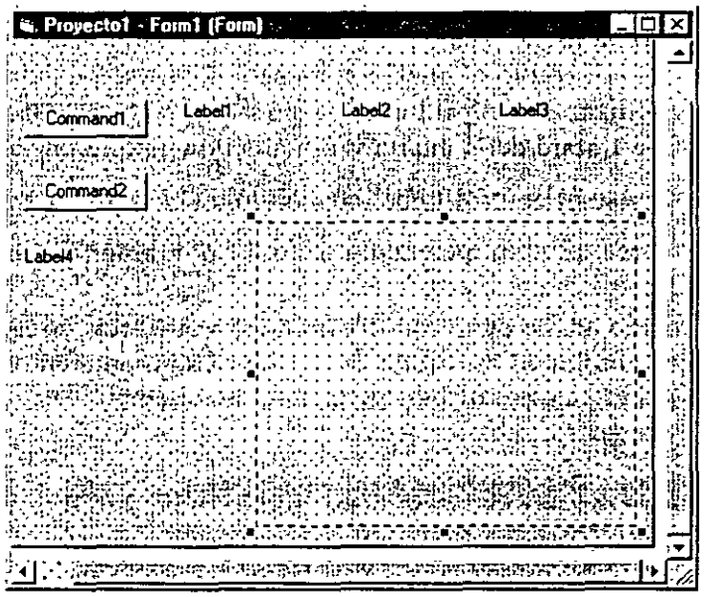
A continuación, añada un cuadro de imagen al formulario para mostrar la pi. de monedas que el jugador ganará cada vez que consiga un siete. Los cuadros de imagen están diseñados para mostrar mapas de bits, iconos y otros tipos de dibujos en sus programas. Uno de los mejores empleos que le podrá dar a sus cuadros de imagen es el de mostrar un dibujo «clip art» de Visual Basic.

Cómo añadir un dibujo

Control Image

1. Pulse el control Image contenido en el cuadro de herramientas.
2. Mediante este control podrá crear un gran cuadro rectangular justo debajo de las tres etiquetas numéricas.

   Una vez que haya terminado todo el proceso, el cuadro de imagen deberá tener un aspecto similar al mostrado en la siguiente figura.



Este objeto recibirá el nombre de Image1 en su programa; más adelante, verá este nombre en el código del programa.

Ahora ya está preparado para personalizar la interfaz definiendo unas cuantas propiedades.

Definición de las propiedades

Como ya ha visto anteriormente (en el programa Saludo), podrá modificar las propiedades asociadas con un objeto, seleccionándolo del formulario y utilizando su ventana de Propiedades. Comenzaremos a definir las propiedades de los elementos contenidos en este programa modificando los rótulos asociados con los dos botones de opciones.

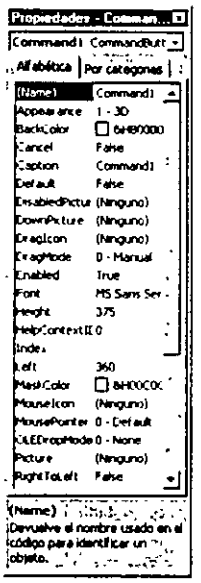
Definición de las propiedades del botón de orden

1. Pulse con el ratón sobre el primer botón de orden (Command1) contenido en el formulario.

   El botón de orden quedará seleccionado presentándose una serie de marcas cuadradas a su alrededor (los famosos «manejadores de selección»).

2. Realice una doble pulsación sobre la barra de títulos de la ventana Propiedades.

   En su pantalla aparecerá la ventana Propiedades, tal y como se muestra en la siguiente figura:



La ventana de Propiedades lista el conjunto de parámetros asociados con el primer botón de orden. Entre otros se incluyen valores para el color de fondo, rótulos, altura de la fuente y anchura del botón de orden.

3. Realice una doble pulsación sobre la propiedad Caption (rótulo) situada en la columna izquierda de la ventana de Propiedades.

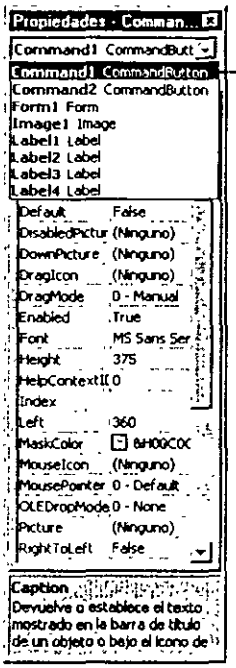
   El valor actual de la propiedad Caption («Command1») se resaltará en la columna de la derecha de la ventana de propiedades.

4. Escriba **Jugar** y pulse la tecla INTRO.

   La propiedad asociada con Caption cambiará a «Jugar» dentro de la ventana de Propiedades y, automáticamente, en el formulario. A continuación, podrá cambiar por el mismo procedimiento la etiqueta asociada con el segundo botón de orden introduciendo «Fin» (en esta ocasión, le mostraré cómo puede seleccionar el segundo botón de orden de una forma diferente).

5. Abra el cuadro de lista desplegable que se encuentra en la parte superior de la ventana de Propiedades.

   En el cuadro aparecerá la lista de los objetos de la interfaz contenidos en su programa, tal y como se muestra a continuación.



6. Seleccione Command2 (el segundo botón de orden) en el cuadro de lista.

   Los parámetros de propiedades asociados con el segundo botón de orden aparecerán en la ventana Propiedades.

7. Realice una doble pulsación sobre la propiedad Caption («Command2»), escriba **Fin** y pulse INTRO.

   El rótulo asociado con el segundo botón de orden cambiará para mostrar «Fin».

Nota: Emplear el cuadro de lista desplegable objeto es una forma muy útil de conmutar entre los distintos objetos presentes en un mismo programa. También podrá pasar de un objeto a otro pulsando con el ratón en el formulario sobre cada uno de ellos.

A continuación, redefiniremos las propiedades asociadas con los rótulos contenidos en el programa. Las tres primeras etiquetas serán las encargadas de mostrar los números aleatorios en el programa y los valores de sus propiedades serán idénticos (la mayoría de ellos se establecerán en modo grupo). Las propiedades asignadas al rótulo descriptivo serán ligeramente distintas.

Definición de las propiedades de las etiquetas numéricas

Para seleccionar simultáneamente más de un objeto de un formulario, pulse y mantenga pulsada la tecla MAYÚS mientras que con el ratón va pulsando sobre los objetos que desea seleccionar.

1. Pulse sobre la primera etiqueta numérica. A continuación, pulse y mantenga pulsada la tecla MAYÚS mientras que pulsa con el ratón sobre la segunda y tercera etiquetas numéricas (Si la ventana de Propiedades le estorba, sitúela en una nueva posición).

   Sobre cada una de estas etiquetas aparecerá un rectángulo de selección. Una vez seleccionadas las tres etiquetas, suelte la tecla MAYÚS.

Nota: *Como se ha seleccionado más de un objeto, sólo se mostrarán en esta ventana aquellas propiedades que puedan ser modificadas conjuntamente, es decir, como grupo. Podrá modificar las propiedades denominadas BorderStyle, Alignment y Font para que los números que aparezcan en estos cuadros estén centrados, recuadrados y con el tipo y tamaño de letra apropiado.*

---

2. Pulse sobre la propiedad Alignment (alineación) y, posteriormente, pulse sobre la flecha del cuadro de lista desplegable que aparece a la derecha.

   En el cuadro aparecerá una lista con las opciones de alineación existentes.

3. Pulse la opción 2-Center.

   La propiedad Alignment asociada con cada una de las etiquetas seleccionadas se modificará a 2-Center. A continuación, deberá modificar la propiedad BorderStyle (Estilo del borde).

4. Pulse con el ratón sobre la propiedad BorderStyle y, posteriormente, pulse sobre la flecha del cuadro de lista desplegable que aparece a su derecha.

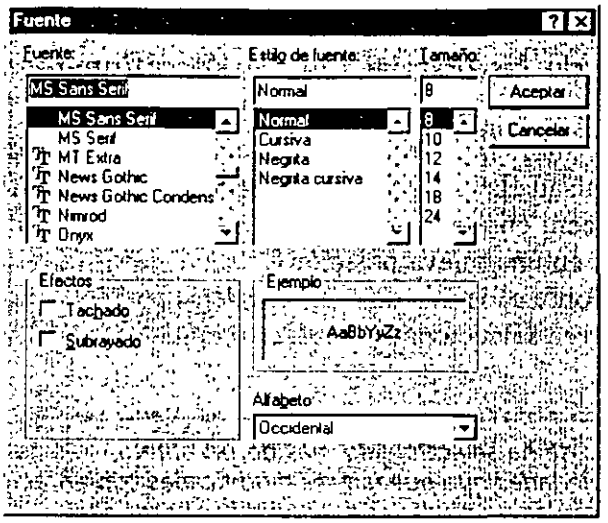
   En este cuadro aparecerá una lista con todos los valores válidos que podrá asociar con esta propiedad (0-None y 1-Fixed Single).

5. Seleccione 1-Fixed Single en el cuadro de lista para añadir un borde fino alrededor de cada una de las etiquetas.

   A continuación, se trata de variar la fuente utilizada en las etiquetas. Para ello, deberá modificar el valor asignado a la propiedad Font.

6. Realice una doble pulsación sobre la propiedad Font contenida en la ventana de Propiedades.

   En su pantalla aparecerá el cuadro de diálogo Fuente, tal y como se muestra a continuación:



7. Active la fuente Times New Roman, como estilo utilice Negrita y como tamaño de punto 24. Finalmente, pulse Aceptar.

   Los rótulos de las etiquetas aparecerán en el tipo de letra, estilo y tamaño que acaba de especificar. A continuación, borre los tres rótulos de tal forma que las cajas se muestren vacías cuando se ejecute el programa (las selecciones que haya realizado seguirán siendo válidas para los rótulos que se mostrarán en dichos cuadros, ya que las propiedades se almacenan de forma independiente). Para completar esta operación, necesitará seleccionar individualmente cada una de las etiquetas.

8. Pulse sobre cualquier parte libre del formulario para eliminar los manejadores de selección de las tres etiquetas y, finalmente, pulse sobre la primera de ellas.

9. Realice una doble pulsación sobre la propiedad Caption. Finalmente, pulse la tecla SUPR.

   Se borrará el rótulo relacionado con el objeto Label1. Más adelante, en este mismo capítulo, le mostraré cómo añadir código para introducir un número en este cuadro.

10. Borre los rótulos contenidos en la segunda y tercera etiquetas del formulario.

Con esta última operación ha terminado con las primeras tres etiquetas. A continuación le mostraré cómo puede cambiar las propiedades Font, ForeColor y Caption de la última etiqueta.

Definición de las propiedades del título del programa

1. Pulse con el ratón sobre el cuarto objeto de etiqueta contenido en el formulario.

2. Modifique la propiedad Caption para introducir el rótulo «Siete Afortunado».

3. Realice una doble pulsación sobre la propiedad Font y utilice el cuadro de diálogo Fuente para cambiar la fuente a Arial, el estilo a Negrita y el tamaño de punto a 20. Pulse sobre Aceptar.

   La fuente utilizada en el cuadro de etiqueta habrá cambiado. Observe que el texto contenido en el cuadro se habrá dispuesto en dos líneas porque ya no cabe en una sola. Estamos ante un hecho importante: el contenido de un objeto deberá caber dentro de dicho objeto. En caso contrario, será dividido en varias líneas o se verá truncado. A continuación le mostraré cómo cambiar el color de primer plano para el texto.

4. Realice una doble pulsación sobre la propiedad ForeColor contenida en la ventana Propiedades.

   Aparecerá un cuadro de lista conteniendo dos etiquetas Paleta y Sistema, lo que le proporcionará dos formas distintas para modificar el color asociado con el objeto seleccionado. La etiqueta Sistema muestra los colores que se están utilizando en ese momento para representar los elementos de la interfaz de su sistema (la lista refleja los parámetros definidos en la etiqueta Apariencia de la hoja de propiedades del escritorio de Windows). La etiqueta Paleta mostrará todos los colores disponibles en su sistema.

5. Pulse sobre la etiqueta Paleta y seleccione el cuadro de color morado oscuro.

   El texto contenido en el cuadro de etiqueta se mostrará ahora en color morado. El color que acaba de seleccionar se mostrará como un número hexadecimal en la ventana de Propiedades. La mayoría de los programadores no tendrán necesidad de utilizar este formato, pero resulta interesante ver cómo Visual Basic graba dicha información en el programa. A estas alturas ya se encuentra preparado para establecer las propiedades del último objeto.

Propiedades del cuadro imagen

El objeto cuadro de imagen contendrá el gráfico que representa la pila de monedas. Este gráfico aparecerá en pantalla cuando el usuario obtenga un «jackpot» (es decir, cuando dentro de los cuadros de etiqueta aparezca, al menos, un siete). Necesitará definir la propiedad Stretch para dar un tamaño apropiado al gráfico; del mismo modo deberá definir la propiedad Picture, que especifica el nombre del archivo gráfico que se cargará en el cuadro de imagen. También deberá asignar un valor a la propiedad Visible, que especifica el estado del dibujo al ponerse en marcha el programa.

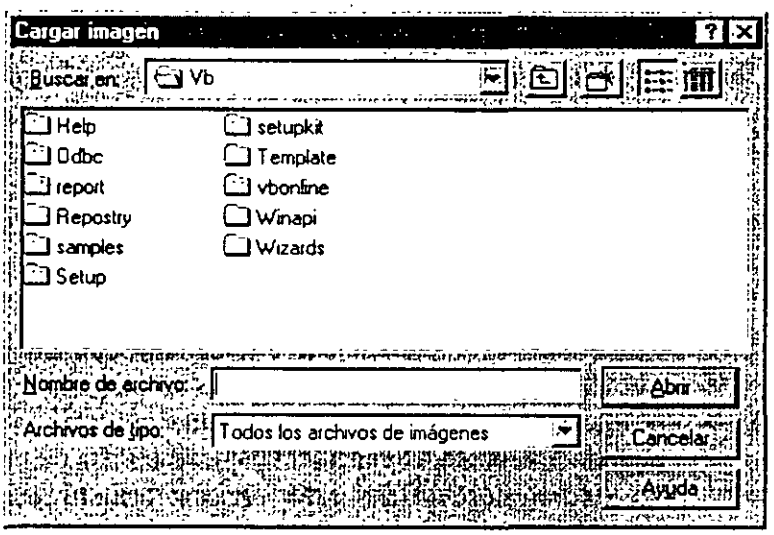
Definición de las propiedades del cuadro de imagen

1. Pulse el objeto cuadro de imagen contenido en el formulario.
2. Pulse la propiedad Stretch contenida en la ventana de Propiedades, pulse sobre la flecha situada a la derecha del cuadro de lista desplegable y seleccione el valor True.

Al definir como True el valor de Stretch (dilatar) antes de abrir un gráfico hará que Visual Basic modifique el valor del gráfico para que tenga las dimensiones exactas del cuadro de imagen (normalmente, tendrá que definir esta propiedad antes de configurar la propiedad Picture).

3. Realice una doble pulsación sobre la propiedad denominada Picture dentro de la ventana de Propiedades.

   En su pantalla aparecerá el cuadro de diálogo Cargar Imagen, tal y como se muestra en el siguiente figura.



4. Utilizando el botón Subir un nivel active la carpeta \Vb5Sbs dentro del cuadro de diálogo Cargar Imagen.

   En su pantalla aparecerán las subcarpetas contenidas en \Vb5Sbs.

5. Realice una doble pulsación sobre la carpeta denominada Less01.

   En el cuadro de diálogo denominado Cargar Imagen aparecerá el nombre del archivo Coins.wmf (meta-archivo de Windows). Este tipo de archivos contienen gráficos que pueden ser visualizados en una amplia variedad de tamaños; por ello, tienen un excelente aspecto independientemente de que el cuadro utilizado para mostrarlos sea grande o pequeño.

6. Seleccione el archivo Coins.wmf contenido en el cuadro de diálogo y pulse el botón Abrir.

   En el cuadro de imagen del formulario se mostrará el contenido del metafichero Coins.

   El programa «Siete Afortunado» tendrá que ocultar inicialmente esta imagen. Para ello, deberá asignar el valor False a la propiedad Visible para que las monedas no se muestren cuando se ponga en marcha el programa (las hará aparecer más tarde mediante código de programa).

7. Pulse sobre la propiedad Visible y, posteriormente, pulse de nuevo sobre la flecha del cuadro de lista desplegable asociado.

   En el cuadro de lista aparecerán los valores válidos que podrá asignarle a la propiedad Visible.

8. Pulse sobre False para que la imagen no aparezca cuando comience el programa.

   De esta forma, la propiedad Visible tendrá ahora el valor False. Este valor afectará al cuadro de imagen cuando se ejecute el programa pero no durante el período de diseño del mismo. El formulario completo tendrá el siguiente aspecto:

9. Realice una doble pulsación sobre la barra de títulos de la ventana Propiedades para volver a la posición fijada.

Desarrollo del código

A estas alturas, ya está preparado para escribir el código asociado con el programa Siete Afortunado. La mayor parte de los objetos que ya ha creado «conocen» cómo tendrán que trabajar cuando se ejecute el programa. Por ello, están preparados para recibir las instrucciones que desee comunicarle el usuario y procesarlas automáticamente. Uno de los puntos fuertes de Visual Basic es que todos los objetos cuentan con una serie de propiedades inherentes, y esto significa que cuando introduzca un objeto en uno de sus programas aquél ya contará con una serie de propiedades definidas y plenamente funcionales sin que tenga que desarrollar una sola línea de código.

Lectura en tablas de las propiedades

En este capítulo ha asignado de uno en uno el valor a las propiedades del programa Siete Afortunado. En próximos capítulos las instrucciones que deberá seguir para asignar nuevos valores a las propiedades del programa se le presentarán en forma de tabla, a menos que una de estas operaciones sea especialmente complicada. A continuación, le mostraré en la primera tabla el conjunto de propiedades que ya ha definido para el programa Siete Afortunado. A partir de ahora, todas las operaciones de definición de propiedades se mostrarán así:

Objeto	Propiedad	Valor
Command1	Caption	«Jugar»
Command2	Caption	«Fin»
Label1, Label2, Label3	BorderStyle	1 - Fixed Single
	Alignment	2 - Center
	Font	Times New Roman, Negrita, 24 puntos
	Caption	(Vacío)
Label4	Caption	«Siete Afortunado»
	Font	Arial, Negrita, 20 puntos
	ForeColor	Morado oscuro (&H008000808)
Image1	Picture	«\Vb5Sbs\Less1\coins.wmf»
	Stretch	True
	Visible	False

Sin embargo, el «meollo» del juego Siete Afortunado (que no es otro que el código encargado de calcular los números aleatorios, mostrarlos en los cuadros de etiqueta y detectar si se ha producido o no un «jackpot») todavía no ha sido introducido en el programa. Esta lógica de cálculo sólo podrá ser incluida en la aplicación utilizando *sentencias o instrucciones de programa* (código que indique claramente qué pasos deberá dar el programa en cada una de las situaciones presentadas). Como el funcionamiento del programa vendrá dado por el estado de los botones Jugar y Fin, tendrá que asociar el código a dichos botones. La *ventana Código* es una ventana especial dentro del entorno de programación que deberá utilizar para introducir o editar cualquier sentencia de programa en Visual Basic.

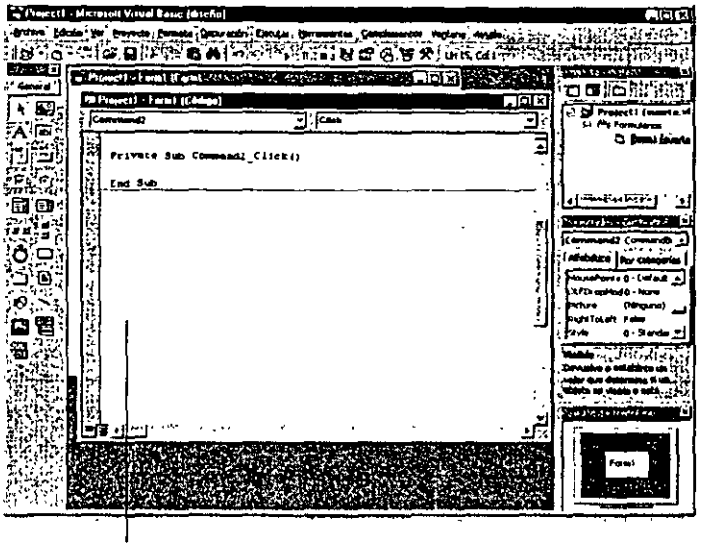
El código asociado con el programa se deberá introducir en la ventana Código

En los siguientes pasos le mostraré cómo introducir el código asociado con el programa Siete Afortunado dentro de la ventana Código.

Empleo de la ventana Código

1. Realice una doble pulsación sobre el botón de orden Fin contenido en el formulario.

   Al realizar la operación anterior aparecerá en su pantalla la ventana Código, que tendrá el aspecto mostrado en la figura siguiente:



Ventana de Código

Si la ventana es más pequeña que la que se muestra en la figura anterior, modifique su tamaño con el ratón (el tamaño exacto no es demasiado importante, ya que la ventana Código contiene barras de desplazamiento que podrá utilizar para examinar las instrucciones de gran longitud contenidas en el programa).

La ventana Código contendrá dos sentencias de programa, aquellas que marcan el comienzo y el final de esta particular subrutina de Visual Basic, también denominada *procedimiento de suceso* (un bloque de código asociado con un determinado objeto de la interfaz de usuario):

```
Private Sub Command2_Click()

End Sub
```

El cuerpo principal de un procedimiento siempre deberá encontrarse situado entre las dos sentencias anteriores. Dichas instrucciones se ejecutarán en cuanto el usuario active el elemento de la interfaz asociado con dicho procedimiento. En este caso, el suceso desencadenante será la pulsación del botón con el ratón, pero como verá más adelante en este libro, también podrá ser una operación de cualquier otro tipo.

La sentencia End finaliza la ejecución de un programa.

2. Escriba **End** y pulse la tecla FLECHA ABAJO.

   A medida que vaya escribiendo esta instrucción sus letras aparecerán en negro en la ventana Código. Cuando pulse la tecla FLECHA ABAJO (también podrá pulsar la tecla INTRO o, simplemente, situar el cursor con el ratón en cualquier otra línea de la ventana) la sentencia se mostrará en azul, indicando que Visual Basic la ha reconocido como una sentencia correcta, también denominada *palabra clave*. End es la instrucción que se utilizará para detener la ejecución del programa y eliminarlo de la pantalla. El sistema de programación de Visual Basic dispone de varios cientos de palabras claves como la anterior, a las que habrá que sumar sus operadores y símbolos asociados. Un tema crítico cuando se está introduciendo código es utilizar siempre la sintaxis apropiada para cada una de estas palabras clave, evitando, por ejemplo, cometer algún error ortográfico o introducir algún espacio en blanco en lugares inapropiados. De esta forma, permitirá que el compilador de Visual Basic reconozca adecuadamente las sentencias que usted introduzca en el cuerpo principal del programa.

Nota: El nombre utilizado para describir la ortografía correcta, los operadores que se pueden utilizar y otros temas relacionados con las palabras clave es: sintaxis.

---

3. Desplace el cursor al principio de la línea que contiene la instrucción End y pulse la BARRA ESPACIADORA un total de cuatro veces.

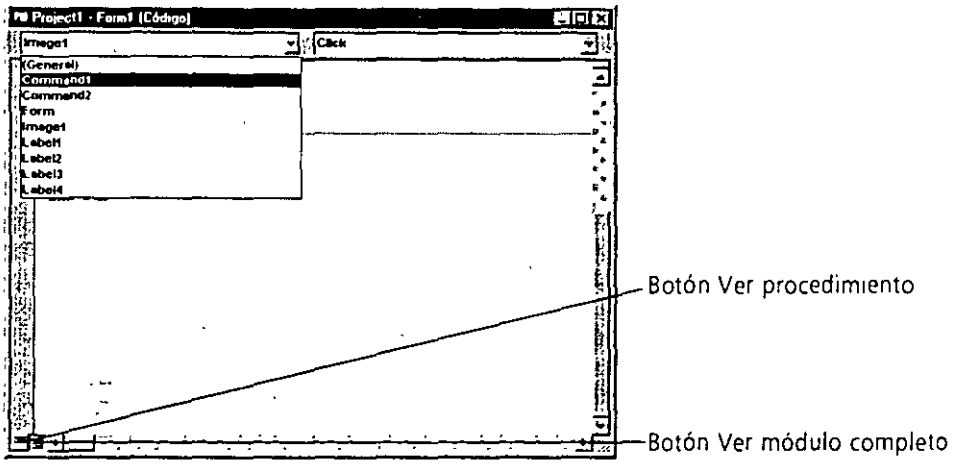
   El editor moverá la palabra End cuatro espacios hacia la derecha. De esta forma, dicha sentencia se podrá distinguir con claridad de las instrucciones Private Sub y End Sub. Este esquema de sangrías es una de las normas de programación que se utilizan en el presente libro para mejorar la legibilidad de los programas desarrollados. El grupo de normas relacionadas con la forma en que se tiene que organizar el código introducido en un programa se suele conocer como *estilo de programación*.

Ahora que ya ha introducido las instrucciones relacionadas con el botón Fin le toca el turno al código asociado con el botón Jugar. Estas sentencias de programación serán más complicadas y le brindarán la oportunidad de conocer más detalles sobre la sintaxis y el estilo de programación. Más adelante analizaremos en mayor profundidad cada una de las sentencias introducidas en este programa, por lo que no necesitará, por el momento, comprenderlas totalmente. Su objetivo ahora es fijarse en la estructura general del código del programa e introducir las sentencias tal y como se muestran en el presente libro (Visual Basic es muy estricto en lo referente a la corrección ortográfica de las sentencias y en el orden en que deben aparecer las palabras reservadas y los operadores).

Escritura de código para el botón Jugar

1. Despliegue el cuadro de lista denominado Objeto contenido dentro de la ventana Código.

   Los objetos pertenecientes a la interfaz de usuario del programa Siete Afortunado aparecerán en el cuadro de lista tal y como se muestra en la figura siguiente:



2. Pulse la opción Command1 dentro del cuadro de lista.

   Debajo del primer procedimiento aparecerá otro asociado con el botón Command1. Por defecto, Visual Basic muestra todos los procedimientos de suceso relacionados con un formulario en una única ventana; de esta forma podrá pasar con facilidad de uno a otro (entre dos procedimientos consecutivos aparecerá una línea recta, por lo que podrá diferenciar con facilidad el código asociado a cada uno de ellos). Por otro lado, podrá ver cada procedimiento en su propia ventana sin más que pulsar el pequeño botón denominado Ver Procedimiento situado en la esquina inferior izquierda de la ventana de Código. Para ver todos los procedimientos juntos otra vez en una única ventana, pulse el botón Ver módulo completo (situado a la derecha del botón Ver procedimiento).

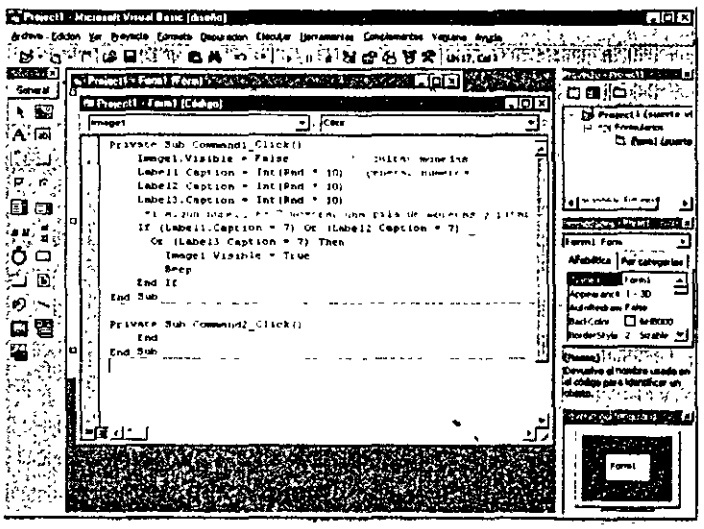
   Aunque usted haya cambiado el rótulo asociado con este botón y lo haya definido como «Jugar», su nombre en el programa sigue siendo Command1 (el nombre y el rótulo de un elemento de la interfaz puede ser diferente, para adaptarse a las necesidades del programador). Cada objeto puede contar con varios procedimientos asociados con él, uno para cada uno de los sucesos que pueda reconocer. El suceso «pulsar» (click) es, en este momento, en el que estamos interesados, ya que los usuarios podrán pulsar los botones Jugar y Fin cuando manejen el programa.

3. Introduzca las siguientes líneas del programa entre las sentencias Private Sub y End Sub, pulse INTRO después de terminar cada línea y ponga el máximo cuidado para escribir las sentencias del programa tal y como aparecen a continuación. Si comete algún error (normalmente serán fáciles de identificar porque aparecen en rojo), elimine la sentencia incorrecta y pruebe de nuevo.

```
Image1.Visible = False          ' ocultar monedas
Label1.Caption = Int(Rnd * 10)  ' generar números
Label2.Caption = Int(Rnd * 10)
Label3.Caption = Int(Rnd * 10)
'si algún número es 7 mostrar una pila de monedas y pitar
If (Label1.Caption = 7) Or (Label2.Caption = 7) _
    Or (Label3.Caption = 7) Then
        Image1.Visible = True
        Beep
End If
```

Nota: A medida que vaya introduciendo el código del programa, Visual Basic da formato al texto y muestra las diferentes partes del programa en distintos colores para ayudarle a identificar los diferentes elementos. Cuando comience a escribir una propiedad, Visual Basic mostrará las propiedades disponibles para el objeto que esté utilizando en un cuadro de lista; por ello, podrá realizar una doble pulsación sobre la propiedad o seguir escribiendo su nombre. Cuando Visual Basic muestre un mensaje de error será porque usted habrá introducido erróneamente una sentencia. Verifique que la línea resaltada está correctamente escrita (comparándola con la sentencia mostrada en este libro), realice las correcciones adecuadas y continúe escribiendo (también podrá borrar toda la sentencia y comenzar a escribir de nuevo). Los mandatos de menú y las teclas de edición que deberá utilizar en Visual Basic son similares a las opciones de edición que suele emplear en otras aplicaciones para Windows. Para obtener más información acerca de este tema consulte la ayuda en línea de Visual Basic.

Una vez introducido el código, la ventana Código tendrá el siguiente aspecto:



Estudio detallado del procedimiento Command1_Click

El procedimiento Command1_Click es el corazón del programa Siete Afortunado.

El procedimiento Command1_Click se ejecutará cuando el usuario pulse el botón Jugar contenido en el formulario. El procedimiento cuenta con algunas sentencias aparentemente complicadas y, como todavía no he hablado formalmente sobre ninguna de ellas, el aspecto global del programa puede resultar confuso, si no completamente ininteligible. Sin embargo, si se toma la molestia de analizar detenidamente el código, podrá detectar algunos detalles que quizás le resulten familiares. La lectura detallada del contenido de los procedimientos nos proporcionará una idea del tipo de programas que creará más adelante si sigue las instrucciones contenidas en este libro (si de momento no le interesa este tema, podrá saltar al siguiente apartado, denominado «Cómo almacenar el programa»).

El procedimiento Command1_Click realiza, en realidad, tres tareas: oculta la pila de monedas, crea tres números aleatorios que se mostrarán en las ventanas de rótulos y visualizará la pila de monedas cuando aparezca algún siete. Veremos con mayor detalle cada uno de estos pasos.

La primera tarea de este procedimiento viene descrita por la siguiente línea:

```
Image1.Visible = False          ' ocultar monedas
```

Esta línea está constituida por dos partes: una sentencia de programa y un comentario. La sentencia del programa (Image1.Visible = False) define como False (uno de los dos posibles) el valor de la propiedad Visible asignada al primer cuadro de imagen de la interfaz (el denominado Image1). Debe recordar que, anteriormente, ya definió como False el valor de esta propiedad utilizando la ventana Propiedades. Ahora está volviendo a hacer lo mismo en el código del programa porque la primera tarea es ejecutar una nueva «jugada» y necesitará borrar cualquier moneda que se muestre en pantalla como resultado de una jugada anterior.

Como el valor de esta propiedad necesita modificarse en tiempo de ejecución y no durante el proceso de diseño, tendrá que asignar el valor de la propiedad utilizando código de programa. Como puede comprender, se trata de una característica muy útil en Visual Basic y la analizaremos con más detalle en el Capítulo 2.

Los comentarios describen lo que hacen las sentencias del programa.

La segunda parte de la primera línea (aquella que se muestra en color verde en su pantalla) recibe el nombre de *comentario*. Los comentarios son notas explicativas incluidas en el código del programa y que se caracterizan por comenzar por una comilla simple ('). Los programadores utilizan los comentarios para describir la función que tienen en el programa las sentencias importantes. Estos comentarios no serán procesados por Visual Basic cuando se ejecute el programa; sólo existen como información describiendo qué hace el programa. En los programas que desarrolle con Visual Basic podrá utilizar con frecuencia comentarios escritos en su propia lengua para facilitar la relectura posterior del código.

Las siguientes tres líneas son las encargadas de calcular los tres números aleatorios. La función Rnd incluida en cada una de las líneas proporciona un número aleatorio comprendido entre 0 y 1 (un número decimal). La función Int multiplica dicho número aleatorio por 10 y trunca los decimales para obtener

únicamente el número entero. De esta forma se obtendrán números aleatorios comprendidos entre 0 y 9 (ambos inclusive). Finalmente, estos números se asignarán como valores a la propiedad Caption de las tres primeras etiquetas del programa. Esta asignación, además, hará que los números se muestren en negrita y en la fuente Times New Roman con un tamaño de 24 puntos dentro de las ventanas de etiquetas.

El último grupo de sentencias del programa comprueba si alguno de los números calculados es un siete. Si uno o más de ellos resulta ser siete, en la ventana del programa aparecerá la pila de monedas y sonará un pitido anunciando que se ha obtenido premio. En definitiva, cada vez que el usuario pulse el botón Jugar se llamará al procedimiento Command1_Click y se ejecutarán las sentencias asociadas.

Podrá almacenar el programa en su disco duro en cualquier momento del proceso de generación del mismo.

Cómo almacenar el programa

Ahora que ya ha terminado el programa Siete Afortunado podrá almacenarlo en el disco. Visual Basic guarda el código asociado con el formulario y los objetos manejados en un único archivo, mientras que la «lista de paquetes» será almacenada en otro archivo (en la ventana Proyecto se listarán los componentes del proyecto). Podrá utilizar de forma individual estos archivos de componentes en cualquier otro proyecto de programa utilizando el mandato Agregar Archivo contenido en el menú Proyecto. Para guardar un programa desarrollado en Visual Basic, deberá seleccionar la opción Guardar Proyecto Como contenida en el menú Archivo o pulsar con el ratón el botón Guardar Proyecto contenido en la barra de herramientas.

Grabación del programa Siete Afortunado

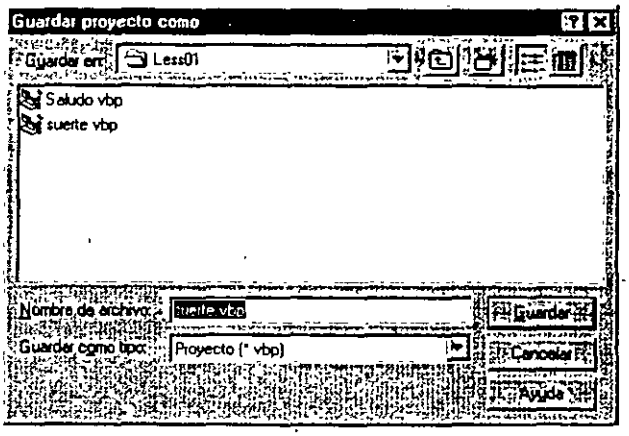
1. Despliegue el menú Archivo y seleccione la opción Guardar Proyecto Como.

   En su pantalla aparecerá el cuadro de diálogo Guardar archivo como, solicitándole el nombre y el lugar de almacenamiento que desea asignar a su formulario.

2. Seleccione la carpeta Less01 en el cuadro de diálogo, si es que todavía no se encuentra seleccionada. De esta forma, grabará el proyecto en la carpeta de prácticas que el programa de instalación de Aprenda Microsoft Visual Basic 5 Ya creó en su disco duro (podrá especificar una carpeta distinta si así lo desea).

3. Escriba **MiSuerte** en el cuadro de texto Nombre Archivo y pulse INTRO.

   El formulario Siete Afortunado se almacenará con el nombre MiSuerte.frm. A continuación, aparecerá en su pantalla el cuadro de diálogo Guardar Proyecto Como:



Nota: *En el presente libro le recomendaré que guarde siempre cada uno de los proyectos que desarrolle utilizando el prefijo* Mi. *De esta forma podrá apreciar sus progresos y conservar los archivos de prácticas originales. Así, podrá analizar los archivos originales en el caso de que surja algún problema.*

---

4. Escriba **MiSuerte** y pulse INTRO.

   El proyecto Siete Afortunado se almacenará con el nombre MiSuerte.vbp. Más adelante, si desea cargar de nuevo en memoria este proyecto, deberá seleccionar la opción Abrir Proyecto contenida en el menú Archivo y pulsar con el ratón sobre el nombre MiSuerte en el cuadro de diálogo Abrir Proyecto que aparecerá en su pantalla. También podrá cargar en memoria un proyecto utilizado de forma reciente pulsando el nombre del proyecto que aparecerá en la parte inferior del menú Archivo de Visual Basic.

El programa Suerte completo se encuentra almacenado en la carpeta \Vb5Sbs\Less01

¡Enhorabuena! Ya está en condiciones de ejecutar su primer programa real. Para ejecutar un programa de Visual Basic desde el entorno de programación deberá seleccionar la opción Iniciar contenida en el menú Ejecutar. También podrá hacerlo pulsando el botón Iniciar contenido en la barra de herramientas o pulsando, simplemente, la tecla F5.

Botón Iniciar

Intente poner en marcha el programa Siete Afortunado. Si Visual Basic muestra un mensaje de error, puede que haya cometido algún error ortográfico al escribir el código del programa. Intente detectarlo comparando la versión escrita del código que aparece en este libro con aquella que usted ha introducido en el programa, o cargue en memoria la versión del programa que venía en el CD de este libro.

Ejecución del programa

1. Pulse el botón Iniciar contenido en la barra de herramientas.

   Se ejecutará el programa Siete Afortunado en el entorno de programación. En su pantalla aparecerá la interfaz del usuario tal y como lo había diseñado.

2. Pulse el botón Jugar.

   El programa calculará y mostrará en pantalla tres números aleatorios, tal y como puede ver en la figura siguiente.

   En este caso, como ha aparecido un siete en el primer cuadro de etiqueta, la pila de monedas se muestra en pantalla y el ordenador habrá emitido un sonoro pitido (el tipo de sonido que oirá dependerá de la configuración del Panel de Control en Windows 95). ¡Habrá ganado!

3. Pulse el botón Jugar 15 o 16 veces más y observe los resultados de cada jugada que aparece en las ventanas de números.

   La mitad de veces que juegue volverá a ganar. Felicidades (la media de aciertos estará, aproximadamente, en tres veces de cada 10 jugadas; por eso, podemos decir que en un principio usted es un hombre afortunado). Como ve, se trata de un juego con altas probabilidades de premio. Quizás, más adelante, se sienta con fuerzas para hacer que el juego sólo muestre las monedas cuando se consigan simultáneamente dos o tres sietes. Le enseñaré a realizar esta labor cuando le comente algunos conceptos avanzados sobre módulos y variables públicas en el Capítulo 9.

4. Cuando haya terminado de experimentar con su programa, pulse el botón Fin.

   El programa se detendrá y el entorno de programación reaparecerá en su pantalla.

Creación de un archivo ejecutable

Los archivos .exe podrán ejecutarse bajo Windows 95 y Windows NT.

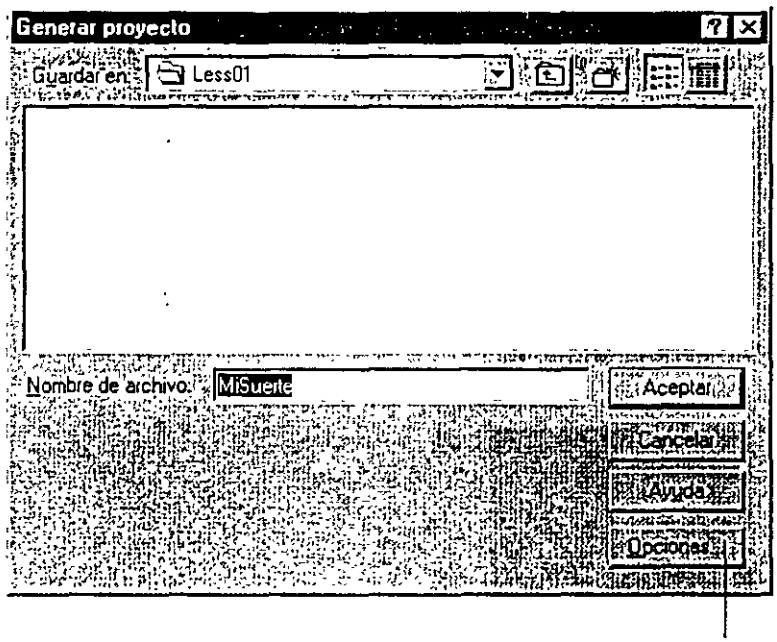
Su última tarea en este capítulo será completar el proceso de desarrollo y crear una aplicación para Windows, es decir, un archivo *ejecutable*. Las aplicaciones para Windows creadas con Visual Basic tienen la extensión .exe y podrán ejecutarse en cualquier computadora que tenga instalado Windows 95 o Windows NT ver-

sión 3.51 o posterior, además de los archivos de soporte necesarios (Visual Basic instala estos archivos de soporte, incluyendo las librerías de vínculos dinámicos y los controles personalizados, de forma automática. Si tiene pensado distribuir las aplicaciones que construya, consulte la *Guía del programador* de Visual Basic para obtener más información). A continuación le mostraré cómo crear el programa MiSuerte.exe.

Cómo crear un archivo ejecutable

1. En el menú Archivo seleccione el mandato Generar MiSuerte.exe (Visual Basic añadirá automáticamente el nombre del programa al nombre del mandato).

   En su pantalla aparecerá el cuadro de diálogo Generar Proyecto tal y como se muestra a continuación:



Utilice el botón Opciones para definir los parámetros avanzados del compilador.

El cuadro de diálogo contiene cuadros de texto y cuadros de lista que podrá utilizar para especificar el nombre y la localización en el disco del archivo ejecutable. También contiene un botón Opciones que podrá pulsar para abrir el cuadro de diálogo Propiedades del proyecto que podrá utilizar para controlar el icono asociado con el programa y otra información asociada con el archivo. Por defecto, Visual Basic le sugerirá la carpeta Less01 como destino del archivo ejecutable.

Truco: Como novedad presente en Visual Basic versión 5, el cuadro de diálogo Propiedades del proyecto (al que se accede desde el menú Proyecto) contiene una etiqueta denominada Compilar que podrá utilizar para controlar opciones avanzadas relacionadas con la compilación de su programa. Entre ellas se incluyen las necesarias para desarrollar código más rápido, eficiente, pequeño y compacto, además de corregir errores y de poder definir otras condiciones especiales de funcionamiento. Al añadir estas funciones sofisticadas al proceso de compilación, Visual Basic ha entrado en el mundo de las herramientas de desarrollo profesional, que tradicionalmente venían asociadas a otros compiladores más efectivos, tales como Microsoft Visual C++.

2. Pulse el botón Aceptar para aceptar el nombre del archivo y su localización propuestas por defecto por el programa.

   Visual Basic creará un programa ejecutable en el disco y lo almacenará en la carpeta especificada.

Si desea ejecutar más adelante este programa desde Windows 95, utilice el mandato Ejecutar del menú Inicio o realice una doble pulsación sobre el nombre del archivo en la ventana del Explorador de Windows. También podrá crear un acceso directo para el programa Siete Afortunado en la ventana del escritorio sin más que pulsar el botón derecho del ratón sobre la ventana del escritorio, seleccionar Nuevo y, finalmente, escogiendo la opción Acceso directo. Cuando Windows le solicite la localización del programa, pulse el botón Examinar y seleccione el archivo ejecutable MiSuerte contenido en la carpeta \Vb5Sbs\Less01. Pulse los botones Abrir, Siguiente y Finalizar, y Windows mostrará un icono en el escritorio que podrá pulsar para ejecutar el programa que acaba de crear. El aspecto del icono será el siguiente:

Por ahora, ha terminado con Visual Basic. En el siguiente ejercicio le mostraré como salir del entorno de programación.

Cómo salir de Visual Basic

1. Almacene los cambios que haya podido realizar en su programa.
2. Despliegue el menú Archivo y seleccione la opción Salir.

   Terminará la ejecución de Visual Basic.

UN PASO MÁS ALLA: ADICIÓN A UN PROGRAMA

Podrá volver a ejecutar Visual Basic en el momento que desee y trabajar en cualquier proyecto de programación que haya desarrollado anteriormente y que tenga almacenado en el disco. Como prácticas extraordinarias le mostraré cómo ejecutar otra vez Visual Basic y añadir en el programa Siete Afortunado una sentencia adicional denominada Randomize.

Cómo volver a cargar el programa Siete Afortunado

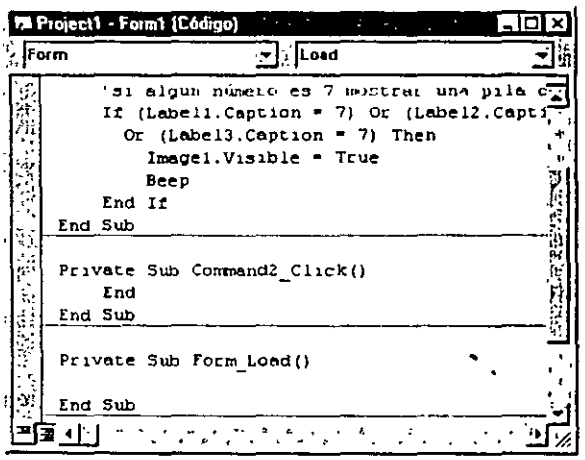
1. Pulse el botón Inicio contenido en la barra de tareas de Windows, despliegue el menú Programas, despliegue el submenú Visual Basic 5.0 y pulse el icono correspondiente al programa de Visual Basic 5.0.
2. Pulse sobre la etiqueta Reciente contenida en el cuadro de diálogo Nuevo proyecto.

   En su pantalla aparecerá una lista con los proyectos con los que haya trabajado de forma más reciente. Como acaba de terminar de trabajar con Siete Afortunado, el archivo MiSuerte deberá encontrarse el primero de dicha lista.
3. Realice una doble pulsación sobre MiSuerte para cargar el programa Siete Afortunado que está almacenado en el disco.

   En memoria se cargará el programa mencionado y en el formulario MiSuerte aparecerá en una ventana (si no lo ve, pulse el formulario MiSuerte contenido en la ventana Proyecto y vuelva a pulsar sobre el botón Ver Objeto).

   Ahora deberá añadir la sentencia Randomize al procedimiento Form_Load, un procedimiento especial que se encuentra asociado con el formulario y que se ejecutará cada vez que se ponga en marcha el programa.
4. Realice una doble pulsación sobre el formulario (pero no sobre uno de sus objetos) para mostrar el procedimiento Form_Load.

   El procedimiento Form_Load aparecerá en la ventana Código, tal y como se muestra en la figura siguiente:



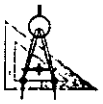
5. Pulse cuatro veces la tecla BARRA ESPACIADORA, escriba **Randomize** y pulse la tecla FLECHA ABAJO.

   La sentencia Randomize se encuentra ahora en el programa y se ejecutará cada vez que el programa se ponga en marcha. La instrucción Randomize utiliza el reloj de la computadora para crear un punto de inicio aleatorio que será utilizado posteriormente por la sentencia Rnd que utilizamos en el procedimiento Command1_Click. Quizás no llegara a darse cuenta, pero sin la función Randomize, el programa Siete Afortunado produciría siempre la misma cadena de combinaciones aleatorias cada vez que se inicie el programa. Una vez introducida la sentencia Randomize, el programa generará números realmente aleatorios cada vez que se ponga en marcha. Los números mostrados no seguirán ningún patrón reconocible.

6. Ejecute la nueva versión de «Siete Afortunado» y, finalmente, grabe la nueva versión del proyecto en el disco. Si tiene pensado utilizar con frecuencia el programa que acaba de crear quizás quiera generar también un nuevo archivo .exe. Visual Basic no actualiza automáticamente los archivos ejecutables cuando el programador cambia el código fuente.

Si desea continuar en el siguiente capítulo

➤ No salga de Visual Basic y pase al Capítulo 2.



Si desea salir ahora de Visual Basic

➤ En el menú Archivo seleccione Salir.
Si en su pantalla se muestra un cuadro de diálogo que le pregunta si desea guardar los cambios realizados, conteste que Sí.

RESUMEN DEL CAPÍTULO

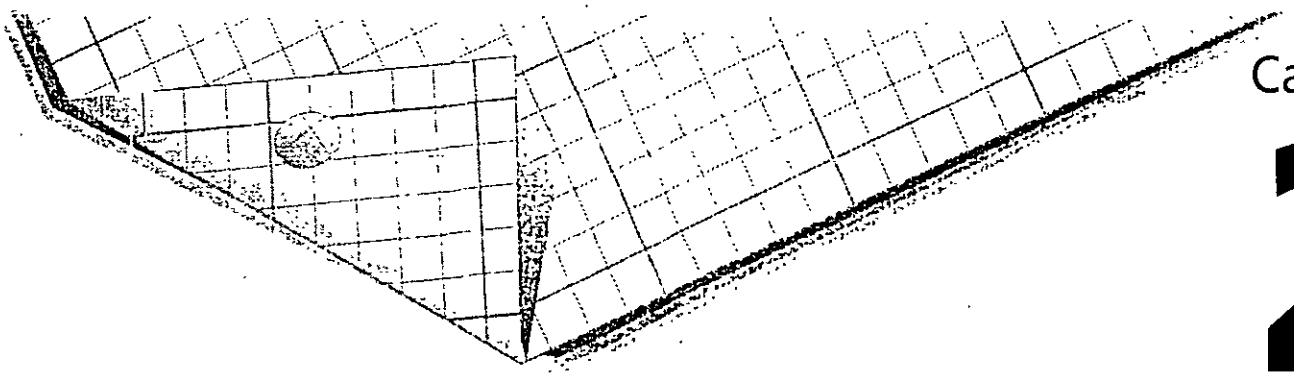
Para	Haga esto	Botón
Iniciar Visual Basic	Pulse el botón Inicio de la barra de tareas. Seleccione Programas, pulse sobre la carpeta de Visual Basic 5.0 y, finalmente, pulse el icono correspondiente al programa Visual Basic 5.0	
Mostrar la función asociada con un botón	Sitúe el puntero del ratón sobre el botón deseado	
Abrir un proyecto existente	En el menú Archivo seleccione la opción Abrir Proyecto.	
Comenzar un nuevo proyecto	En el menú Archivo seleccione la opción Nuevo Proyecto.	

Para	Haga esto	Botón
Ejecutar un programa	Pulse el botón Iniciar contenido en la barra de herramientas *o* Pulse F5	
Fijar una herramienta del programa	Pulsar sobre la barra de títulos y arrastrar la herramienta hasta el borde de otra ventana. Si desea aumentar el tamaño de la ventana acoplada realice una doble pulsación sobre la barra de títulos o modifique el tamaño con el ratón.	
Mover el cuadro de herramientas	Arrastre el cuadro de herramientas utilizando el ratón.	
Definir propiedades	Pulse el botón Ventana Propiedades contenido en la barra de herramientas para mostrar la ventana Propiedades (si es que no está ya abierta) y, finalmente, realice una doble pulsación sobre la barra de títulos de esta ventana. Despliegue el cuadro de lista Objeto para mostrar los elementos que componen la interfaz de usuario en su formulario y, finalmente, seleccione el valor que desee asignar en el cuadro de lista Propiedades.	
Mostrar la ventana Proyecto	Pulse el botón Explorador de proyecto de la barra de herramientas (si la ventana Proyecto no se encuentra abierta) y, a continuación, realice una doble pulsación sobre la barra de títulos de la ventana Proyecto.	
Crear una interfaz de usuario	Utilice los controles del cuadro de herramientas para introducir objetos en el formulario y, posteriormente, defina las propiedades que desee. Modifique el tamaño del formulario y de los objetos como guste.	
Mover un objeto	Arrastre el objeto por el formulario utilizando el ratón.	
Modificar el tamaño de un objeto	Seleccione el objeto y arrastre el manejador de selección del lateral que desee mover.	
Borrar un objeto	Seleccione el objeto y pulse SUPR	

Para	Haga esto	Botón
Abrir la ventana Código	Realice una doble pulsación sobre un objeto o sobre el propio formulario	
	o	
	pulse el botón Ver Código de la ventana Proyecto una vez resaltado el formulario o el nombre del módulo dentro de la ventana Proyecto.	
Escribir el código del programa	Escriba las sentencias del programa Visual Basic asociadas con el objeto que desee programar dentro de la ventana Código.	
Almacenar un programa	En el menú Archivo seleccione la opción Guardar Proyecto Como	
	o	
	pulse el botón Guardar Proyecto situado en la barra de herramientas.	
Crear un archivo .exe	En el menú Archivo, seleccione Generar *nombreficherо.exe*	
Salir de Visual Basic	En el menú Archivo, seleccione la opción Salir	
Volver a cargar un proyecto	En el menú Archivo seleccione la opción Abrir Proyecto	
	o	
	realice una doble pulsación sobre el nombre del archivo mostrado en la etiqueta Reciente del cuadro de diálogo Nuevo Proyecto.	
Sobre la ayuda en línea	**En el menú Ayuda de Visual Basic, seleccione el mandato Temas de ayuda de Microsoft Visual Basic, seleccione la etiqueta Índice y a continuación**	
Empleo de controles	Escriba **Controles, introducción** y pulse INTRO	
Definición de propiedades	Escriba **ventana Propiedades, elementos** y pulse INTRO	
Escritura del código de un programa	Escriba **código, escribir y editar** y pulse INTRO	

AVANCE DEL SIGUIENTE CAPÍTULO

En el siguiente capítulo, «Empleo de controles», explorará en detalle los controles de la interfaz de usuario contenidos en el cuadro de herramientas. Aprenderá a introducir objetos en el formulario, cómo hacer referencia a ellos en el código del programa y cómo utilizarlos de manera eficaz en los programas.



Capítulo

2

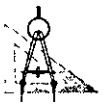
Empleo de controles

Como vio en el Capítulo 1, los controles de Microsoft Visual Basic son las herramientas gráficas que utilizará para construir la interfaz de usuario de cualquier programa desarrollado con este entorno de programación. Los controles están localizados en el cuadro de herramientas dentro del entorno de programación y podrá utilizarlos para introducir objetos en un formulario utilizando una serie simple de pulsaciones de ratón y de movimientos de arrastre. En este capítulo aprenderá a mostrar información en un cuadro de texto, analizar el contenido de unidades de disco y carpetas contenidos en su sistema, procesar los datos introducidos por el usuario, poner en marcha aplicaciones basadas en Microsoft Windows y ver los registros de una base de datos. Los ejercicios contenidos en este capítulo le ayudarán a diseñar sus propias aplicaciones en Visual Basic y le proporcionarán más información sobre objetos, propiedades y código de programa. También aprenderá a añadir controles ActiveX (anteriormente conocidos como controles OLE) en el cuadro de herramientas con el objetivo de extender las funciones asociadas a Visual Basic.

EMPLEO BÁSICO DE LOS CONTROLES: PROGRAMA HOLA MUNDO

Una buena tradición en los libros de introducción a la programación es la presentación del programa Hola mundo. Este es el nombre asignado a un breve programa que demuestra la facilidad con la que se puede crear y ejecutar un determinada utilidad en un lenguaje de programación dado. En la época en que dominaba la programación basada en carácter, la utilidad Hola mundo solía ser un programa de dos o tres líneas de código, que se escribían haciendo uso de un editor de texto y se compilaban con un compilador autónomo. Sin embargo, con la llegada de las herramientas de programación gráficas, el programa Hola mundo fue creciendo has-

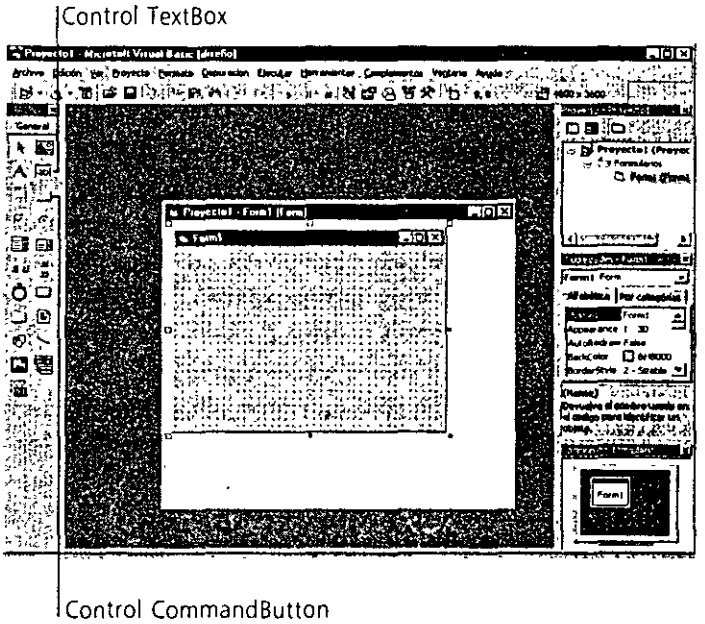
ta convertirse en un programa complejo que contiene docenas de líneas de programación y necesita varias herramientas de programación para su construcción. Por fortuna, crear un programa del tipo Hola mundo sigue siendo una labor sencilla si se utiliza Visual Basic. Podrá construir una interfaz de usuario completa creando dos objetos, definiendo dos propiedades y escribiendo, únicamente, una línea de código. Vamos a intentarlo.



Creación del programa Hola mundo

1. Inicie Visual Basic y elija crear una nueva aplicación estándar de 32 bits.

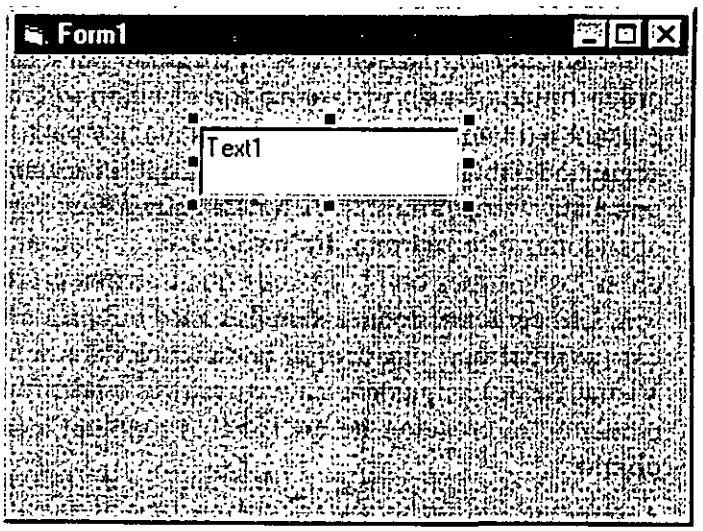
   En su pantalla aparecerá el entorno de programación de Visual Basic, tal y como se muestra en la siguiente figura. Los dos controles que utilizará en este ejercicio (TextBox y CommandButton) están marcados en la figura.



Control TextBox

2. Pulse con el ratón el control TextBox contenido en el cuadro de herramientas.
3. Sitúe el puntero del ratón en la parte superior central del formulario (el puntero se convertirá en una cruz cuando se encuentre sobre algún lugar del formulario) y dibuje un cuadro de texto similar al que se muestra en la figura de la página siguiente.

   Se utilizan los cuadros de texto para mostrar un rótulo en el formulario o para solicitarle al usuario que introduzca información durante la ejecución del programa desarrollado en Visual Basic. La forma en que trabajará el cuadro de texto dependerá de las propiedades que le asocie y del modo en que haga referencia a este elemento dentro del código del programa. En esta sencilla aplicación se utilizará un cuadro de texto para mostrar el men-



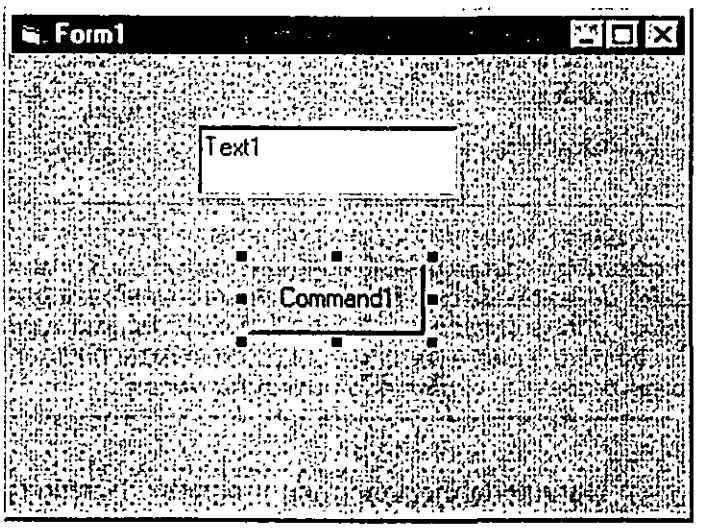
saje «¡Hola, Mundo!» siempre que el usuario pulse sobre un determinado botón de orden contenido en el formulario.

A continuación le mostraré cómo añadir el botón de orden.

Control CommandButton

4. Pulse el control CommandButton contenido en el cuadro de herramientas.
5. Desplace el puntero del ratón hasta situarlo justo debajo del cuadro de texto y dibuje el botón de orden.

   Su pantalla deberá tener ahora un aspecto similar al siguiente:



El botón de orden es el método más sencillo para interaccionar con el usuario. Cuando un usuario pulsa un botón de orden estará solicitando que el programa lleve a cabo una acción determinada de forma inmediata.

En términos de Visual Basic, el usuario utilizará el botón de orden para crear un *suceso* que deberá ser procesado por el programa. Los botones de orden típicos en un programa son el botón OK o Aceptar, que el usuario pulsará para aceptar una lista de opciones y para indicar que está listo para seguir adelante; el botón Cancelar, que el usuario pulsará para descartar una lista de opciones; y el botón Salir o Fin, que el usuario pulsará para abandonar el programa. En cada caso, el diseñador del programa necesita crear los botones con la idea de que trabajen correctamente cuando se pulsen. Las características asociadas a los botones de orden se podrán modificar (al igual que sucede con la de todos los tipos de objetos utilizables en Visual Basic) asignando nuevos valores a las propiedades de los mismos y mediante la creación de código de programa que haga referencia a dicho objeto.

Si desea obtener más información sobre la definición de propiedades de configuración, consulte el apartado titulado «Definición de las propiedades» en el Capítulo 1.

6. Asigne las siguientes propiedades a los objetos cuadro de texto y botón de orden, utilizando la ventana Propiedades. El valor Vacío significa que deberá borrar el valor actual y dejar vacía el espacio correspondiente a esa propiedad. Los valores que deberá introducir se muestran encerrados entre comillas dobles. En el programa no deberá introducir estas comillas.

Control	Propiedad	Valor
Text1	Text	(Vacío)
Command1	Caption	«Aceptar»

El programa completo Hola.vdp se encuentra localizado en el disco en la carpeta \Vb5Sbs\Less02

7. Realice una doble pulsación sobre el botón Aceptar e introduzca la siguiente sentencia entre las instrucciones Private Sub y End Sub dentro de la ventana Código:

```
Text1.Text = "¡Hola, Mundo!"
```

Truco: Cuando acabe de introducir el nombre de objeto Text1 y escriba el punto, Visual Basic mostrará un cuadro de lista conteniendo todas las propiedades válidas para los objetos cuadro de texto; de esta forma refrescará su memoria por si usted ha olvidado la lista completa. Podrá seleccionar la propiedad de dicha lista realizando una doble pulsación sobre su nombre o podrá continuar escribiendo hasta finalizar (normalmente yo sigo escribiendo salvo que esté explorando nuevas funciones).

Durante la ejecución del programa, la sentencia anterior modificará el valor de la propiedad Text relacionada con el cuadro de texto y le asignará el valor «¡Hola, Mundo!» cuando el usuario pulse el botón de orden (el signo de igualdad asignará a la propiedad Text del objeto Text1 cualquier cosa que se encuentre encerrada entre las comillas). La instrucción anterior modificará el valor asignado a la propiedad durante la ejecución del pro-

grama (en tiempo de ejecución); se trata de uno de los usos más comunes del código en un programa desarrollado con Visual Basic. Como la sentencia se encuentra dentro de lo que denominaríamos *procedimiento de suceso* (que se ejecuta siempre que el usuario pulse sobre el botón Command1), el valor asignado a la propiedad de texto (y, por lo tanto, el contenido del cuadro de texto) cambiará automáticamente en cuanto se pulse dicho botón de orden.

8. Use la ventana de Posición del formulario para definir la posición del formulario cuando se ejecute el programa (si la ventana Posición del formulario no se encuentra visible, ejecute el mandato Ventana Posición del formulario contenido en el menú Ver).

   Por defecto, el formulario se situará en la esquina superior izquierda de la pantalla, pero podrá modificar este hecho arrastrando el pequeño icono mostrado dentro de la ventana Posición del formulario. Esta función es especialmente útil en programas que muestran más de una ventana.

En este momento se encuentra preparado para ejecutar el programa Hola mundo y para almacenarlo en el disco.

Ejecución del programa Hola Mundo

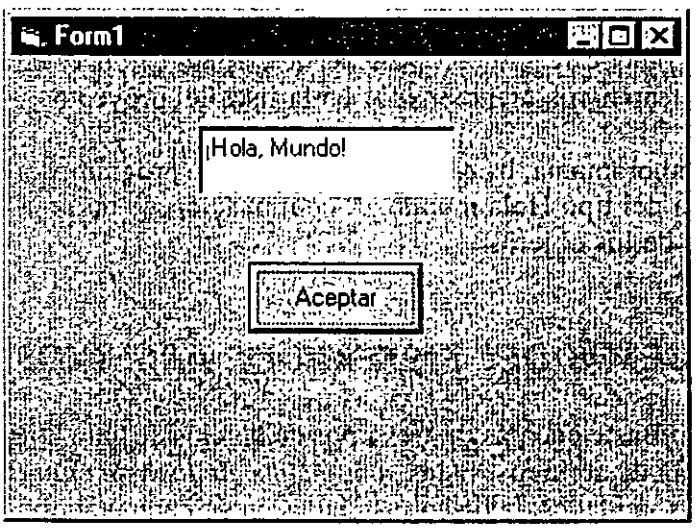
Botón Iniciar

1. Pulse el botón Iniciar contenido en la barra de herramientas.

   El programa Hola mundo se ejecutará en el entorno de programación de Visual Basic.

2. Pulse el botón de orden Aceptar.

   El programa mostrará en pantalla el saludo «¡Hola, Mundo!» dentro del cuadro de texto, tal y como se muestra a continuación:



Cuando pulse el botón de orden Aceptar el código del programa modificará la propiedad Text perteneciente al cuadro de texto Text1 asignándole el valor «¡Hola, Mundo!», rótulo que será el que se muestre dentro del cuadro. Si no ha obtenido este resultado repita los pasos comentados en el apartado anterior y vuelva a construir el programa. Puede ser que haya definido de forma incorrecta una propiedad o que haya cometido un error ortográfico cuando tecleó la sentencia del código del programa (los errores ortográficos aparecen en rojo en la pantalla).


Botón Terminar

3. Pulse el botón Terminar contenido en la barra de herramientas para detener el programa.

   También podrá parar la ejecución del mismo pulsando sobre el botón Cerrar del formulario del programa.

4. En el menú Archivo, seleccione la opción Guardar Proyecto Como.

5. Active la carpeta denominada \Vb5Sbs\Less02, escriba MiHola y pulse el botón Guardar.

   Visual Basic almacenará el formulario en el disco con el nombre MiHola.frm. Visual Basic graba por separado los formularios de los archivos de proyecto, por lo que podrá reutilizar los formularios y procedimientos en otros proyectos de programación futuros, sin tener que comenzar todo partiendo desde cero.

   Una vez que haya almacenado el formulario, Visual Basic le pedirá que introduzca un nombre para el proyecto (es decir, un nombre para el archivo que utiliza Visual Basic para construir su programa). Este archivo recibe el nombre de «archivo de proyecto de Visual Basic» y cuenta con la extensión .vbp.

6. Vuelva a escribir el nombre **MiHola** y pulse Guardar.

   Visual Basic almacenará su proyecto en el disco bajo el nombre de MiHola.vbp. Cuando desee abrir más adelante este proyecto, deberá seleccionar este archivo utilizando el mandato Abrir Proyecto contenido en el menú Archivo. Visual Basic cargará entonces en memoria cada uno de los archivos mencionados en la lista del proyecto.

Enhorabuena, ha entrado en el grupo de programadores que han escrito un programa del tipo Hola mundo. A continuación, le invito a entrar en el mundo fascinante de los objetos.

USO DE LOS OBJETOS DEL SISTEMA DE ARCHIVOS

Visual Basic cuenta con tres tipos de objetos de gran utilidad para poder acceder al sistema de archivos. Éstos son: *cuadro de lista de unidad*, que le permitirá examinar las unidades de disco instaladas en su computadora; *cuadro de lista de direc-*

torios, que le permitirá navegar por las carpetas contenidas en una determinada unidad de disco; y *cuadro de lista de archivos*, que le permitirá seleccionar un determinado archivo contenido dentro de una carpeta y unidad de disco. En el siguiente ejercicio utilizará los tres objetos relacionados con el sistema de archivos para construir un programa denominado «Navegador» con el que podrá buscar y mostrar archivos contenidos en su sistema y que almacenen imágenes gráficas.

Nota: También utilizará un objeto de imagen en este programa. Los objetos de imagen pueden visualizar seis tipos de formatos gráficos: mapas de bits (archivos .bmp), metaarchivos de Windows (archivos .wmf, que contendrá imágenes gráficas cuyo tamaño podrá modificar), iconos (archivos .ico), cursores (archivos .cur), formato JPEG (archivos .jpg) y formato GIF (archivos .gif).

El programa de navegación

El programa Navegador utilizará los tres objetos relacionados con el sistema de archivos, un objeto imagen y varias líneas de código de programación. El objetivo final de toda esta serie de elementos es crear un programa de navegación para localizar con facilidad los archivos gráficos y artísticos que tenga almacenados en su sistema. Una vez terminado el Navegador, podrá utilizarlo en su trabajo diario para localizar este tipo de archivos en las unidades de disquete, discos fijos, unidades de red o unidades CD-ROM. Los objetos del tipo sistema de archivos pueden trabajar con todos los tipos de unidades.

Construcción del programa Navegador

1. En el menú Archivo seleccione la opción Nuevo Proyecto y pulse el botón Aceptar para crear un nuevo archivo ejecutable estándar.

   El programa Hola mundo desaparece y en su pantalla se mostrará un formulario vacío. En este momento tendrá la oportunidad de almacenar cualquier modificación que haya realizado en el programa Hola mundo y que todavía no haya sido grabada en su disco fijo.

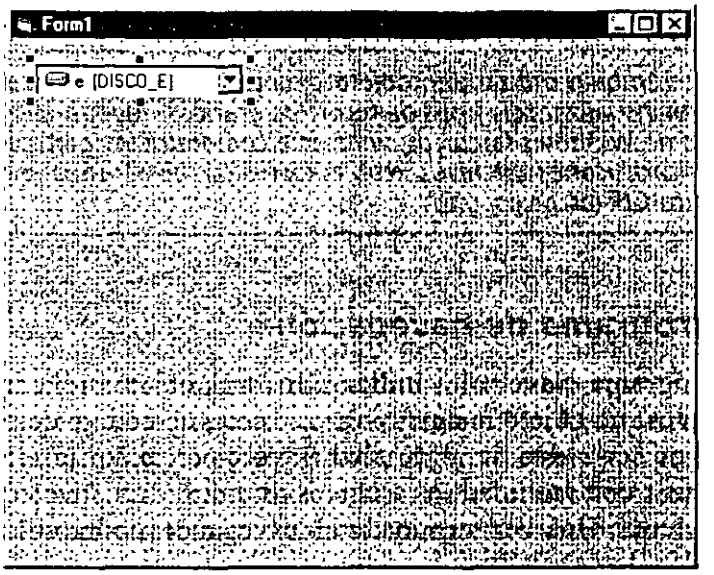
2. En el menú Herramientas, seleccione Opciones y pulse sobre la etiqueta Editor. Si la casilla de verificación denominada Requerir declaración de variables se encuentra seleccionada, pulse sobre la misma para eliminar la marca de verificación (esta opción se analizará en profundidad en el Capítulo 4). Pulse Aceptar.

3. Aumente el tamaño del formulario para que sea lo suficientemente grande como para almacenar todos los controles del sistema de archivos y una ventana con el tamaño adecuado para mostrar imágenes.

   Aparecerán barras de desplazamiento en torno al formulario para que pueda ver las partes ocultas a medida que vaya desarrollando el programa.

Control DriveListBox

4. Pulse el control DriveListBox almacenado en el cuadro de herramientas.
5. Desplace el puntero del ratón a la esquina superior izquierda del formulario y dibuje un cuadro de lista de unidades, tal y como se muestra en la siguiente figura:



Visual Basic mostrará dentro de este objeto el nombre y la etiqueta de volumen de la unidad de disco activa en ese momento. Esta información se mostrará para que el usuario pueda identificar con facilidad la unidad de discos que se encuentra activa en el momento en que esté utilizando el programa. También será de utilidad para el programador que esté desarrollando el programa para que pueda asignar a este objeto un tamaño adecuado. Si en su pantalla no se muestra correctamente toda la información relacionada con la unidad de discos activa, aumente el tamaño del cuadro de lista de unidades hasta que estos datos se visualicen correctamente.

Control DirListBox

6. Pulse el control DirListBox contenido en el cuadro de herramientas para añadir un cuadro de lista de directorios al formulario que está desarrollando. Introduzca este nuevo cuadro justo debajo del anterior. Deje suficiente espacio para mostrar, al menos, cuatro o cinco nombres de carpetas en este cuadro de lista.

   Los objetos del cuadro de lista de directorios le permitirán acceder a las carpetas contenidas en el sistema de archivos. Cuando introduzca un objeto de este tipo en un formulario de Visual Basic, dentro de este cuadro se mostrarán las carpetas en la misma forma en que lo harán cuando se ejecute el programa. Seguro que es tentador comenzar ahora a pulsar sobre las distintas carpetas mostradas, pero como el cuadro de lista de directorios no se encuentra activo, no ocurrirá nada. En este instante los nombres de las

carpetas aparecerán con el único objetivo de que usted pueda asignar un tamaño apropiado al objeto.


Control FileListBox

7. Pulse el control FileListBox contenido en el cuadro de herramientas y, a continuación, añada un cuadro de lista de archivos al formulario, justo debajo de donde se encuentra el cuadro de lista de directorios. Deje el suficiente espacio para que dentro de esta caja se puedan mostrar, al menos, cuatro o cinco nombres de archivos.

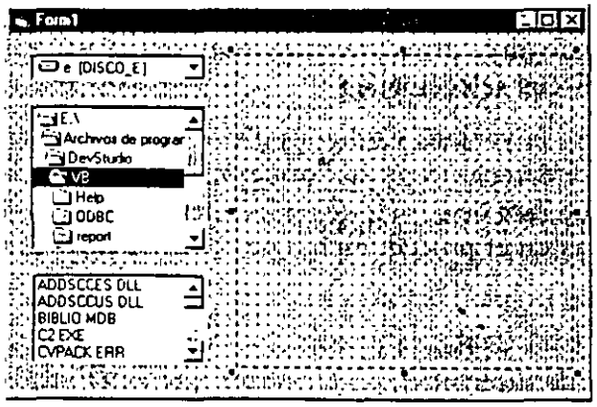
   Los objetos del tipo cuadro de lista de archivos permitirán que el usuario del programa seleccione un archivo perteneciente al sistema de archivos. Cuando el usuario seleccione un archivo, Visual Basic asignará su nombre en la propiedad FileName del objeto cuadro de lista de archivos. De igual manera, la propiedad Drive perteneciente al objeto cuadro de lista de unidades y la propiedad Path del objeto cuadro de lista de directorios adoptarán el valor de la unidad de disco y la carpeta que el usuario seleccione en los correspondientes cuadros de lista. En el programa Navegador que estamos utilizando, haremos uso posteriormente de estas tres propiedades para poder abrir el archivo de imagen deseado. Éste es un ejemplo típico del empleo de objetos y propiedades en un programa. El usuario modifica el valor asignado a un objeto durante la ejecución del programa, el cambio es asignado a la propiedad, y la propiedad se procesa en el código del programa.

Nota: *Las propiedades Drive, Path y FileName sólo podrán ser asignadas en tiempo de ejecución (contienen valores que serán asignados cuando se utilicen los cuadros de lista del sistema de archivos). No podrá asignarles un valor cuando esté utilizando la ventana Propiedades.*


Control Image

8. Pulse el control Image contenido en el cuadro de herramientas y añada en el formulario un cuadro de imagen de gran tamaño. Deberá situar este cuadro de imagen a la derecha de los cuadros de lista de unidades, directorios y archivos.

   Una vez que haya introducido el objeto imagen, su pantalla deberá tener un aspecto similar al mostrado en la siguiente figura:



9. A continuación defina las siguientes propiedades utilizando la ventana Propiedades:

Objeto	Propiedad	Valor
File1	Pattern	*.bmp;*.wmf;*.ico
Image1	Stretch	True
Image1	BorderStyle	1-Fixed Single

La propiedad Pattern perteneciente al cuadro de lista de archivos es especialmente importante en este caso. Listará los formatos gráficos válidos que Visual Basic puede mostrar en un programa utilizando un cuadro de imagen. Si a esta propiedad no se le asigna ningún valor (se la deja en blanco), el cuadro de lista de archivos listaría todos los archivos contenidos en la carpeta, y si el usuario selecciona un formato gráfico con el cual Visual Basic no sea compatible (tal como TIFF), esta acción produciría un error en tiempo de ejecución. Si es posible, siempre es mejor eliminar este tipo de situaciones antes de que aparezcan.

A continuación le mostraré cómo añadir unas cuantas líneas de programa a los procedimientos asociados con los objetos del sistema de archivos. Estos procedimientos reciben el nombre de *procedimientos de suceso* porque se ejecutan cuando se produce un suceso (tal como la pulsación del ratón) en el objeto.

Realice una doble pulsación sobre un objeto para mostrar su procedimiento de suceso asociado por defecto.

10. Realice una doble pulsación sobre el cuadro de lista de unidades contenido en el formulario y, posteriormente, introduzca la siguiente instrucción del programa entre las sentencias Private Sub y End Sub dentro del procedimiento de suceso denominado Drive1_Change:

```
Dir1.Path = Drive1.Drive
```

Esta instrucción actualizará el valor de la propiedad Path perteneciente al cuadro de lista de directorios cuando el usuario del programa seleccione una unidad de discos distinta dentro del cuadro de lista de unidades. Esta instrucción relaciona estos dos objetos de tal forma que el cuadro de lista de directorios mostrará siempre las carpetas contenidas en la unidad de discos seleccionada.

11. Cierre la ventana Código (pulse el botón Cerrar situado en la esquina superior derecha) y, posteriormente, realice una doble pulsación sobre el cuadro de lista de directorios. Finalmente, añada la siguiente sentencia de programa en el procedimiento de suceso denominado Dir1_Change:

```
File1.Path=Dir1.Path
```

Esta sentencia enlaza el cuadro de lista de archivos con el cuadro de lista de directorios de tal forma que en el cuadro de lista de archivos se

mostrarán siempre los archivos contenidos en la carpeta seleccionada en el otro cuadro.

12. Cierre la ventana Código y realice una doble pulsación sobre el cuadro de lista de archivos. Añada la siguiente líneas de código al procedimiento de suceso denominado File1_Click:

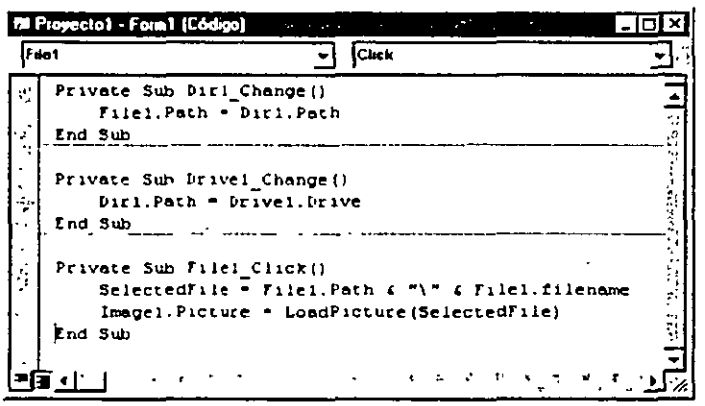
```
SelectedFile=File1.Path & "\" &File1.filename
Image1.Picture = LoadPicture(SelectedFile)
```

Estas dos líneas son el corazón del programa. La primera línea utiliza el operador «&» para combinar la propiedad Path de File1, el carácter «\» y la propiedad Filename de File1. El resultado se almacena en la variable denominada SelectedFile (archivo seleccionado). Una *variable* es un espacio de almacenamiento que se reserva de forma temporal para guardar la información manejada por el programa. En este caso, la variable SelectedFile almacenará el nombre completo del archivo que ha sido seleccionado por el usuario (incluyendo los nombres de la unidad y de la carpeta que lo contiene). La segunda sentencia contenida en el procedimiento de suceso utiliza la variable SelectedFile cuando carga el archivo dentro del cuadro de imagen (Image1) del formulario con la función LoadPicture y la propiedad Picture.

Aprenderá más acerca de los operadores, variables y funciones en el Capítulo 4.

Una vez que haya escrito el código para el procedimiento de suceso File1_Click, la ventana Código tendrá un aspecto similar al que ve en la figura (esta figura muestra una ventana Código con un tamaño aumentado).

El programa Navegador.vdp completo está localizado en su disco fijo dentro de la carpeta \Vb5Sbs\Less02



En este momento ya está preparado para ejecutar el programa Navegador y para almacenar su contenido en el disco fijo.

Ejecución del programa Navegador


Botón Iniciar

1. Pulse el botón Iniciar contenido en la barra de herramientas.

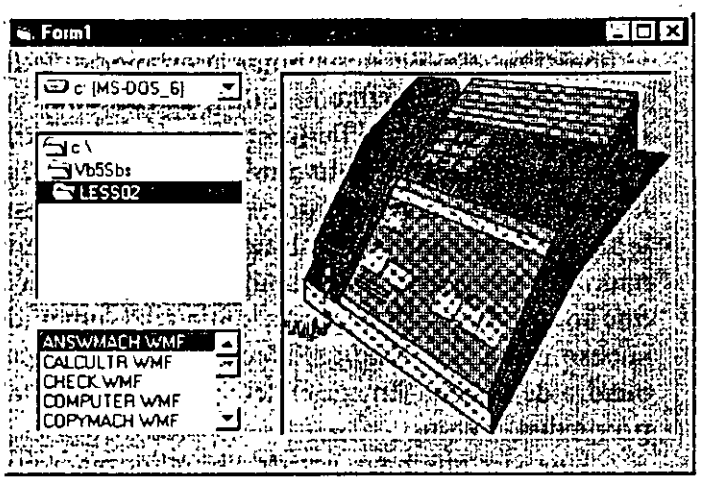
El programa Navegador se ejecutará en el entorno de programación.

2. Abra la carpeta \Vb5Sbs\Less02 utilizando el cuadro de lista de directorios.

   Los metaarchivos de Windows contenidos en esta carpeta aparecerán en el cuadro de lista de archivos.

3. Seleccione el archivo denominado answmach.wmf.

   El archivo seleccionado (un dibujo de un contestador automático) se mostrará en el cuadro de imagen, tal y como puede ver a continuación:



4. Muestre el final de la lista de archivos y seleccione el nombre poundbag.wmf.

   En el cuadro de imagen aparecerá el dibujo de un saquito de libras esterlinas.

5. Utilice los cuadros de lista de unidades, directorios y archivos para ver otros mapas de bits, metaarchivos de Windows e iconos contenidos en su computadora.

   Seguramente, encontrará varios archivos de mapas de bits interesantes en la carpeta Windows.

   Cuando termine de experimentar con el programa Navegador, detenga el programa y grábelo en el disco.

6. Pulse el botón Cerrar del formulario.

7. En el menú Archivo seleccione la opción Guardar Proyecto Como y grabe el formulario con el nombre **MiNavegador** y asigne el mismo nombre al archivo de proyecto.

OBJETOS PARA CONSEGUIR LA ENTRADA DE DATOS

Visual Basic cuenta con varios objetos con los que podrá gestionar la entrada de datos en el programa. Los cuadros de texto permiten que el usuario escriba la en-

Qué hacer cuando el programa falle

Si utiliza con frecuencia el programa Navegador puede advertir que se producirá un error en tiempo de ejecución en dos tipos específicos de situaciones. Ya que este programa ha sido creado únicamente con propósitos pedagógicos no se ha añadido el código necesario para proteger al programa contra problemas extraordinarios. Sin embargo, cuando comience a desarrollar programas más complejos deberá verificar con sumo cuidado su código para asegurarse de que no fallará en condiciones normales de uso ni en condiciones operativas extremas.

El primer problema que podrá detectar en Navegador es que el programa falla cuando el usuario selecciona en el cuadro de lista de unidades una unidad de disco que no contiene un disco, es decir, que no estará preparada para trabajar (un ejemplo de esta situación puede darse en el caso de que la unidad contenga un disquete que no esté formateado o que la unidad de red no se encuentre disponible en ese momento). Para comprobar que este fallo ocurre, verifique que la unidad A de disquetes de su computadora no contiene ningún disquete. En esta situación, seleccione con el programa Navegador la unidad A:. De forma inmediata, el programa detendrá su ejecución y Visual Basic mostrará en su pantalla el mensaje: «Error 68 en tiempo de ejecución. El dispositivo no está disponible». Este mensaje significa que Visual Basic no ha podido encontrar el disquete y detiene la ejecución del programa porque no sabe cómo proceder cuando se da esta situación. El programa Navegador exige que el usuario no cometa ningún error con los discos, hipótesis que con seguridad dejará de cumplirse en un plazo de tiempo más o menos breve.

El segundo problema se dará cuando intente ver en pantalla con Navegador el contenido de archivos de dibujo que se encuentren en la carpeta raíz. Como los archivos almacenados en esta carpeta cuentan con un nombre de ruta formado únicamente por el carácter barra invertida (\), la sentencia del programa

```
SelectedFile=File1.Path & "\" &File1.filename
```

tendrá problemas porque crea un nombre de ruta que contendrá dos barras invertidas seguidas (esta sentencia describiría al archivo coche.wmf contenido en la carpeta raíz de la unidad C como: C:\\coche.wmf). Cuando Visual Basic intente cargar un archivo cuyo nombre contenga dos barras invertidas seguidas, se provocará un error en tiempo de ejecución y el programa se detendrá.

La forma de solucionar estos problemas es utilizar sentencias de código que eviten las condiciones de error (podrá utilizar esta técnica para evitar el

(continúa)

Qué hacer cuando el programa falle *(continuación)*

problema del nombre de ruta de archivos contenidos en la carpeta raíz) o crear rutinas especiales denominadas *manejadores de error* que ayudarán a su programa a salir de una situación apurada en caso de que el problema se presente. El estudio de los manejadores de error se encuentra fuera de la intención de este comentario, pero por ahora podrá tener en cuenta que aunque Visual Basic puede manejar la mayoría de las condiciones operativas, los usuarios serán capaces de crear situaciones inmanejables que desembocarán en la interrupción del programa. Comentaremos en mayor profundidad el tema de detección y corrección de errores en los Capítulos 5 y 7.

trada, los menús cuentan con opciones que pueden ser seleccionadas y los cuadros de diálogo ofrecen una amplia variedad de elementos que pueden ser elegidos de forma individual o seleccionados en forma de grupo. En este ejercicio aprenderá a utilizar cuatro objetos de especial importancia que le ayudarán a recoger las entradas proporcionadas por el usuario en diversas situaciones. En este apartado aprenderá los pormenores de los botones de opción, de los cuadros de verificación, de los cuadros de lista y de los cuadros combo. Explorará cada uno de estos tipos de objetos desarrollando un programa al que denominaremos Compra Interactiva, la interfaz de usuario de una aplicación Internet o de cualquier otra utilidad que le permitirá adquirir de forma gráfica computadoras y equipos de oficina. Cuando ejecute el programa, verá la utilidad que tiene utilizar estos objetos de entrada. En el siguiente capítulo le mostraremos la forma en que podrá utilizar estos objetos en combinación con menús en un programa más completo.

El programa Compra Interactiva

El programa Compra Interactiva simula un entorno electrónico de pedidos en el que podrá ver los artículos que está comprando al mismo tiempo que realiza su elección. Si trabaja en una empresa que realiza un gran número de operaciones de compra de equipos quizás se sienta interesado en expandir este programa para crear un programa completo de adquisición de material con interfaz gráfica (las herramientas gráficas como ésta son muy populares en Web). A medida que vaya experimentando con Compra Interactiva deberá fijarse en la forma en que trabajan los botones de opción, los cuadros de verificación, los cuadros de lista y los cuadros combo. Todos ellos se podrán crear utilizando unos pocos pasos en Visual Basic..

Ejecución del programa Compra Interactiva

1. En el menú Archivo de Visual Basic seleccione la opción Abrir Proyecto. En su pantalla aparecerá el cuadro de diálogo Abrir Proyecto.

2. Abra el archivo compra.vdp contenido en la carpeta \Vb5Sbs\Less02.


Botón Ver Objeto

3. En la ventana Proyecto, seleccione el formulario Compra y pulse el botón Ver Objeto.

4. Cierre las ventanas Propiedades, Proyecto, Posición del formulario e Inmediato para ver el formulario Compra Interactiva por completo (no va a utilizar estas herramientas en el presente ejercicio).

   En su pantalla aparecerá el formulario de Compra Interactiva, tal y como se muestra en la siguiente figura:

El formulario Compra Interactiva contiene botones de opción, cuadros de verificación, un cuadro de lista, un cuadro combo, cuadros de imagen y un botón de orden, además de diversas etiquetas. Todos estos objetos trabajan juntos para crear un programa de pedidos, de gran simplicidad pero que muestra la forma en que trabajan los objetos de entrada de datos en Visual Basic. Cuando ejecute el programa, Compra Interactiva cargará en memoria metaarchivos Windows contenidos en la carpeta \Vb5Sbs\Less02 del disco fijo C y los mostrará en los seis cuadros de imagen del formulario.

Nota: *Si ha instalado los archivos de prácticas en una localización distinta a la carpeta C:\Vb5Sbs, que es la propuesta por defecto, las sentencias del programa que cargan los archivos gráficos contendrán un nombre de ruta incorrecto (cada una de estas instrucciones comienza con C:\Vb5Sbs\Less02, como podrá comprobar dentro de poco). Si éste es su caso, podrá hacer funcionar correctamente a su programa modificando el nombre de la carpeta que contiene estos archivos de dibujo (y asignando a esta carpeta el nombre C:\LearnVb) o modificando los nombres de ruta contenidos en el código, desde la ventana Código, utilizando las teclas de edición o la opción Reemplazar contenida en el menú Edición.*


Botón Iniciar

5. Pulse el botón Iniciar contenido en la barra de herramientas.

   El programa se ejecutará en el entorno de programación.

6. Pulse el botón de opción denominado Portátil contenido en el cuadro Computadora.

Los botones de opción permiten al usuario seleccionar un elemento de una lista.

En su pantalla aparecerá la imagen de un portátil dentro del área denominada Productos Pedidos que está situada en la parte derecha del formulario. El cuadro Computadora contiene un grupo de *botones de opción* para recoger la información proporcionada por el usuario. Los botones de opción fuerzan al usuario a elegir un (y sólo uno) elemento de una lista de posibilidades. El usuario puede pulsar de forma repetida las diferentes opciones existentes. Después de cada pulsación, la opción elegida se mostrará gráficamente en el área de pedidos de la derecha.

7. Pulse sobre los cuadros de verificación denominados Contestador, Calculadora y Fotocopiadora contenidos en el cuadro Equipos de oficina.

Los cuadros de verificación permiten que el usuario seleccione cualquier número de elementos.

Los *cuadros de verificación* se utilizan en un programa cuando se puede seleccionar simultáneamente más de una opción de una lista. Pulse de nuevo el cuadro de verificación Calculadora y observe que la imagen de la calculadora que antes se mostraba habrá desaparecido de la pantalla. Al igual que los otros elementos contenidos en la interfaz del usuario, el cuadro de verificación responde inmediatamente a las pulsaciones del usuario y las peticiones de compra se reflejan instantáneamente.

8. Pulse sobre la opción Antena contenida dentro del cuadro de lista Periféricos.

Los cuadros de lista permiten al usuario seleccionar un elemento de una lista de opciones de longitud variable.

En el área de pedidos se mostrará una imagen de una antena de recepción de datos por satélite. Los *cuadros de lista* se utilizan para obtener una única respuesta de una lista de opciones. Los cuadros de lista pueden contener muchos elementos de los que sólo se puede escoger uno simultáneamente (aparecerán barras de desplazamiento si el cuadro de lista es incapaz de mostrar simultáneamente todos los elementos que aparecen en la lista). A diferencia de los botones de opción, no es necesario que exista una opción seleccionada por defecto. En un programa desarrollado con Visual Basic se podrán añadir, eliminar u ordenar los elementos contenidos en el cuadro de lista cuando el programa se esté ejecutando.

9. A continuación, seleccione Dólares USA (lo sentimos, pero no vendemos a crédito) de la lista de pago mostrada en el cuadro combo de Método de Pago.

Los cuadros combo ocupan menos espacio que los cuadros de lista.

Los *cuadros combo*, o cuadros de lista desplegables, son similares a los cuadros de lista regulares con la única diferencia de que ocupan menos espacio. Visual Basic maneja de forma automática la apertura, cierre y desplazamiento de los elementos de los cuadros de lista combo. Todo lo que tendrá que hacer como programador es escribir el código que le permita añadir los elementos deseados en el cuadro de lista antes de ejecutar el programa y procesar la elección realizada por el usuario. Podrá ver ejemplos de cada una de estas tareas en el código del programa Compra Interactiva.

Después de llevar a cabo la selección de los pedidos, su pantalla tendrá un aspecto similar al mostrado en la siguiente figura:

10. Practique realizando unos cuantos cambios adicionales en la lista de pedidos del programa (pruebe con diferentes tipos de PC, periféricos y métodos de pago) y, finalmente, pulse el botón Salir para abandonar la ejecución del programa.

    El programa se cerrará cuando pulse Salir y volverá a aparecer en su pantalla el entorno de programación.

Análisis del código del programa Compra Interactiva

En los Capítulos 4, 5 y 6 comentaremos en detalle el código del programa.

Aunque no cuente con demasiada experiencia en codificación, es hora ya de echar un vistazo a unos cuantos procedimientos de suceso contenidos en el programa Compra Interactiva. De esta forma, verá cómo procesa el programa las entradas realizadas por el usuario utilizando los distintos elementos de la interfaz. En estos procedimientos verá las sentencias If...Then y Select Case. Le mostraré detalles acerca de éstas y otras estructuras de decisión en el Capítulo 5. Por el momento, deberá concentrarse en la propiedad Value, que resultará modificada cuando se seleccione un cuadro de verificación, y la propiedad ListIndex, cuyo valor cambia cuando se selecciona un nuevo elemento de un cuadro de lista.

Análisis del código asociado con los cuadros de verificación y con el cuadro de lista

1. Asegúrese de que el programa no está en marcha y, posteriormente, realice una doble pulsación sobre el cuadro de verificación denominado Contestador contenido dentro del cuadro Equipos de Oficina para mostrar el procedimiento Check1_Click en la ventana Código.
2. Aumente el tamaño de la ventana Código para ver el siguiente listado de código:

Cuando en el código del programa aparezca el carácter subrayado (_) al final de una línea se estará indicando que la sentencia del programa continúa en la línea siguiente.

```
Private Sub Check1_Click()
      If Check1.Value = 1 Then
        Image2.Picture = _
          LoadPicture("c:\vb5sbs\less02\answmach.wmf")
        Image2.Visible = True
      Else
        Image2.Visible = False
      End If
End Sub
```

El procedimiento de suceso Check1_Click contiene el código que se ejecutará cuando un usuario realice una pulsación con el ratón sobre el cuadro de verificación Contestador. En este caso, la palabra clave es Check1.Value, que se puede leer en nuestro idioma como «la propiedad Value del primer cuadro de verificación». Check1 es el nombre del primer cuadro de verificación contenido en el formulario; los cuadros de verificación posteriores llevan por nombre Check2, Check3, etc. La propiedad Value es la que se verá modificada cuando el usuario pulse sobre el cuadro de verificación. Cuando dentro del cuadro de verificación aparezca una «x» o una marca de verificación, la propiedad Value tendrá el valor 1; por el contrario, cuando el cuadro de verificación se encuentre vacío, la propiedad Value tendrá el valor 0 (cero).

La propiedad Value puede ser definida utilizando la ventana Propiedades durante la programación del código asociado con el cuadro de verificación (puede asignar un valor por defecto al cuadro de verificación) y, también, su valor podrá ser modificado por el usuario cuando el programa esté en ejecución (simplemente, cuando se pulse sobre un cuadro de verificación). En el código anterior, la propiedad Value es analizada mediante una estructura de decisión del tipo If...Then...Else. Si la propiedad se evalúa como 1, el programa cargará en memoria la figura de un contestador automático y la mostrará en el segundo cuadro de imagen del formulario. En caso contrario, si la propiedad Value es 0, no se mostrará en pantalla la imagen del contestador automático. Si todo esto le parece complicado, no deberá preocuparse. En el Capítulo 5 le contaré en detalle todo lo que necesita saber sobre las estructuras de decisión.

3. Cierre la ventana Código y realice una doble pulsación sobre el cuadro de lista de Periféricos contenido en el formulario.

   En la ventana Código aparecerá el código asociado con el procedimiento List1_Click. Se mostrarán las siguientes sentencias:

```
Private Sub List1_Click()
    Select Case List1.ListIndex
    Case 0
        Image3.Picture = _
          LoadPicture("c:\vb5sbs\less02\harddisk.wmf")
    Case 1
        Image3.Picture = _
          LoadPicture("c:\vb5sbs\less02\printer.wmf")
```

```
    Case 2
        Image3.Picture = _
          LoadPicture("c:\vb5sbs\less02\satedish.wmf")
    End Select
    Image3.Visible = True
End Sub
```

Resumen sobre terminología

A lo largo de este libro he utilizado varios términos que pueden resultarle poco familiares cuando describía los elementos constitutivos de un programa Visual Basic. Aunque no los he definido todos de manera formal, es preferible comentar el significado de alguno de ellos ahora para evitar cualquier confusión posterior. ¿Podría distinguirlos ahora?

Control. Un control es una herramienta que puede utilizar para crear objetos dentro de un formulario de Visual Basic. Podrá seleccionar los controles contenidos en el cuadro de herramientas y utilizarlos para dibujar objetos en un formulario utilizando, simplemente, el ratón. Podrá emplear la mayoría de los controles para crear elementos de la interfaz del usuario, tales como botones de orden, cuadros de imagen y cuadros de lista.

Objeto. Objeto es el nombre que reciben los elementos de la interfaz de usuario que puede crear utilizando los controles contenidos en el cuadro de herramientas. Podrá mover, modificar el tamaño y adaptar a sus gustos personales los objetos sin más que utilizar sus propiedades asociadas. Los objetos cuentan también con una característica conocida con el nombre de *funcionalidad inherente*: saben cómo tienen que funcionar y pueden responder a ciertas situaciones de forma «natural» (por ejemplo, un cuadro de lista «sabe» cómo tiene que desplazar los elementos contenidos en ella). Podrá programar los objetos de un programa Visual Basic utilizando procedimientos de suceso que deberán adaptarse a las diferentes situaciones a las que se puede enfrentar el programa. En Visual Basic, el propio formulario es otro objeto.

Propiedad. Una propiedad es un valor o característica que pertenece a un objeto de Visual Basic, tal como el rótulo asociado (Caption) o el color del primer plano (ForeColor). Durante el proceso de programación podrá asignar valores a estas propiedades utilizando la ventana Propiedades o bien, durante la ejecución del programa, utilizando las sentencias contenidas en el código del programa. Cuando se definen desde el código, la asignación de un valor a una propiedad tiene el siguiente formato:

```
Objeto.Propiedad = Valor
```

(continúa)

Resumen sobre terminología *(continuación)*

donde *Objeto* es el nombre del objeto que desea particularizar, *Propiedad* es la característica que desea modificar y *Valor* es el nuevo valor asignado a la propiedad. Por ejemplo, la sentencia

```
Command1.Caption = "Hola"
```

podrá ser utilizada en el código del programa para definir como «Hola» la propiedad Caption del objeto Command1.

Procedimiento de suceso. Un procedimiento de suceso es un bloque de código que se ejecutará cuando desde el programa se haga referencia o se active un determinado objeto. Por ejemplo, cuando el usuario pulse con el ratón el primer botón de orden de un programa, se pondrá en marcha el procedimiento de suceso denominado Command1_Click. Los procedimientos de suceso evaluan las condiciones y, según éstas, definen las propiedades y utilizan otras sentencias del programa para llevar a cabo la tarea asignada al programa.

Sentencia de programa. Una sentencia o instrucción de un programa es una palabra clave de código que lleva a cabo una determinada tarea. Las sentencias de Visual Basic, entre otras importantes tareas, pueden: crear espacio de almacenamiento para los datos, abrir archivos, realizar cálculos, etcétera.

Variable. Una variable es un «contenedor» especial utilizado para almacenar datos de forma temporal durante la ejecución de un programa. El programador crea variables para almacenar los resultados del cálculo, crear nombres de archivos, procesar la entrada de datos, etc. En general, una variable puede almacenar cualquier tipo de números, nombres y valores de propiedades.

Método. Un método es una sentencia especial que lleva a cabo una acción o un servicio para un objeto particular dentro de un programa. La sintaxis que deberá utilizar para definir un método dentro de un programa es la siguiente:

```
Objeto.Método Valor
```

donde *Objeto* es el nombre del objeto que desea cambiar, *Método* es el mandato que va a utilizar para modificar el objeto y *Valor* es un argumento opcional que puede utilizarse para definir el método. Por ejemplo, la sentencia

```
List1.AddItem "Cheque"
```

utiliza el método AddItem para introducir la palabra *Cheque* en el cuadro de lista denominado List1.

Cuando el usuario seleccione cualquiera de los elementos contenidos en un cuadro de lista, Visual Basic comunicará al programa el nombre del elemento en la propiedad List1.Text

El listado anterior muestra el código que se ejecutará cuando el usuario seleccione un elemento del cuadro de lista Periféricos. En este caso, la palabra clave importante es List1.ListIndex, que, en nuestro propio lenguaje, se podría leer como: «La propiedad ListIndex del primer objeto del cuadro de lista». Una vez que el usuario realice una selección en el cuadro de lista, la propiedad ListIndex devuelve un número que se corresponde con el lugar que ocupa dicho elemento dentro del cuadro de lista (el primer elemento llevará asociado el número 0, el segundo el número 1, etc.).

El texto asociado con dicho elemento (el nombre del elemento del cuadro de lista) también se incluirá en la propiedad List1.Text, y en muchas ocasiones, los programadores en Visual Basic suelen utilizar este valor en sus programas. En el código mostrado anteriormente, la estructura de decisión Select Case evalúa el valor de la propiedad List1.ListIndex, y según el valor de la misma, se cargará un metaarchivo Windows u otro. Si el valor resulta ser cero se cargará una imagen de un disco fijo; si el valor es 1 se cargará la figura de una impresora; si, por el contrario, el valor es 2 se mostrará la imagen de una antena de recepción de datos por satélite (mi periférico soñado). En el Capítulo 5 le mostraré en mayor profundidad los pormenores de la estructura de decisión Select Case.

4. Cierre la ventana Código y realice una doble pulsación en el formulario (pero no sobre ninguno de los objetos allí mostrados) al objeto de mostrar el código asociado con el propio formulario.

Las sentencias incluidas en el procedimiento de suceso denominado Form_Load se ejecutarán cuando se ponga en marcha el programa.

En la ventana Código aparecerá el código asociado con el procedimiento Form_Load. Éste es el código que se ejecutará cada vez que se ponga en marcha el programa de Compra Interactiva. Los programadores introducen en este procedimiento especial aquellas sentencias que desean ejecutar cada vez que el programa se ponga en marcha. A menudo, al igual que ocurre en el programa Compra Interactiva, estas instrucciones definen un aspecto de la interfaz de usuario que no se puede especificar utilizando los controles del cuadro de herramientas o desde la ventana Propiedades. A continuación se muestra el contenido del código Form_Load:

```
Private Sub Form_Load()
    Image1.Picture =
      LoadPicture("c:\vb5sbs\less02\pcomputr.wmf")
    List1.AddItem "Disco fijo adicional"
    List1.AddItem "Impresora"
    List1.AddItem "Antena"

    Combo1.AddItem "Dólares USA"
    Combo1.AddItem "Cheque"
    Combo1.AddItem "Libras esterlinas"
End Sub
```

La primera línea cargará en pantalla, dentro del primer cuadro de imagen, el contenido del metaarchivo Windows correspondiente a una computadora personal. Se trata del valor por defecto que se mostrará en el botón

de opción Computadora. Las siguientes tres líneas añaden elementos al cuadro de lista Periféricos (List1) del programa. Las palabras encerradas entre comillas aparecerán en el cuadro de lista. Por debajo de las sentencias relacionadas con el cuadro de lista se muestran aquellas que guardan relación con el cuadro combo denominado Método de Pago (Combo1). La palabra clave asociada con estos dos grupos es AddItem, que es una función especial, o método, válida para los cuadros de lista y combo.

Un *método* es una sentencia especial que realiza una acción o un servicio para un objeto determinado, tal como añadir elementos a un cuadro de lista. Los métodos se diferencian de las propiedades (que contienen un valor) y de los procedimientos de suceso (que se ejecutan cuando un usuario manipula un objeto). Los métodos también pueden ser compartidos entre objetos, por lo que cuando aprenda a utilizar un método también será capaz de aplicar estos conocimientos en otras circunstancias distintas. Comentaremos el funcionamiento de diversos métodos de uso frecuente a lo largo de este libro.

A estas alturas, ya ha terminado con el programa Compra Interactiva. Dedique unos minutos para analizar otras partes del programa y, finalmente, comience con en el siguiente ejercicio.

EMPLEO DE UN OBJETO OLE PARA PONER EN MARCHA APLICACIONES

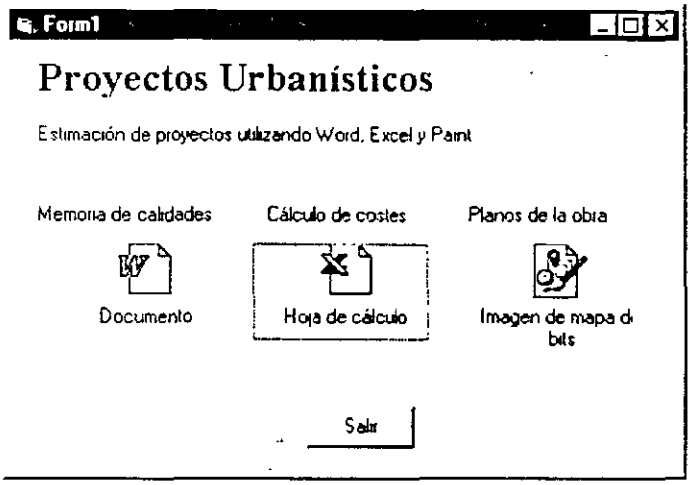
Un objeto OLE le permitirá poner en marcha aplicaciones para Windows desde una aplicación de Visual Basic.

Una de las características más importantes de Visual Basic es su capacidad para trabajar de manera muy estrecha con otras aplicaciones desarrolladas para Windows. Al utilizar un objeto OLE podrá ejecutar aplicaciones desde su propio programa a la vez que estará ejecutando y procesando varios tipos de información. También podrá utilizar un objeto OLE para poner en marcha componentes individuales de otras aplicaciones (tales como el corrector ortográfico de Microsoft Word) utilizando una tecnología especial denominada *Automatización* (formalmente conocida como Automatización OLE).

Analizaremos en profundidad los objetos OLE y la Automatización OLE en el Capítulo 10. En el siguiente ejercicio conocerá la forma en que trabajan los objetos OLE y cómo pueden utilizarse (sin código de programa) para crear una aplicación denominada Proyectos Urbanísticos que pondrá en marcha los programas Word, Excel y Paint para que el usuario pueda introducir información relacionada con este tema y desarrollar dibujos para un proyecto de construcción.

Para ejecutar el programa Proyectos Urbanísticos necesitará contar con una copia de Word, Excel y Paint 6 instaladas en su disco fijo (Paint está incluido en Microsoft Windows 95). Cuando cree un objeto OLE, se mostrará un cuadro de diálogo denominado Insertar Objeto que listará los objetos disponibles y que podrá utilizar en su programa. Si no cuenta con los programas Word, Excel y Paint

instalados en su disco fijo, el cuadro de diálogo Insertar Objeto no los listará, pero podrá seleccionar cualquier otro objeto contenido en dicha lista (el propósito de este ejercicio es practicar en el empleo de aplicaciones que desea ejecutar desde un programa de Visual Basic). Cuando ejecute la utilidad una vez terminada tendrá el siguiente aspecto:



Creación del programa Proyectos Urbanísticos

1. En el menú Archivo seleccione Nuevo Proyecto y pulse Aceptar para crear un archivo .exe estándar.

   Se cerrará el programa Compra Interactiva y en el entorno de programación se mostrará un formulario vacío.

2. En la esquina superior izquierda del formulario, cree un rótulo que muestre el título «Proyectos Urbanísticos». Debajo de éste cree un segundo rótulo con el contenido: «Estimación de proyectos utilizando Word, Excel y Paint».

   Deje algo de espacio extra en el primer rótulo, ya que posteriormente le pediré que aumente el tamaño del tipo de letra utilizado en el mismo, modificando alguna de sus propiedades asociadas.

3. Debajo de la segunda etiqueta, cree tres rótulos adicionales equiespaciados que tengan el siguiente contenido: «Memoria de calidades», «Cálculo de costes» y «Planos de la obra» (consulte la figura anterior).

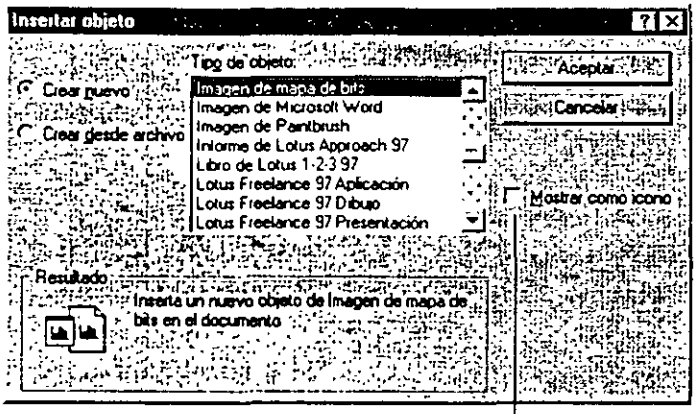
   El objetivo de estos tres rótulos es identificar los objetos OLE utilizados para poner en marcha las aplicaciones Word, Excel y Paint, respectivamente. A continuación le mostraré cómo añadir los objetos OLE al formulario.


Control OLE

4. Pulse el control OLE contenido en el cuadro de herramientas.

5. Debajo del rótulo Memoria de calidades cree un rectángulo que tenga el tamaño de una caja de cerillas, utilizando el control OLE.

   Cuando suelte el botón del ratón aparecerá en su pantalla el cuadro de diálogo Insertar Objeto, tal y como se muestra un poco más abajo, que contendrá una lista de todas las aplicaciones que podrá utilizar en su programa. La lista exacta variará de PC a PC.



6. Desplace hacia abajo la lista de objetos y seleccione el elemento Documento de Microsoft Word, si es que tiene instalado en su sistema el programa Word para Windows 95.

   Si no es así, seleccione dentro de este cuadro de diálogo una versión anterior de Word u otro procesador de texto basado en Windows o una aplicación similar.

7. Pulse el cuadro de verificación Mostrar como icono contenido dentro del cuadro Insertar Objeto con el objetivo de que la aplicación aparezca como un icono en el programa de Visual Basic que está desarrollando.

   Si no pulsa este cuadro de verificación el objeto de la aplicación (normalmente, un documento) se mostrará en una ventana independiente. El empleo de esta característica le proporcionará importantes ventajas, como veremos posteriormente en este libro. Por ahora, sin embargo, seleccione el cuadro Mostrar como icono.

8. Pulse Aceptar para cerrar el cuadro de diálogo Insertar Objeto y para abrir Word.

   Se ejecutará el programa Word y se mostrará un documento vacío dentro de este procesador de texto. Este documento se convertirá en una *plantilla* dentro del programa Proyectos Urbanísticos.

   Podrá contener cualquier información que un contratista pueda encontrar de utilidad cuando maneje el programa, tales como detalles sobre la compañía constructora, nombres, direcciones, precios, materiales, etc.

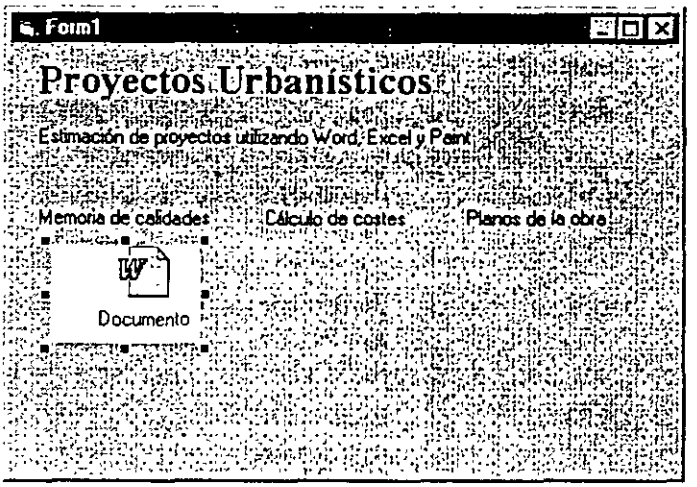
9. Por ahora, sólo deberá introducir el rótulo «**Notas estimativas**». A continuación, seleccione la opción Fecha y Hora del menú Insertar para añadir a la plantilla la fecha actual del sistema.

   El texto aparecerá posteriormente en el programa tal y como ahora lo introduzca en Word.

10. En el menú Archivo de Word seleccione la opción Salir.

    Si el programa le pregunta si desea actualizar el documento fuente pulse el botón Sí. Este mensaje de aviso aparecerá en su pantalla cada vez que cierre ciertos objetos de aplicación.

    Una vez que haya acabado de introducir el primer objeto OLE, su formulario tendrá un aspecto similar al mostrado en la figura siguiente. Modifique el tamaño del objeto OLE o de la etiqueta de texto si existe algún solapamiento entre ambos objetos.



Nota: El aspecto tridimensional del objeto OLE está controlado por la propiedad Appearance, que puede adoptar los valores 3D y Flat (plano). En el punto 13 le mostraré cómo asignar el valor Flat a la propiedad Appearance de cada uno de los objetos OLE introducidos, así como el valor gris claro a la propiedad BackColor (color de fondo).

11. Repita los pasos 4 a 10 para añadir en el formulario el objeto Microsoft Excel (o su equivalente) justo debajo de la etiqueta Cálculo de costes y un objeto imagen del tipo mapa de bits debajo de la etiqueta Planos de obra.

    Asegúrese de seleccionar el cuadro de verificación Mostrar como icono dentro del cuadro de diálogo Insertar Objeto en ambas ocasiones. Si maneja con soltura ambos programas puede añadir alguna información a la plantilla (por ejemplo, notas o instrucciones) tanto en la hoja de trabajo de Excel como en la hoja de dibujo de Paint. En el caso de la hoja de trabajo de Excel es fácil imaginar que si usted es un contratista ordenado introducirá varias filas y columnas de información contractual, como pue-

den ser gastos de energía, materiales y mano de obra. La gran ventaja de utilizar otras aplicaciones basadas en Windows dentro de su programa es que podrá acceder de forma automática a todas las funciones disponibles en dichas aplicaciones, ¡no tendrá que reinventar la rueda!

12. Inserte un botón de orden en la parte inferior del formulario. Una vez que haya añadido dicho botón, realice una doble pulsación sobre dicho objeto e introduzca la sentencia **End** en el procedimiento de suceso Command1_Click.

    La sentencia End hará que el programa finalice cuando el usuario pulse este botón de orden.

13. Defina las siguientes propiedades para los objetos incluidos en el formulario utilizando la ventana Propiedades:

Objeto	Propiedad	Valor
Command1	Caption	«Salir»
Label1	Font	Times New Roman Negrita, 18-puntos
OLE1	BorderStyle Appearance BackColor	0-None 0-Flat Gris claro
OLE2	BorderStyle Appearance BackColor	0-None 0-Flat Gris claro
OLE3	BorderStyle Appearance BackColor	0-None 0-Flat Gris claro

14. En el menú Archivo seleccione la opción Guardar Proyecto como y almacene en el disco el formulario con el nombre **MiOleProy**. Del mismo modo, grabe el proyecto en el disco bajo el nombre **MiOleProy.**

    Una vez que haya terminado, su formulario MiOleProy deberá tener un aspecto semejante al que se muestra en la figura de la página siguiente.

Si quiere ver el contenido completo del programa Proyecto Urbanístico, éste se encuentra localizado en el disco en la carpeta Chapter 2.

A continuación, ejecute el programa para ver cómo funcionan los objetos OLE introducidos.

Ejecución del programa Mi Proyecto Urbanístico

1. Pulse el botón Iniciar contenido en la barra de herramientas.

   El programa se ejecutará dentro del entorno de programación. El objeto OLE 1 (el icono Documento) estará rodeado de una línea punteada, indicando que cuenta con la atención o *foco* del programa.

Nota: El foco es importante para las operaciones que se realizan a través del teclado. Cuando el usuario pulse la tecla INTRO el objeto que se ejecutará será el que se encuentre seleccionado o activado (foco) en el programa. El usuario podrá activar o cambiar el foco a cualquier otro objeto sin más que pulsar la tecla TAB o pulsando con el ratón dicho objeto. Podrá modificar el orden en que los objetos se muestran activados por defecto en el programa modificando la propiedad TabIndex asociada a cada uno de los objetos.

2. Realice una doble pulsación sobre el icono Documento en el programa.
   El procesador de textos se pondrá en marcha y en su pantalla aparecerá la plantilla de documento creada anteriormente en Word.

3. Escriba unas cuantas líneas de texto (imagínese que usted es un importante constructor) y, finalmente, abra el menú Archivo y seleccione la opción Salir para volver al programa MiOleProy.

4. Realice una doble pulsación sobre el icono Hoja de cálculo del programa.
   La hoja de cálculo se pondrá en marcha y en una ventana aparecerá la plantilla creada anteriormente con Excel.

5. Introduzca algunas filas y columnas de datos dentro de la hoja de cálculo (pruebe a utilizar las funciones típicas de Excel y las funciones de formato si así lo desea) y, a continuación, en el menú Archivo, pulse Salir para volver al programa MiOleProy.

6. Realice una doble pulsación sobre el icono Imagen de Paint.
   El programa Paint se ejecutará y aparecerá en una ventana. Paint es un programa de dibujo de gran sencillez de manejo que contiene herramientas diversas y paletas de color que le permitirán crear imágenes sin demasiada complicación.

7. Cree un pequeño bosquejo de un plano de construcción con este programa (inténtelo al menos) y en el menú Archivo seleccione la opción Salir y volver a.
8. Pulse el botón Salir para finalizar el programa.

¡Felicidades! Acaba de construir el primer programa que utiliza los objetos de aplicación de Microsoft Office. Podrá utilizar esta técnica para incluir en un programa cualquier objeto de aplicación que tenga instalado en su PC. A continuación, trabajaremos con otro tipo de archivos dentro del entorno de Windows: una base de datos preexistente que contiene los nombres y las direcciones de sus clientes.

EMPLEO DE UN OBJETO DE DATOS PARA CONSULTAR UNA BASE DE DATOS DE MICROSOFT ACCESS

Si trabaja en una empresa en la que comparte información con otros usuarios informáticos seguro que empleará bases de datos para controlar la información relacionada con sus clientes, empleados o con los proyectos que están en marcha. Una *base de datos* es un conjunto organizado de información que se almacena de forma electrónica en un archivo. Las aplicaciones de bases de datos tales como Microsoft Access, dBASE y Paradox son programas especiales que crean y procesan la información almacenada en bases de datos. Estos programas cuentan con las herramientas que le permitirán diseñar la base de datos, manipular la información almacenada en ella y buscar datos específicos. Para mejorar su trabajo diario con las bases de datos, Visual Basic proporciona tres objetos que le permitirán mostrar y modificar la información contenida en los archivos de la base de datos. El objeto principal, los datos, le permitirán acceder directamente desde el formulario a los campos y registros de una base de datos. Practique ahora utilizando un objeto de datos para mostrar la información contenida en la base de datos de Access denominada Compras.mdb.

Nota: El archivo Access utilizado en el presente ejercicio se encuentra en su disco fijo en la carpeta Less02, por lo que podrá practicar con el presente ejercicio aunque no tenga instalado Access en su PC. Asimismo, si así lo desea, podrá utilizar su propio archivo de base de datos en lugar de emplear el mencionado en estas líneas.

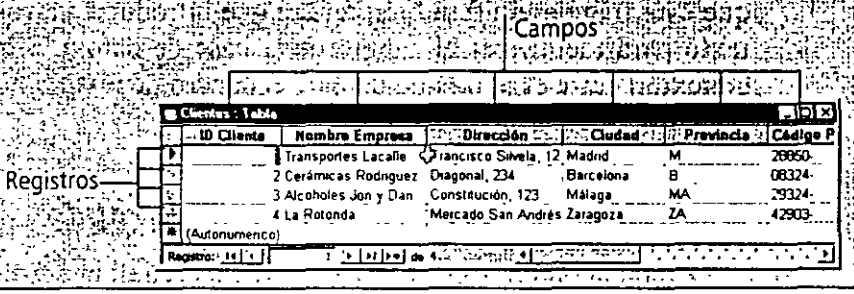
Creación de un objeto de datos

1. En el menú Archivo seleccione la opción Nuevo proyecto y pulse Aceptar para crear un archivo .exe estándar.

   Se cerrará el programa MiOleProy y aparecerá un nuevo formulario en el entorno de programación. Almacene los cambios realizados en el programa MiOleProy si Visual Basic así se lo indica.
2. Pulse el control Data contenido en el cuadro de herramientas.

Campos y registros

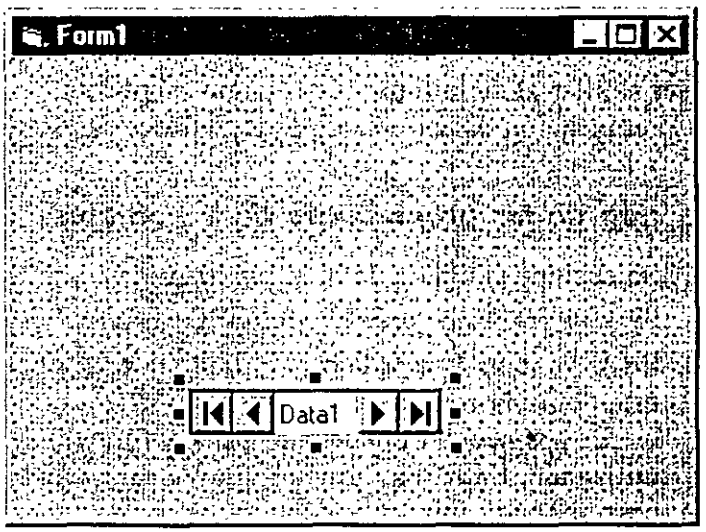
En las bases de datos se utilizan dos términos de especial relevancia relacionados con la información almacenada en ella: *campos* y *registros*. Los campos son los tipos de información almacenados en una base de datos. Los campos típicos de una base de datos de clientes pueden ser sus nombres, direcciones, números telefónicos y comentarios acerca de ellos. Toda la información relacionada con un determinado cliente o negocio se denomina *registro*. En las bases de datos personales cada una de las tarjetas que contiene información sobre un empleado determinado recibe el nombre de registro. Cuando un programador crea una base de datos la información se almacena en tablas formadas por registros y campos. Típicamente, los registros se corresponden con filas en la tabla y los campos son las columnas de la misma.



Control Data

3. Desplace el puntero del ratón al centro del formulario, cerca de su borde inferior, y dibuje un cuadro rectangular utilizando el control.

   En el formulario aparecerá un objeto de datos denominado Data1, tal y como se muestra en la figura siguiente.



El objeto contiene flechas que le permitirán desplazarse por los registros de su base de datos cuando el programa esté en ejecución. El objeto contiene también un rótulo (Data1) que podrá utilizar para describir la base de datos a la que accederá utilizando este objeto. Normalmente, este objeto tendrá el mismo nombre que la base de datos a la que apunta. Las flechas más cercanas a los bordes exteriores del objeto se utilizan para desplazarse al principio o al final de la base de datos.

Son muchas las operaciones sofisticadas que podrá llevar a cabo con una base de datos en Visual Basic. En este ejercicio mostrará el campo Nombre de la base de datos Compras.mdb (en realidad, será capaz de desplazarse por toda la base de datos y ver todos los nombres contenidos en este archivo). Para mostrar el campo Nombre en el formulario, necesitará introducir un objeto adicional que será el encargado de almacenar los datos. Como los datos que queremos mostrar en esta ocasión son rótulos de texto, deberá añadir un cuadro de texto al formulario con el que está trabajando (también tendrá que introducir un rótulo cercano al cuadro de texto para identificar el campo de la base de datos). Finalmente, establecerá una conexión entre el objeto de datos y el cuadro de texto. Para ello, tendrá que asignar nuevos valores a algunas propiedades.

Creación de los cuadros de texto y de rótulo

Control TextBox

1. Pulse el control TextBox contenido en el cuadro de herramientas.
2. Cree un cuadro de texto en el formulario, por encima del objeto dato.

   El cuadro de texto deberá tener el mismo tamaño que el objeto dato. El cuadro deberá tener la suficiente anchura como para mostrar nombres de hasta 20 caracteres de longitud.
3. Pulse el control Label contenido en el cuadro de herramientas.
4. Cree un rótulo a la izquierda del cuadro de texto contenido en el formulario.

Control Label

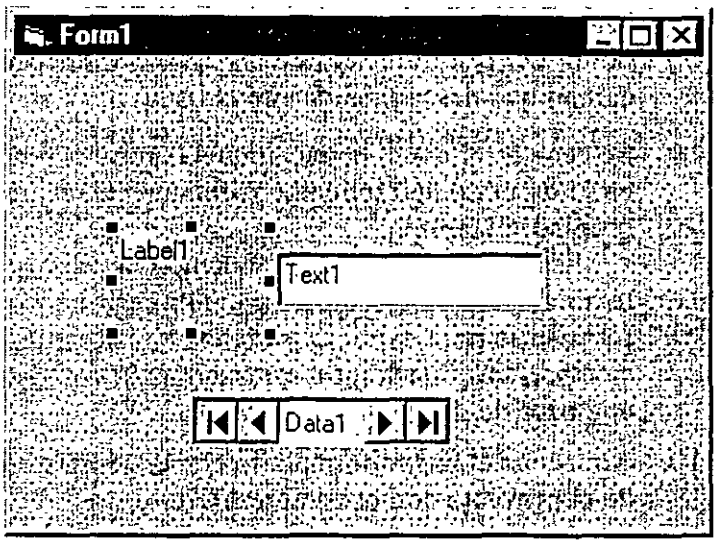
Cuando termine de crear los objetos nombrados, el aspecto del formulario deberá ser similar al de la figura de la página siguiente.

A continuación definiremos las propiedades de estos tres objetos.

Definición de las propiedades de los objetos

1. Seleccione el objeto datos y, posteriormente, pulse sobre el botón Ventana de Propiedades contenido en la barra de herramientas.
2. En la ventana Propiedades verifique que la propiedad Connect (conectar) está definida como Access (valor por defecto).

   La propiedad Connect almacena el formato de la base de datos que esté utilizando. Entre los formatos que Visual Basic puede leer se incluyen Access, dBASE, FoxPro y Paradox.



3. En la ventana Propiedades, defina la propiedad DatabaseName como C:\Vb5Sbs\Less02\Compras.mdb seleccionando este archivo dentro del cuadro de diálogo denominado DatabaseName.

   Compras.mdb es la base de datos ejemplo desarrollada en Access con la que va a trabajar en el presente ejercicio. Contiene nombres, direcciones, números de teléfonos y otra información utilizada para realizar el seguimiento de un grupo de tiendas ficticias.

4. En la ventana Propiedades seleccione la propiedad RecordSource y, seguidamente, pulse sobre el botón del cuadro de lista desplegable. Cuando aparezca una lista de las tablas contenidas en la base de datos, seleccione la opción Clientes.

   La propiedad RecordSource le permitirá especificar la tabla (colección de datos) de la base de datos con la que desea trabajar.

5. En la ventana Propiedades, asigne a la propiedad Caption el valor «Compras.mdb».

   El rótulo contenido en el objeto mostrará ahora el valor Compras.mdb para identificar a la base de datos que está siendo utilizada. A continuación le mostraré cómo modificar la propiedad DataSource del cuadro de texto para enlazar el cuadro de texto con el objeto datos.

6. Seleccione el objeto cuadro de texto y, a continuación, pulse el botón Ventana Propiedades contenido en la barra de herramientas.

7. En la ventana Propiedades, seleccione la propiedad DataSource, pulse el botón del cuadro de lista desplegable y seleccione Data1 (el primer objeto de datos) de la lista.

8. En la ventana Propiedades seleccione la propiedad DataField, pulse el botón del cuadro de lista desplegable y seleccione Nombre empresa (el campo que desea mostrar) de la lista.

9. Seleccione ahora el objeto etiqueta, pulse el botón Propiedades contenido en la barra de herramientas y, finalmente, asigne el valor «Nombre» a la propiedad Caption.

   Este rótulo identificará al campo de la base de datos en el cuadro de texto cuando se ejecute el programa. El hecho de añadir etiquetas a su formulario para explicar qué elementos se encuentran allí contenidos es, siempre, una buena idea, especialmente si está trabajando con campos de una base de datos.

10. Almacene el nuevo formulario con el nombre **MiDatos**. Asimismo, guarde el proyecto con el nombre **MiDatos**.

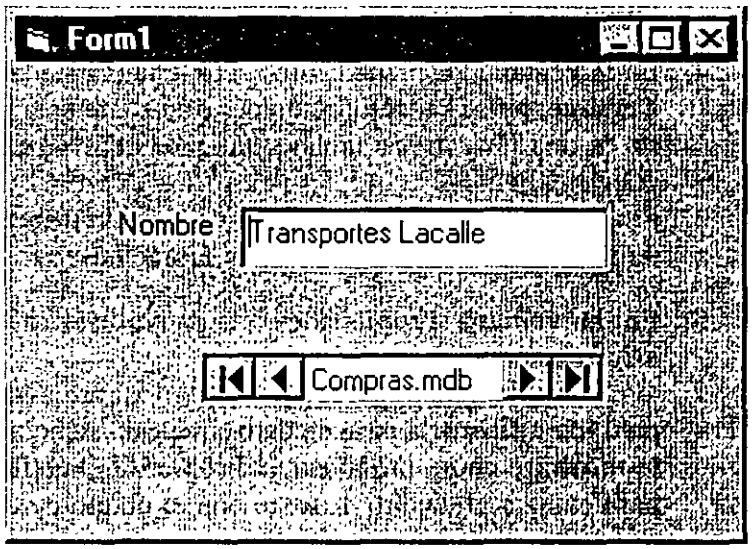
¡Eso es todo! Ahora, sólo le restará ejecutar el programa.

Ejecución del programa MiDatos

El programa Datos.vbp completo se encuentra en el disco en la carpeta \Vb5Sbs\Less02.

1. Pulse el botón Iniciar contenido en la barra de herramientas.

   El programa se ejecutará en el entorno de programación, tal y como se muestra en el figura siguiente:



Visual Basic carga la base de datos Compras.mdb e introduce el primer campo, denominado Nombre Empresa, en el cuadro de texto. Podrá examinar otros datos almacenados en este campo pulsando los botones del objeto de datos.

2. Pulse la flecha interior derecha del objeto de datos.
   En la ventana de texto aparecerá el segundo nombre contenido en la base de datos.
3. Pulse el botón exterior derecho del objeto de datos.
   Visual Basic mostrará el último nombre de campo de la base de datos.
4. Pulse el botón exterior de la parte izquierda del objeto de datos.
   Visual Basic muestra de nuevo el primer nombre contenido en la base de datos.
5. Pulse el botón Cerrar del formulario para detener el programa.

CÓMO SE MODIFICA UNA BASE DE DATOS

Un objeto de datos también le permitirá modificar la información contenida en una base de datos. Para cambiar un nombre contenido en Compras.mdb deberá ejecutar el programa MiDatos y desplazarse hasta localizar el nombre que desee modificar. A continuación, pulse en el cuadro de texto Nombre e introduzca el rótulo que desee. Cuando pase a otro registro, el nombre que haya introducido se copiará de forma inmediata en la base de datos original. Vamos a comprobar este extremo.

Cómo cambiar un nombre de la base de datos

1. Pulse el botón Iniciar contenido en la barra de herramientas para ejecutar el programa MiDatos.
   En el cuadro de texto aparecerá el primer nombre contenido en la base de datos.
2. Resalte el primer nombre utilizando el ratón, pulse la tecla SUPR y escriba **Autobuses Pedro**.
3. Pulse la flecha interior de la parte derecha del objeto de datos para desplazarse hasta el siguiente registro.
   El primer nombre de la base de datos será ahora Autobuses Pedro.
4. Pulse la flecha interior de la parte izquierda del objeto de datos para volver a mostrar el primer registro.
   El nombre que aparece ahora en pantalla es Autobuses Pedro.
5. Pulse el botón Cerrar del formulario para detener el programa.

Como puede ver, los objetos de datos le permiten acceder con facilidad a bases de datos preexistentes. Podrá mostrar cualquier campo contenido en la base de datos y procesar a su gusto la información contenida en ella. En el Capítulo 11 aprenderá más detalles sobre los objetos de datos y cómo gestionar archivos.

UN PASO MÁS ALLÁ: INSTALACIÓN DE CONTROLES ACTIVEX

Podrá extender las funciones asociadas a Visual Basic instalando los controles ActiveX incluidos en Visual Basic o los controles ActiveX desarrollado por otros fabricantes de programas. Podrá instalar los controles personalizados en el cuadro de herramientas para un proyecto determinado utilizando la opción Componentes contenida en el menú Proyecto. Los nuevos controles ActiveX que se aprovechan de la tecnología OLE de 32 bits (un estándar de Microsoft para objetos programables contenidos en aplicaciones, sistemas operativos y herramientas de Internet) tienen la extensión de archivo .ocx y se pueden utilizar con el fin de añadir nuevas funciones a los programas desarrollados para Windows 95 y Windows NT. Los controles personalizados más antiguos, que fueron diseñados para versiones de 16 bits de Visual Basic y Windows 3.1, tienen como extensión del nombre de archivo la terminación .vbx. Siempre que pueda, deberá utilizar los nuevos controles .ocx. En muchos casos, Visual Basic sustituirá automáticamente los viejos controles por sus nuevos homónimos cuando instale los nuevos controles.

Instalación de los controles ActiveX denominados Grid y CommonDialog

La edición de Aprendizaje de Visual Basic 5 incluye algunos controles ActiveX adicionales que podrá utilizar en sus proyectos. Por ejemplo, si está escribiendo un programa para mostrar datos en una tabla, podrá instalar el control Grid (rejilla), que podrá utilizar para introducir en un formulario una rejilla de celdas (un objeto rejilla tiene un aspecto similar a una hoja de trabajo de Excel). Este control ActiveX se encuentra localizado en el archivo Grid32.ocx. Otro control ActiveX que suele ser de mucha utilidad para la creación de cuadros de diálogo estándar, como Abrir y Guardar como, es el control CommonDialog, contenido en el archivo Comdlg32.ocx.

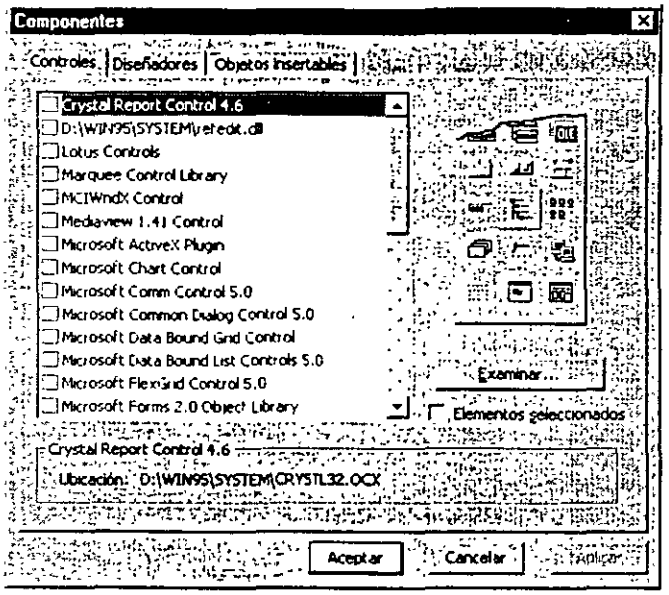
Ejecute los siguientes pasos para instalar los controles ActiveX:

1. En el menú Archivo seleccione la opción Nuevo Proyecto y pulse Aceptar para crear un archivo .exe estándar.

2. En el menú Proyecto seleccione la opción Componentes y, finalmente, pulse sobre la etiqueta Controles.

   En su pantalla aparecerá el cuadro de diálogo denominado Componentes, tal y como se muestra en la figura de la página siguiente.

   En este cuadro de diálogo se muestra una lista ordenada alfabéticamente de los controles ActiveX instalados en su sistema y que podrá añadir al cuadro de herramientas de su proyecto. Para proporcionarle una mayor flexibilidad a la hora de desarrollar programas, cada proyecto tendrá su



propio y exclusivo cuadro de herramientas, que contendrá los controles mostrados por defecto en Visual Basic y cualquier otro control ActiveX que usted haya podido seleccionar. Según lo que acabamos de decir, cualquier control que incluya en el proyecto con el que esté trabajando sólo aparecerá, en principio, en el cuadro de herramientas de dicho proyecto. En los siguientes pasos le mostraré cómo añadir el control Grid (Grid32.ocx) y el control CommonDialog (Comdlg32.ocx) a su cuadro de herramientas.

Truco: El cuadro de diálogo Componentes contiene una etiqueta denominada Objetos insertables que podrá utilizar para añadir objetos de aplicaciones a su caja de herramientas del proyecto. Un objeto insertable *es un componente del programa suministrado por otra aplicación para Windows, tal como un documento de Word o una hoja de trabajo de Excel. Estas herramientas le parecerán tan útiles como los demás controles ActiveX.*

---

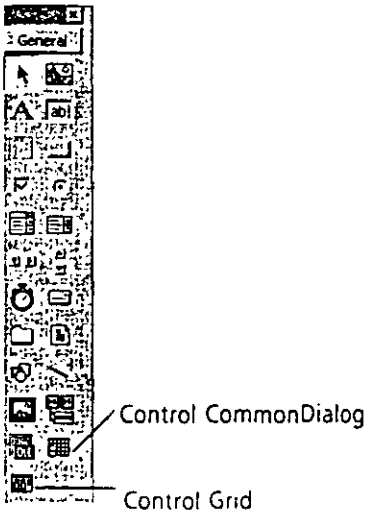
3. Pulse la casilla de verificación situada cerca del control denominado Microsoft Common Dialog Control 5.0.

   Se seleccionará el control ActiveX y la localización del archivo .ocx aparecerá en la parte inferior del cuadro de diálogo.

4. Pulse sobre la casilla de verificación situada a la izquierda del control denominado Microsoft Grid Control.

5. Pulse el botón Aceptar para que los controles ActiveX seleccionados se incluyan a partir de ahora en el cuadro de herramientas del proyecto.

El cuadro de herramientas mostrará dos nuevos controles, tal y como se muestra en la siguiente figura:



Los controles Grid y CommonDialog trabajarán exactamente igual que el resto de los controles contenidos en el cuadro de herramientas de Visual Basic. De hecho, si no supiera *a priori* que se trata de controles personalizados sería difícil que pudiera distinguirlos del resto de controles. Podrá seleccionar los controles ActiveX sin más que pulsar sobre ellos y podrá utilizarlos para crear e introducir objetos en un formulario en la misma forma en que emplea los otros controles. Los controles ActiveX cuentan también con propiedades cuyos valores asociados pueden ser modificados y también pueden ser utilizados en el código del programa como el resto de los controles que ha manejado en este capítulo.

Controles ActiveX de Windows 95

Si está desarrollando programas que se ejecutarán posteriormente sobre Windows 95 o Windows NT 4.0, las ediciones Profesional y Empresarial de Visual Basic 5 incluyen una biblioteca especial de controles ActiveX que pueden ser utilizados en el diseño de la interfaz del usuario para aplicaciones compatibles con Windows 95. La siguiente tabla proporciona una breve descripción de estos controles ActiveX, la mayoría de los cuales pueden activarse utilizando la opción Microsoft Windows Common Controls 5.0 contenida en el cuadro de diálogo Componentes. Si desea añadir estas funciones avanzadas en sus programas, deberá utilizar las ediciones Empresarial o Profesional de Visual Basic y, de esta forma, podrá ahorrarse tiempo de desarrollo.

(continúa)

Contoles ActiveX de Windows 95 *(continuación)*

Control ActiveX	Función
RichTextBox	Es un control TextBox mejorado que proporciona funciones de formato adicionales.
ListView	Muestra listas de archivos y de iconos en la misma forma en que el Explorador de Windows los muestra.
TreeView	Muestra la información de una forma jerárquica (por ejemplo, carpetas y subcarpetas) en la misma forma en que lo hace el Explorador de Windows.
Tabstrip	Crea cuadros de diálogo que cuentan con etiquetas al estilo Windows 95.
Toolbar	Añade barras de herramientas estándares a los programas.
Slider	Añade un control deslizante al programa, permitiendo que el usuario pueda cambiar con facilidad y rapidez la posición del control en un archivo o en una presentación.
ProgressBar	Muestra al usuario cuánto le queda al programa para completar la tarea.
StatusBar	Presenta información periférica al usuario en un formato estándar.
ImageList	Contiene una colección de figuras que suministra a otros controles.
UpDown	Muestra flechas arriba y abajo que permitirán incrementar o decrementar valores.
Animation	Permite la visualización de un video AVI.

Si desea continuar con el siguiente capítulo

➤ No cierre el programa Visual Basic y pase al Capítulo 3.
Si el programa le pregunta si desea almacenar los cambios del proyecto actual, conteste No.

Si desea salir de Visual Basic por ahora

➤ En el menú Archivo seleccione Salir.
Si en su pantalla se muestra un cuadro de diálogo Guardar, pulse No. No necesita almacenar este proyecto y su lista de controles ActiveX.

RESUMEN DEL CAPÍTULO

Para	Haga esto	Botón
Crear un cuadro de texto	Pulse el control TextBox y dibuje el cuadro.	
Crear un botón de orden	Pulse el control CommandButton y dibuje el botón.	
Crear una propiedad en tiempo de ejecución	Modifique el valor de la propiedad utilizando código de programa. Por ejemplo: `Text1.Text = "¡Hola!"`	
Crear un cuadro de lista de unidades	Seleccione el control DriveListBox y dibuje el cuadro.	
Crear un cuadro de lista de directorios	Seleccione el control DirListBox y dibuje el cuadro.	
Crear un cuadro de lista de archivos	Pulse sobre el control FileListBox y dibuje el cuadro.	
Evitar que un programa falle	Escribir un procedimiento de suceso utilizando código de programación (consulte el Capítulo 7).	

Cargar una imagen en tiempo de ejecución	Llamar a la función LoadPicture y asignar el resultado a la propiedad Picture de un objeto imagen o a un objeto cuadro de imagen. La sintaxis de esta sentencia es la siguiente:
	`Objeto.Picture=LoadPicture ArchivoSeleccionado)`
	donde *Objeto* es el nombre del objeto y *ArchivoSeleccionado* es una variable que almacena el nombre del archivo gráfico. Por ejemplo:
	`Archivo = "c:\camion.bmp"`
	`Imagel.Picture = LoadPicture(Archivo)`	

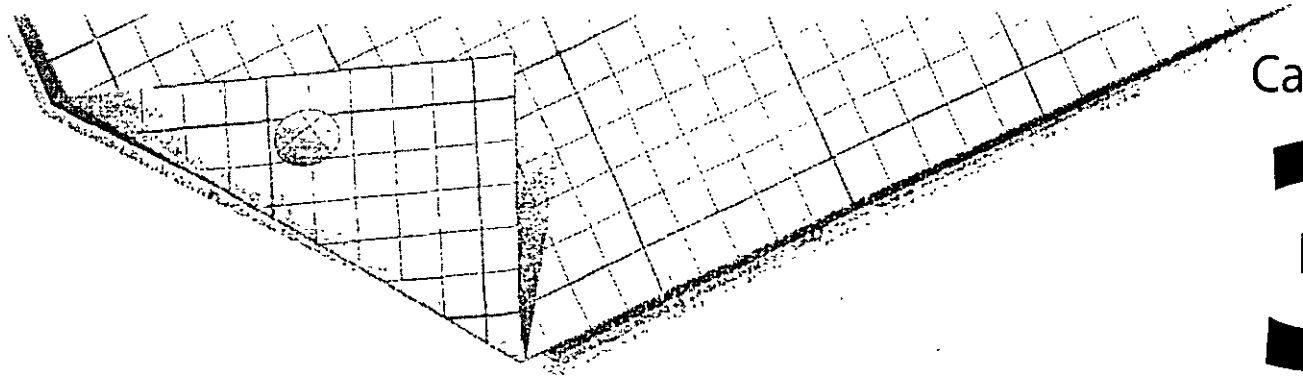
Creación de un botón de opción	Utilice el control OptionButton y dibuje el botón de opción. Para crear varios botones de opción, introduzca más de un botón de opción dentro de una caja que haya creado utilizando el control Frame.	
Creación de un cuadro de verificación	Seleccione el control CheckBox y dibuje un cuadro de verificación.	
Creación de un cuadro de lista	Pulse el control ListBox y dibuje un cuadro de lista.	

Para	Haga esto	Botón
Creación de un cuadro de lista desplegable	Seleccione el control ComboBox y dibuje un cuadro de lista desplegable.	
Adición de elementos a un cuadro de lista	Incluya sentencias con el método AddItem en el procedimiento Form_Load de su programa. Por ejemplo: `List1.AddItem "Impresora"`	
Ejecute aplicaciones Windows desde su programa	Seleccione el control OLE y dibuje un cuadro para introducir la aplicación (o una imagen de ella) en su formulario. Seleccione la aplicación deseada en el cuadro de diálogo Insertar Objeto.	
Cómo mostrar bases de datos existentes desde su programa	Seleccione el control Data y cree un objeto que le permita navegar por la base de datos. Conecte el objeto de datos a un objeto que pueda mostrar los registros de la base de datos (normalmente, un objeto de cuadro de texto).	
Modificar los registros contenidos en una base de datos	Muestre la base de datos en el programa. Edite el registro mostrado en el cuadro de texto en tiempo de ejecución y pulse una de las flechas de navegación para almacenar los cambios en el disco.	
Instalar controles ActiveX	En el menú Proyecto seleccione la opción Componentes y pulse la etiqueta Controles. Seleccione los controles ActiveX que desee añadir al cuadro de herramientas del proyecto y, finalmente, pulse el botón Aceptar.	

Para obtener información sobre	Haga esto
Controles de Visual Basic	En el menú Ayuda de Visual Basic, seleccione la opción Temas de Ayuda de Microsoft Visual Basic, pulse la etiqueta Índice, escriba el nombre del control y pulse Mostrar.

AVANCE DEL SIGUIENTE CAPÍTULO

En el siguiente Capítulo, denominado «Manejo de menús y cuadros de diálogo», aprenderá a añadir menús a sus programas y a procesar la información introducida por el usuario a través de los cuadros de diálogo. También aprenderá a utilizar el control CommonDialog para crear cuadros de diálogo que le permitirán gestionar las entradas realizadas por el usuario.



Capítulo

3

Manejo de menús y cuadros de diálogo

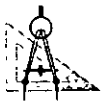
En el Capítulo 2 ha aprendido el manejo de diferentes objetos de Microsoft Visual Basic cuyo objetivo es recoger datos aportados por el usuario cuando éste está utilizando el programa. En este capítulo le mostraré la forma de presentar opciones al usuario utilizando menús y cuadros de diálogo de aspecto profesional. Los *menús* están localizados en la barra de menús y contienen listas de opciones que guardan entre sí alguna relación. Cuando se pulsa sobre el nombre de un menú se despliega un cuadro de lista en el que aparecerán una serie de opciones. La mayoría de las opciones de los menús se ejecutan inmediatamente después de ser pulsados; por ejemplo, cuando el usuario selecciona la opción Copiar del menú Edición, la información seleccionada se copiará inmediatamente en el Portapapeles de Windows. Sin embargo, si el nombre de una opción del menú está seguido de puntos suspensivos, en lugar de ejecutarse automáticamente, en su pantalla aparecerá un cuadro de diálogo solicitando más información al usuario antes de que la opción se ejecute. En este capítulo aprenderá a utilizar el Editor de menús y el control CommonDialog, que le permitirá añadir a sus programas menús y cuadros de diálogos estándar.

INSERCIÓN DE NUEVOS MENÚS UTILIZANDO EL EDITOR DE MENÚS

El Editor de menús es una herramienta gráfica que gestiona los menús contenidos en sus programas. Con este editor podrá añadir nuevos menús, modificar y reordenar los existentes y suprimir los más antiguos y obsoletos.

También le permitirá añadir efectos especiales a sus menús, tales como teclas

de acceso, marcas de verificación y atajos de teclado. Una vez que haya añadid menús al formulario, podrá utilizar procedimientos de suceso para procesar las opciones contenidas en ellos. En el siguiente ejercicio utilizará el Editor de Menú para crear un menú Reloj que contendrá una serie de mandatos que le permitirán mostrar la fecha y la hora actuales.



Creación de un menú

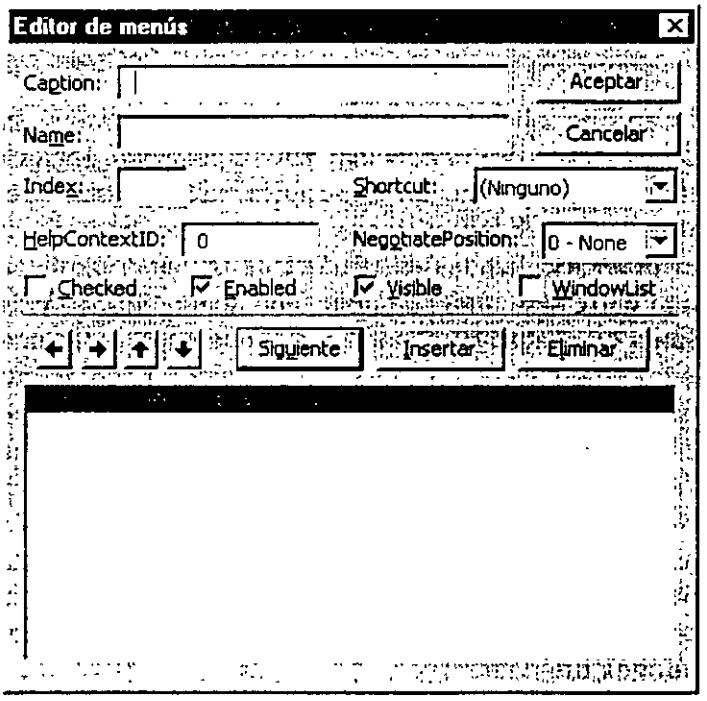
1. Ponga en marcha Visual Basic.

   Si el entorno de programación se encuentra ya en marcha, seleccione la opción Nuevo Proyecto del menú Archivo y comience un archivo .exe estándar.

Botón Editor de menús

2. Pulse el botón Editor de menús contenido en la barra de herramientas.

   En su pantalla aparecerá el Editor de menús, tal y como se muestra en la siguiente figura:



El Editor de menús le ayudará a crear y a modificar los menús.

El Editor de menús mostrará en un cuadro de diálogo las distintas opciones disponibles para construir menús. Deberá especificar el título del menú (el nombre del menú que aparecerá en la pantalla) en el cuadro de texto denominado Caption. También tendrá que introducir el nombre del menú (el nombre que tendrá en el código del programa) dentro del cuadro de texto Name (Nombre). Estos son los dos atributos más importantes de

un menú. Otros parámetros tales como Index, HelpContextID, Shortcut y Checked son opcionales. Al final de este capítulo obtendrá más detalles sobre la definición de accesos directos (Shortcut).

Cuando pulse el botón Siguiente contenido en el cuadro de diálogo Editor de Menús se borrarán todas las asignaciones realizadas y estará en condiciones de definir la siguiente opción de menú. El cuadro de lista de menú que aparece en la parte inferior del cuadro de diálogo muestra los elementos a medida que los vaya creando y visualiza también la estructura global del menú. Utilice ahora el Editor de menús para crear un menú denominado Reloj.

3. Escriba **Reloj** en el cuadro de texto Caption y pulse la tecla TAB.

   La palabra *Reloj* será la que se utilice como el nombre del primer menú y el cursor se desplazará al cuadro de texto denominado Name. Cuando introduzca el título del menú, éste aparecerá también en el cuadro de lista de menús que aparece más abajo.

4. Escriba **mnuReloj** en el cuadro de texto Name.

   Por convenio, las siglas mnu se utilizan para identificar a los menús.

   La palabra *mnuReloj* se introducirá en su programa como el nombre del menú que acaba de definir. Por convenio, se utiliza el prefijo *mnu* para identificar las opciones de menú dentro del código del programa. Al utilizar esta técnica, es decir, al emplear un prefijo de tres caracteres con los elementos pertenecientes a la interfaz de usuario, podrá diferenciar con facilidad los procedimientos de suceso cuando sus programas se hagan más complicados. Del mismo modo, de esta forma podrá identificar los elementos de la interface en la ventana Código.

Nota: En el Capítulo 8, en el apartado denominado «Un paso más allá», podrá encontrar una lista de los convenios de denominación para todos los objetos de Visual Basic.

---

5. Pulse el botón Siguiente para añadir el título Reloj a su programa.

   El menú Reloj se añadirá a la barra de menús y el Editor de menús borrará toda la información mostrada en el cuadro de diálogo para que usted pueda comenzar a introducir los datos relacionados con el siguiente elemento. El título del menú seguirá apareciendo en el cuadro de lista de menús situado en la parte inferior del cuadro de diálogo. A medida que vaya generando sus menús, cada nuevo elemento se irá añadiendo al cuadro de lista de menús para que pueda contemplar en cualquier momento la estructura del sistema de menús que está construyendo.

6. Escriba **Fecha** en el cuadro de texto Caption, pulse la tecla TAB y escriba **mnuFechaItem**.

   El mandato Fecha aparecerá en el cuadro de lista de menús. Como su deseo es hacer que Fecha se convierta en una opción de menú más que en el nombre de un nuevo menú, deberá utilizar un convenio de denominación

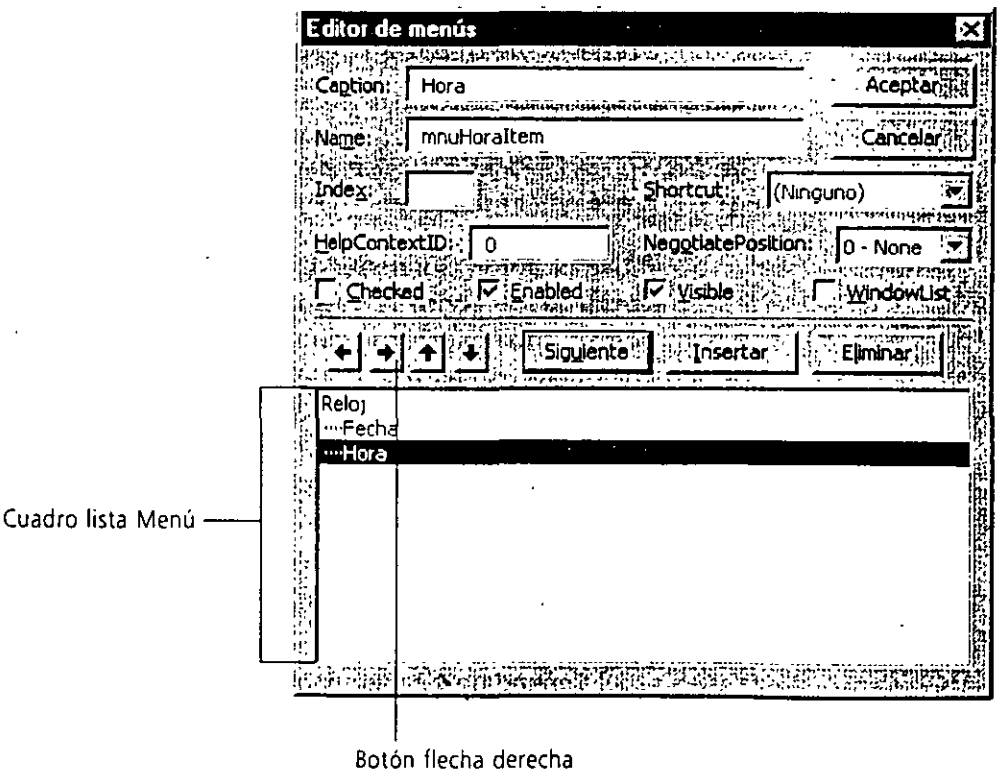
diferente. Por eso, ha añadido el sufijo *Item* (elemento) al final del nombre dentro del cuadro de texto Name. Esta forma de proceder le ayudará a diferenciar los mandatos de los títulos de los menús dentro de la ventana Código.

7. Una vez que tenga resaltado el elemento Fecha en el cuadro de lista de menús, pulse el botón que contiene una flecha hacia la derecha dentro del Editor de menús.

   La opción Fecha se desplaza un tabulador hacia la derecha (cuatro espacios) dentro del cuadro de lista de menús, indicando que este elemento es una opción de un menú. La posición de un elemento dentro de este cuadro de lista determinará si se trata de un título o nombre de menú (pegado a la izquierda), un mandato de menú (un tabulador a la derecha), un título de submenú (dos tabuladores) o una opción de submenú (tres tabuladores). Podrá pulsar el botón flecha derecha contenido en el cuadro de diálogo Editor de menús para desplazar los elementos hacia la derecha, mientras que el botón flecha izquierda le servirá para lo contrario (desplazar los elementos hacia la izquierda). A continuación le mostraré cómo añadir una nueva opción de menú, denominada Hora, en el menú Reloj.

8. Pulse el botón Siguiente, escriba **Hora**, pulse TAB y escriba **mnuHoraItem**.

   La opción Hora aparecerá en el cuadro de lista menú, tal y como se muestra a continuación:



Observe que el Editor de menús ha supuesto que el siguiente elemento es también una opción de menú y lo ha desplazado cuatro espacios hacia la derecha. Por el momento ha finalizado de añadir opciones al menú Reloj y le mostraré cómo cerrar el Editor de menús.

9. Pulse Aceptar para cerrar el Editor de menús.

   El Editor de menús se cierra y su formulario volverá a aparecer en el entorno de programación mostrando una barra de menús y un menú denominado Reloj. Abramos ahora este menú para poder contemplar su contenido.

10. Despliegue el menú Reloj.

    Los mandatos Fecha y Hora aparecen en él.

    Al pulsar sobre una opción de menú en el entorno de programación se mostrará en dicha ventana el procedimiento de suceso que se pondrá en marcha. Un poco más tarde le mostraré cómo crear procedimientos de suceso para los mandatos Fecha y Hora. En primer lugar, añadiremos soporte de teclado a los menús.

11. Pulse sobre el formulario (o pulse la tecla ESC) para cerrar el menú Reloj.

Cómo asociar teclas de acceso a los mandatos de los menús

Podrá definir una tecla de acceso anteponiendo el carácter ampersand (&) a la letra deseada.

Visual Basic facilita la definición de teclas de acceso para el empleo de menús y opciones. Se denomina tecla de acceso a la tecla que podrá pulsar el usuario para ejecutar un determinado mandato. Cuando durante la ejecución del programa el usuario despliegue el menú, la tecla de acceso de una opción aparecerá como una letra subrayada en el nombre de dicha opción. Para asociar una tecla de acceso a una opción de menú, todo lo que tendrá que hacer es volver a abrir el Editor de menú y anteponer a la letra que quiere utilizar como tecla de acceso el carácter ampersand (&). A partir de ese momento su programa admitirá dicha tecla de acceso.

Convenios de menú

Por convenio, cada menú y mandato contenido en los menús de cualquier aplicación ejecutable en Microsoft Windows comienzan por una letra en mayúscula. Archivo y Edición suelen ser los dos primeros nombres contenidos en la barra de menús, mientras que Ayuda suele ser el último. Otros nombres de menús bastante frecuentes son: Ver, Formato y Ventana. No importa qué menús y opciones utilice o introduzca en sus programas, lo importante es que su significado sea claro y su distribución consistente. Los menús y sus mandatos tienen que ser, por definición, fáciles de utilizar y, para mayor comodidad del usuario, deben guardar una estrecha relación con los menús y opciones que aparecen en otros programas. Cuando cree opciones de menús tenga en cuenta, siempre, las siguientes normas básicas:

(continúa)

Convenios de menú *(continuación)*

- Utilice títulos cortos y específicos que consten de una palabra o, como máximo, de dos.
- Asigne a cada opción de un menú una única tecla de acceso. Utilice la primera letra del nombre de la opción, siempre que sea posible.
- Si una opción se utiliza como un interruptor activado/desactivado, coloque una marca de verificación cerca del elemento cuando éste se encuentre activo. Podrá añadir una marca de verificación pulsando el cuadro de verificación Checked contenido en el Editor de menús o definiendo la propiedad Checked de dicha opción de menú como True.
- Introduzca unos puntos suspensivos (...) después de las opciones de menú que requieran que el usuario introduzca más información antes de que dichos mandatos se puedan ejecutar completamente.
- Utilice los convenios de denominación de menús, tales como el prefijo *mnu* y el sufijo *Item*, cuando esté asignando nombres a sus menús y opciones.

Le mostraré cómo añadir una tecla de acceso al menú Reloj.

Definición de una tecla de acceso

Botón Editor de menús

1. Pulse el botón Editor de menús contenido en la barra de herramientas.
   En su pantalla aparecerá el cuadro de diálogo asociado con el Editor de menús; en el cuadro de lista de menús se mostrarán las distintas opciones de menús definidas anteriormente en el programa. En el cuadro de diálogo aparecen los atributos Caption (título) y Name (nombre) del menú Reloj.
2. Sitúe el puntero del ratón justo delante de la palabra *Reloj* contenida en el cuadro de texto Caption y pulse el botón del ratón.
   El cursor parpadeará delante de la letra «R» de *Reloj*.
3. Introduzca el carácter & para definir a la letra «R» como la tecla de acceso para el menú Reloj.
   En el cuadro de texto aparece el carácter ampersand.
4. Seleccione la opción Fecha en la lista de menús.
   En el cuadro de diálogo aparecerán los valores para los atributos Caption y Name correspondientes a la opción Fecha.
5. Introduzca un ampersand delante de la letra «F» dentro del cuadro de texto Caption.
   A partir de ahora la letra «F» quedará definida como la tecla de acceso para la opción Fecha.
6. Seleccione la opción Hora de la lista de menús.

En el cuadro de diálogo aparecerán los valores para los atributos Caption y Name correspondientes a la opción Hora.

7. Introduzca un ampersand delante de la letra «H» dentro del cuadro de texto Caption.

   A partir de ahora la letra «H» quedará definida como la tecla de acceso para la opción Hora.

8. Pulse el botón Aceptar para cerrar el Editor de menús.

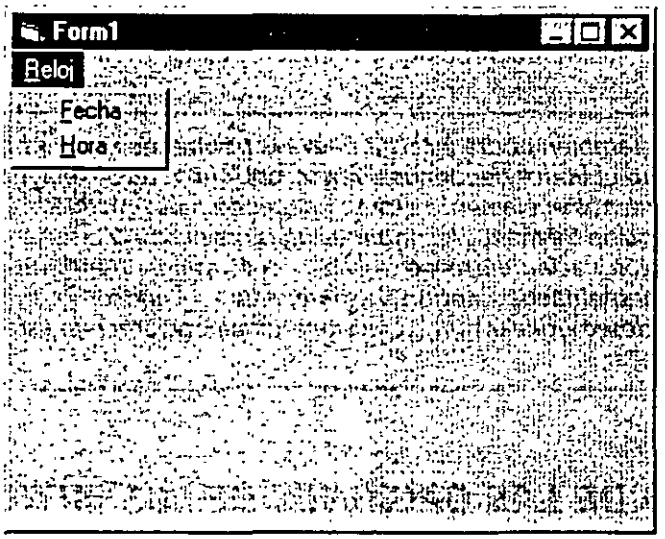
   A continuación, vamos a ejecutar el programa para ver cómo aparecen las distintas teclas de acceso en el programa.

9. Pulse el botón Iniciar.

Botón Iniciar

10. Pulse sobre el menú Reloj.

    En su pantalla se mostrará el menú Reloj y los mandatos contenidos en él. Las teclas de acceso aparecerán subrayadas.



11. Pulse el botón Terminar para salir del programa.

    Ahora podrá practicar utilizando el Editor de menús para modificar el orden en que las opciones Fecha y Hora aparecen en el menú Reloj. Cambiar el orden en que aparecen las opciones dentro de un menú es una posibilidad que puede llegar a ser de gran importancia; en ocasiones, resulta más sencillo tomar ciertas decisiones cuando la estructura del menú ya ha sido completada.

Botón Terminar

Cambio de orden de las opciones de los menús

1. Pulse el botón Editor de menús contenido en la barra de herramientas.

   En su pantalla aparecerá el cuadro de diálogo Editor de menús.

2. Seleccione la opción Hora contenida en la lista de menús.
   En el cuadro de diálogo aparecerán los valores correspondientes a los atributos Caption y Name de la opción Hora.

3. Pulse el botón flecha arriba contenido en el cuadro de diálogo.
   La opción de menú Hora se colocará delante de la opción Fecha.

4. Pulse el botón Aceptar.
   Se cerrará el Editor de menús y, como puede comprobar fácilmente, el orden en que se muestran las opciones Fecha y Hora dentro del menú Reloj ha cambiado. También podrá utilizar el botón flecha abajo dentro del Editor de menús para variar el orden en que se muestran las opciones de un determinado menú; este botón desplazará un lugar hacia abajo la opción de menú seleccionada.

En este momento, ya ha terminado de crear la interfaz de usuario para el menú Reloj. A continuación le mostraré cómo utilizar los procedimientos de suceso para procesar las selecciones que realice el usuario en el programa.

Nota: También podrá insertar nuevas opciones de menú y borrar elementos no deseados utilizando el Editor de menús. Para insertar un nuevo elemento de menú, seleccione en la lista de menús el elemento que se encuentre más próximo al lugar donde desee insertar la nueva opción de menú. A continuación pulse el botón Insertar. El Editor de menús insertará una opción vacía dentro de la lista. Podrá asignarla atributos rellenando los cuadros d texto denominados Caption y Name, que ya conoce. Para borrar una opción de menu que no desee utilizar más, seleccione dicha opción en la lista de menús y pulse el botón Eliminar.

PROCESAMIENTO DE LAS OPCIONES DE LOS MENÚS

Cada opción de menú será procesada por un procedimiento de suceso asociado con dicha opción

Una vez que los nombres de los menús aparecen en la barra de menús, se convierten en objetos del programa. Para que estos objetos realicen un determinado trabajo necesitará desarrollar procedimientos de suceso para cada uno de ellos. Normalmente, los procedimientos de suceso asociados con menús contienen sentencias de programa que muestran o procesan la información contenida en el formulario y modifican una o más propiedades del menú. Si para poder ejecutar la opción de menú seleccionada por el usuario necesita que éste introduzca más información tendrá que desarrollar un procedimiento de suceso que muestre un cuadro de diálogo. Para ello, podrá utilizar un objeto de diálogo común o un objeto de entrada.

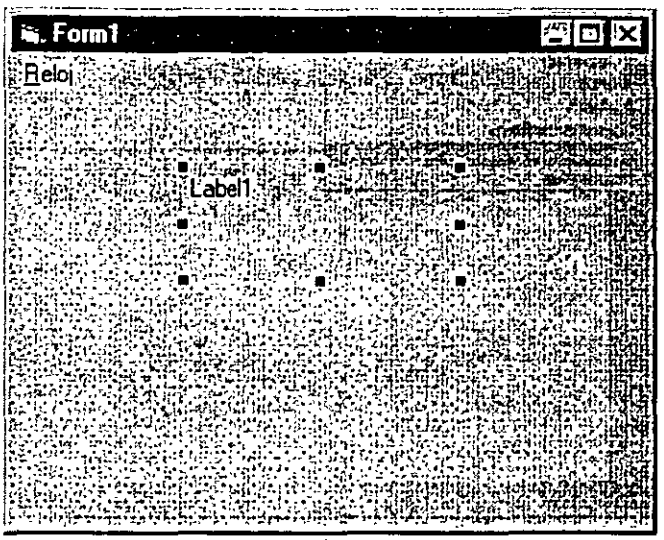
En el siguiente ejercicio le mostraré cómo añadir un objeto de etiqueta a su formulario para mostrar la salida de las opciones Fecha y Hora contenidas en el menú Reloj.

Inclusión de un objeto de etiqueta en el formulario

Control Label

1. Pulse el control Label contenido en la caja de herramientas.
2. Cree un pequeño rótulo en la parte superior central del formulario.

   La etiqueta aparecerá en el formulario. Recibirá el nombre de Label1 en el código del programa. Su pantalla tendrá un aspecto similar a la mostrada a continuación:



Podrá definir las propiedades asociadas a Label1 utilizando la ventana Propiedades

3. Defina las siguientes propiedades para esta etiqueta:

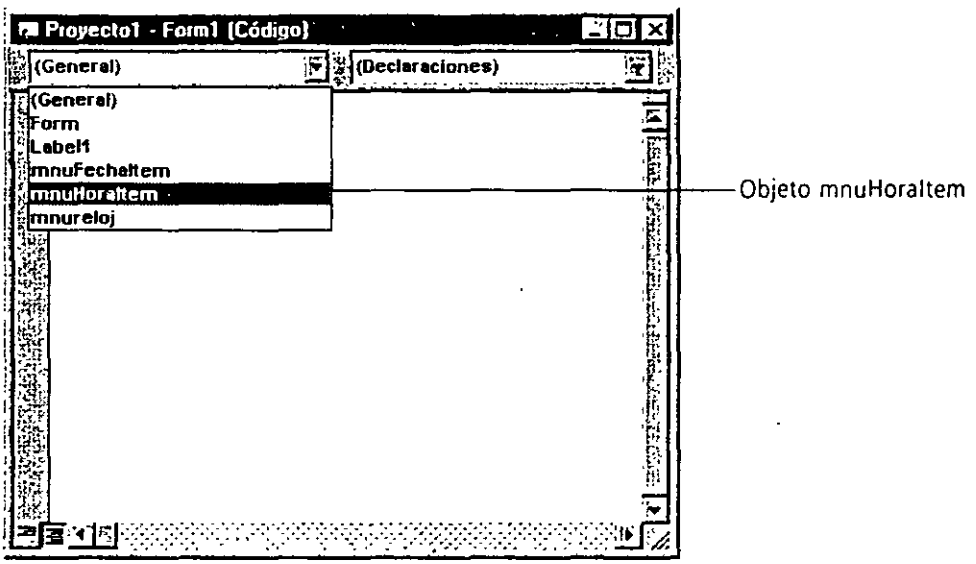
Objeto	Propiedad	Valor
Label1	Alignment	2-Center
	Border Style	1-Fixed Single
	Caption	(Vacío)
	Font	MS Sans Serif, Negrita, 14-puntos

A continuación le mostraré cómo añadir sentencias de programa a los procedimientos de suceso asociados con las opciones Hora y Fecha.

Nota: *En los siguientes ejercicios podrá introducir código de programa para procesar las opciones de los menús. Escriba las sentencias de programa tal y como se muestran aquí. Todavía no le mostraré cómo funcionan las sentencias del programa, la intención es que conozca cómo se utilizan para dar soporte a una interfaz de usuario plenamente funcional. En los Capítulos 4 a 6 aprenderá la forma en que trabajan las sentencias del programa.*

Edición de los procedimientos de suceso asociados con los menús

1. Pulse el botón Ver código contenido en la ventana Proyecto para abrir la ventana Código.
2. Pulse el cuadro de lista desplegable Objeto y seleccione mnuHoraItem.



En la ventana Código aparecerá el procedimiento de suceso mnuHoraItem_Click. Si recuerda, asignó el nombre mnuHoraItem a la opción Hora dentro del Editor de menús. Cuando el usuario seleccione la opción Hora en el programa, se ejecutará el procedimiento de suceso mnuHoraItem_Click.

3. Pulse cuatro veces la BARRA ESPACIADORA y escriba:

```
Label1.Caption=Time
```

Esta instrucción del programa mostrará la hora actual (obtenida del reloj del sistema) dentro de la etiqueta asociada con el objeto Label1, reemplazando la etiqueta mostrada anteriormente en este objeto. Podrá utilizar la función Time en cualquier momento para mostrar la hora con una exactitud de segundos.

Nota: La función Time de Visual Basic devuelve la hora actual del reloj del sistema. Podrá definir esta hora utilizando la opción Fecha y Hora del Panel de Control de Windows. Del mismo modo, podrá modificar el formato en que se presenta la hora de su sistema utilizando la opción Configuración Regional del Panel de Control.

---

4. Pulse la tecla FLECHA ABAJO.

   Visual Basic interpretará la línea e introducirá mayúsculas y espacios en blanco si fuera necesario (Visual Basic verifica el contenido de cada una de las líneas de código introducidas en busca de errores de sintaxis. Podrá introducir una línea pulsando cualquiera de las siguientes teclas: INTRO, FLECHA ARRIBA o FLECHA ABAJO).

5. Pulse el objeto mnuFechaItem contenido en el cuadro de lista desplegable denominado Objeto.

   En la ventana de código aparece el procedimiento de suceso denominado mnuFechaItem_Click. Este procedimiento de suceso se ejecutará cada vez que el usuario seleccione la opción Fecha del menú Reloj.

6. Pulse cuatro veces la BARRA ESPACIADORA y escriba:

```
Label1.Caption=Date
```

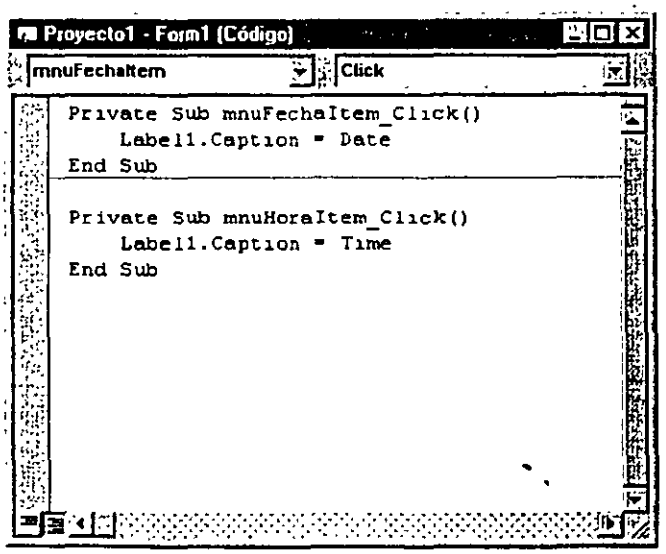
Esta instrucción del programa mostrará la fecha actual (obtenida del reloj del sistema) dentro de la etiqueta asociada con el objeto Label1, reemplazando la etiqueta mostrada anteriormente en este objeto. Podrá utilizar la función Date en sus programas para mostrar en cualquier momento la fecha de su sistema. Asigne Date a un objeto etiqueta siempre que quiera mostrar la fecha actual en su formulario.

Nota: *La función Date de Visual Basic devuelve la fecha actual del reloj del sistema. Podrá definir esta fecha utilizando la opción Fecha y Hora del Panel de Control de Windows. Del mismo modo, podrá modificar el formato en que se presenta la fecha de su sistema utilizando la opción Configuración Regional del Panel de Control.*

---

7. Pulse la tecla FLECHA ABAJO para introducir la línea.

   Su pantalla tendrá un aspecto similar al siguiente:



8. Cierre la ventana Código.

   Con esto ha terminado de introducir el programa de demostración de menús. Ahora, deberá almacenar en el disco el formulario y el proyecto utilizando el nombre de MiMenú.

Grabación del programa MiMenú

Botón Guardar proyecto

1. Pulse el botón Guardar proyecto contenido en la barra de herramientas.

   El botón Guardar proyecto es la alternativa al mandato Guardar Proyecto del menú Archivo.

2. Para asignar un nombre al formulario del proyecto seleccione la carpeta \Vb5Sbs\Less03, escriba **MiMenú** y pulse INTRO.

   El formulario se almacenará en el disco con el nombre MiMenú.frm. A continuación, aparecerá en su pantalla el cuadro de diálogo denominado Guardar proyecto como.

3. Para asignar un nombre al proyecto, escriba **MiMenú** y pulse INTRO.

   El proyecto se almacenará en el disco con el nombre de MiMenú.vbp.

En estos momentos, su programa está listo para funcionar.

Ejecución del programa MiMenú

Botón Iniciar

1. Pulse el botón Iniciar contenido en la barra de herramientas.

   El programa MiMenú se ejecutará en el entorno de programación.

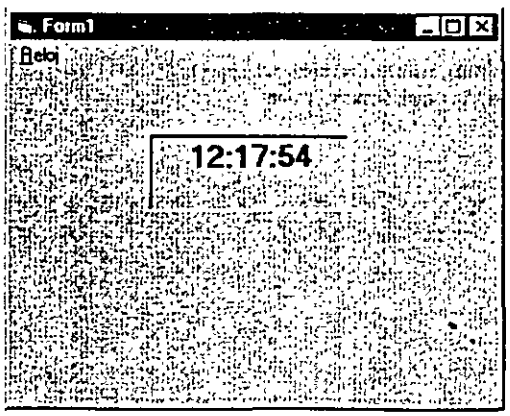
2. Despliegue el menú Reloj contenido en la barra de menús.

   Aparecerán las opciones del menú Reloj.

El código completo del programa Menú.vdp se encuentra en la carpeta \Vb5Sbs\Less03 del disco.

3. Ejecute el mandato Hora.

   En el cuadro de etiqueta se mostrará la hora del sistema, tal y como puede ver en la figura siguiente:



Probablemente, la hora mostrada en su caso será diferente.

A continuación, probaremos a mostrar la fecha actual utilizando las teclas de acceso.

4. Pulse la tecla ALT.
   Se resaltará el nombre Reloj de la barra de menús.
5. Pulse R para mostrar el contenido del menú Reloj.
   El contenido de este menú aparecerá en pantalla.
6. Pulse F para mostrar la fecha actual.
   La fecha actual aparecerá en el cuadro de etiqueta.
7. Pulse el botón Terminar para detener el programa.

Botón Terminar

¡Enhorabuena! Ahora cuenta con un programa que utiliza menús y teclas de acceso. En el siguiente ejercicio aprenderá a utilizar menús para mostrar cuadros de diálogo estándar.

Funciones del reloj del sistema

Podrá utilizar 10 funciones distintas para obtener valores temporales del reloj de su computadora. Con estos valores podrá introducir en sus programas calendarios particularizados, relojes y alarmas. En la siguiente tabla se lista el conjunto completo de las funciones del reloj del sistema. Si desea obtener más información consulte el sistema de ayuda interactiva de Visual Basic.

Función	Descripción
Time	Devuelve la hora actual del reloj del sistema.
Date	Devuelve la fecha actual tomada del reloj del sistema.
Now	Devuelve un valor codificado que representa la hora y la fecha actual. Esta función suele ser muy útil como argumento para otras funciones del reloj del sistema.
Hour (*hora*)	Devuelve el número de hora correspondiente a la hora especificada (0 a 23).
Minute (*hora*)	Devuelve el número del minuto correspondiente a la hora especificada (0 a 59).
Second (*hora*)	Devuelve el número del segundo correspondiente a la hora especificada (0 a 59).
Day (*fecha*)	Devuelve el número del día correspondiente a la fecha especificada (1 a 31).

(continúa)

Funciones del reloj del sistema *(continuación)*

Función	Descripción
Month *(fecha)*	Devuelve el número del mes correspondiente a la fecha especificada (1 a 12).
Year *(fecha)*	Devuelve el número del año correspondiente a la fecha especificada.
Weekday *(fecha)*	Devuelve un número que representa el día de la semana (1 para el Domingo, 2 para el Lunes, etc.).

EMPLEO DE OBJETOS DE DIÁLOGO COMÚN

Los objetos de diálogo comunes le permitirán mostrar cinco cuadros de diálogo estándar en sus programas. Cada cuadro de diálogo común podrá mostrarse desde un único objeto de diálogo común. Podrá mostrar un determinado cuadro de diálogo común utilizando su correspondiente método de objeto de diálogo común (como se mencionó anteriormente, un método es un mandato que realiza una acción o un servicio para un objeto). Podrá controlar los contenidos de un cuadro de diálogo común definiendo sus propiedades asociadas. Cuando el usuario rellene un cuadro de diálogo común en un programa, los resultados se devuelven utilizando una o más propiedades asociadas con el objeto de diálogo común, que a su vez podrán ser utilizadas en el programa para llevar a cabo la tarea que tengan encomendada.

En la tabla que se muestra a continuación se listan los cinco cuadros de diálogo comunes proporcionados por el objeto de diálogo común y los métodos que deberá utilizar para especificarlos.

Cuadro de diálogo	Propósito	Método
Abrir	Proporciona la unidad de disco, nombre de carpeta y nombre de archivo para un archivo existente.	ShowOpen
Guardar como	Proporciona el nombre de unidad, de carpeta y de archivo para un nuevo archivo.	ShowSave
Imprimir	Permite la definición de opciones de impresión.	ShowPrinter
Fuente	Permite que el usuario seleccione una nueva fuente y estilo.	ShowFont
Color	Permite que el usuario seleccione un nuevo color de una paleta.	ShowColor

También tendrá que asignar un nuevo nombre al formulario cuando desee dar un nuevo nombre al proyecto. Es aconsejable que guarde en primer lugar el formulario.

En los siguientes ejercicios le mostraré cómo añadir un nuevo menú al programa MiMenú y practicará con los cuadros de diálogo comunes denominados Abrir y

Color. Si desea conservar una copia del programa MiMenú original, antes de empezar deberá guardar el formulario y el proyecto MiMenú con el nombre MiDiálogo.

Grabación de los archivos MiMenú con el nombre MiDiálogo

1. Si el proyecto MiMenú.vdp no se encuentra abierto, cárguelo desde el disco sin más que ejecutar el mandato Abrir proyecto del menú Archivo.
   Si no creó el programa MiMenú, abra el proyecto Menú.vbp contenido en la carpeta \Vb5Sbs\Less03. Su contenido deberá coincidir con el mostrado en MiMenú.
2. En el menú Archivo seleccione el mandato Grabar MiMenú.frm como.
   En su pantalla aparecerá el cuadro de diálogo Guardar Archivo como.
3. Especifique la carpeta \Vb5Sbs\Less03, escriba **MiDiálogo.frm** y pulse INTRO.
   En el disco fijo se almacenará una copia del formulario MiMenú bajo el nombre de MiDiálogo.frm.

Aviso: Si en primer lugar no almacena el formulario con un nuevo nombre, los programas MiMenú y MiDiálogo compartirán el mismo formulario.

---

4. En el menú Archivo, ejecute el mandato Guardar Proyecto como.
   En su pantalla aparecerá el cuadro de diálogo Guardar Proyecto como.
5. Especifique la carpeta \Vb5Sbs\Less03, escriba **MiDiálogo.VBP** y pulse INTRO.
   En el disco fijo se almacenará una copia del proyecto MiMenú bajo el nombre de MiDiálogo.vbp.

Inserción de un objeto de diálogo común

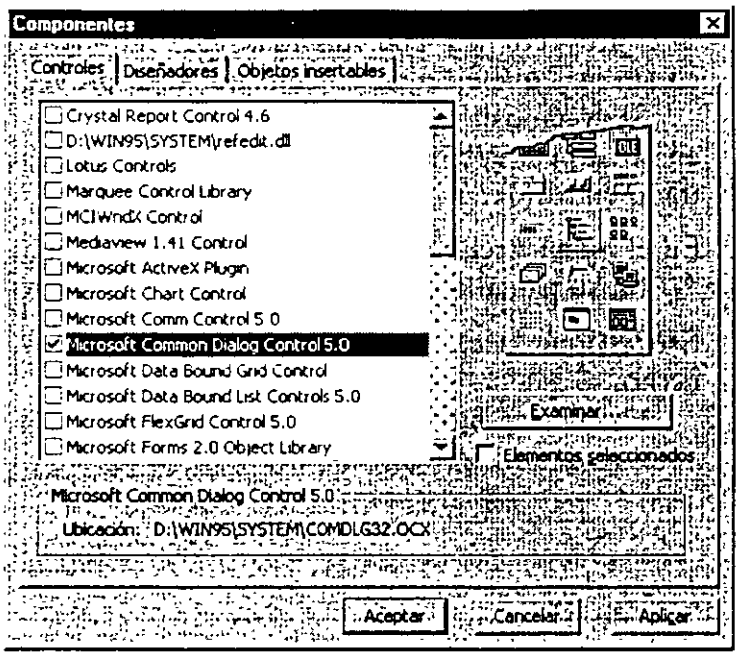
Los objetos de diálogo común no pueden ser vistos por el usuario en tiempo de ejecución

Ahora utilizaremos el control CommonDialog para añadir un objeto de diálogo común al formulario. El objeto de diálogo común aparece en un único tamaño y no es visible para el usuario en tiempo de ejecución (como el objeto no será visible, podrá introducirse en cualquier lugar del formulario). Al introducir el objeto en el formulario podrá utilizar cualquiera de los cinco cuadros de diálogo comunes en su programa.

Cómo añadir el control CommonDialog a su cuadro de herramientas

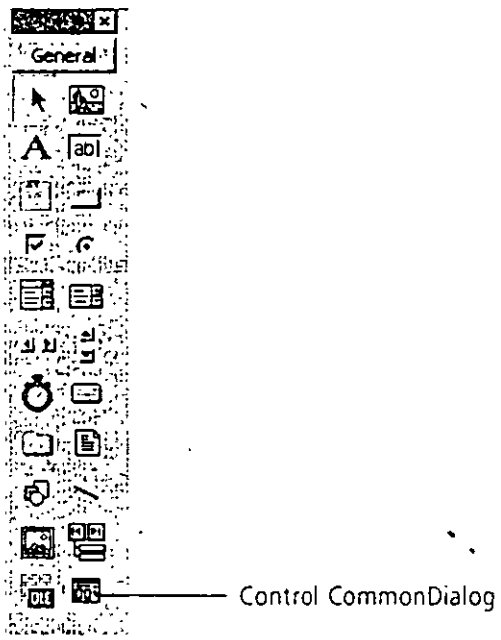
Si el control CommonDialog no se encuentra en su cuadro de herramientas podrá añadirlo sin más que ejecutar el mandato Componentes contenido en el menú Proyecto. Ejecute los siguientes pasos:

1. Abra el menú Proyecto y seleccione el mandato Componentes.
2. Pulse la etiqueta Controles y seleccione la opción Microsoft Common Dialog Control 5.0



3. Pulse Aceptar.

   El control CommonDialog aparece en su cuadro de herramientas tal y como se muestra en la siguiente figura:



Inserción de un objeto de diálogo común

Control CommonDialog

1. Pulse sobre el control CommonDialog contenido en el cuadro de herramientas.
2. Dibuje un objeto de diálogo común en la esquina inferior izquierda del formulario.

   Cuando termine de dibujar el objeto, Visual Basic modificará su tamaño. A partir de este momento podrá utilizar en sus programas el objeto común de diálogo.

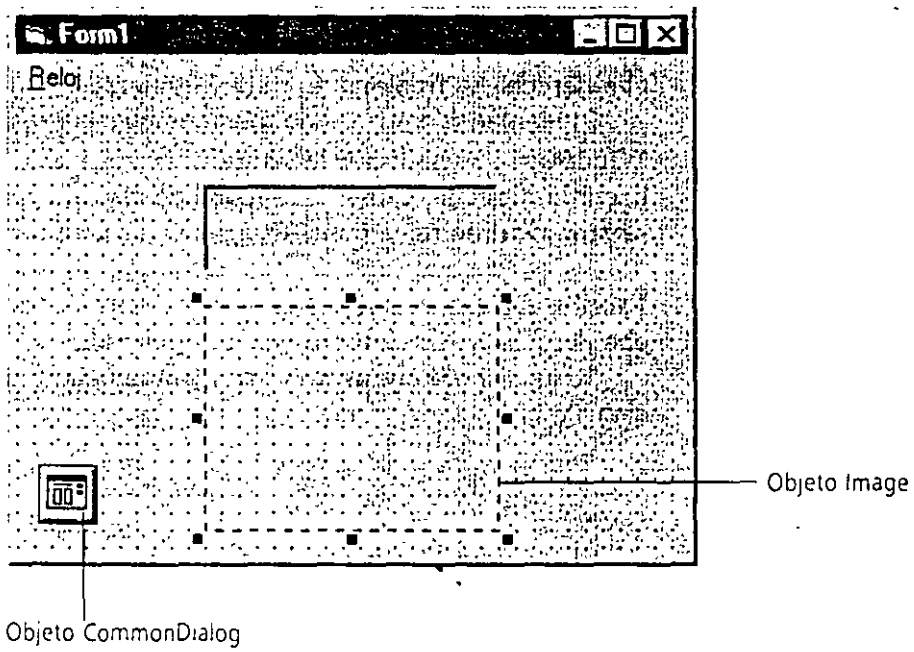
A continuación le mostraré cómo crear un objeto imagen utilizando el control Image. El objeto imagen será el encargado de mostrar los gráficos que el usuario seleccione en su programa utilizando el cuadro de diálogo común denominado Abrir.

Inserción de un objeto imagen

Control Image

1. Pulse el control Image contenido en el cuadro de herramientas.
2. Añada un objeto imagen en el formulario, por debajo de la etiqueta.
3. Utilice la ventana de Propiedades para asignar el valor True a la propiedad Stretch del objeto Imagen.

   Cuando finalice, su pantalla tendrá un aspecto similar al de la figura mostrada a continuación.



A continuación, utilizaremos el Editor de menús para añadir un menú Arch al programa MiDiálogo.

Inserción del menú Archivo

1. Pulse el formulario para seleccionarlo.

   Deberá seleccionar el formulario antes de poder añadir o modificar elementos de menú.

Botón Editor de menús

2. Pulse sobre el botón Editor de menús contenido en la barra de herramientas.

   En su pantalla aparece el Editor de menús. Dentro de este cuadro de diálogo aparece la estructura actual del menú definido en el programa MiDiálogo. A continuación, añadiremos un nuevo menú que contendrá los mandatos Abrir, Cerrar y Salir.

3. Pulse el botón Insertar cuatro veces.

   En la parte superior de la lista de elementos de menú aparecerán cuatro líneas vacías. Esta operación servirá para crear espacio vacío para los mandatos del menú Archivo que introduciremos a continuación.

4. Pulse el cuadro de texto Caption, escriba &Archivo, pulse TAB, escriba **mnuArchivo** y, finalmente, pulse el botón Siguiente.

   Se añadirá el menú Archivo al programa. La letra «F» se convertirá en la tecla de acceso.

5. Escriba **&Abrir...**, pulse TAB, escriba **mnuAbrirItem**, pulse el botón flecha derecha y, finalmente, pulse Siguiente.

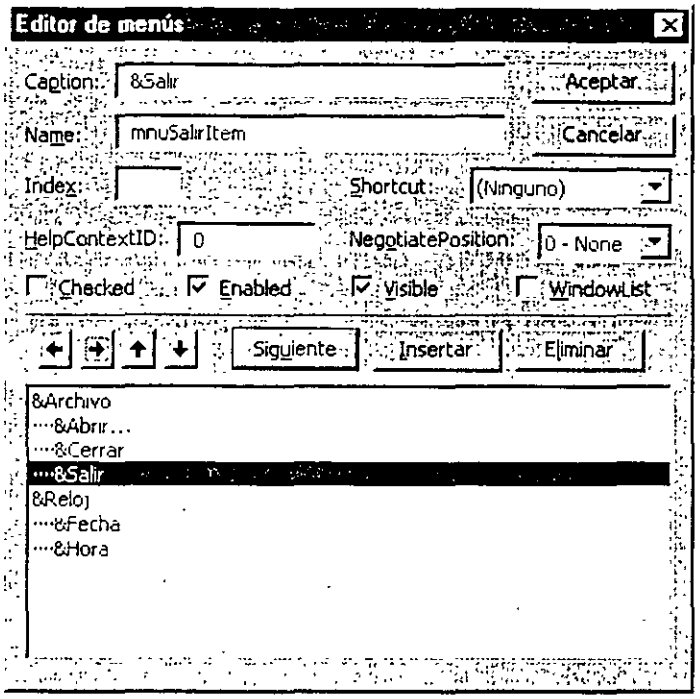
   El elemento Abrir (un mandato que nos permitirá abrir metaarchivos de Windows) se añadirá a la lista de menús y se desplazará cuatro espacios hacia la derecha. Como este mandato irá asociado a un cuadro de diálogo se han añadido unos puntos suspensivos a su nombre.

6. Escriba **&Cerrar**, pulse TAB, escriba **mnuCerrarItem**, pulse el botón flecha derecha y, finalmente, pulse Siguiente.

   Se añadirá el elemento Cerrar (un mandato que cerrará el archivo indicado) a la lista del menú.

7. Escriba **&Salir**, pulse TAB, escriba **mnuSalirItem** y pulse el botón Flecha Derecha.

   El elemento Salir (un mandato que cerrará la aplicación MiDiálogo) se añadirá a la lista del menú. Su pantalla tendrá un aspecto similar al siguiente:



Desactivación de un mandato de menú

En una aplicación típica de Windows no todos los mandatos se encuentran disponibles simultáneamente. Por ejemplo, en el típico menú Edición el mandato Pegar se encuentra disponible sólo si existe algún dato en el Portapapeles. Podrá desactivar un elemento de un menú deseleccionando, desde el Editor de menús, la casilla de verificación Enabled asociada con dicho mandato. Cuando un mandato se encuentra desactivado aparecerá en el menú que lo contiene en un color gris difuso.

En el siguiente ejercicio le mostraré cómo desactivar el mandato Cerrar (Cerrar es una opción que sólo podrá utilizar si un archivo ha sido previamente abierto por el programa). Más adelante, en este mismo capítulo, le mostraré cómo incluir una instrucción en el procedimiento de suceso asociado con el mandato Abrir para activar la opción Cerrar en el momento oportuno.

Cómo desactivar el mandato Cerrar

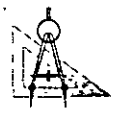
1. Pulse sobre el mandato Cerrar en la lista de menús.

   Las propiedades Caption y Name del mandato aparecen en el cuadro de diálogo.

2. Pulse sobre la casilla de verificación Enabled (activado) que se muestra en el Editor de menús para eliminar la marca de verificación.

   La marca de verificación desaparece de la casilla, por lo que la opción quedará desactivada.

A continuación le mostraré cómo añadir el mandato Color del Texto al me. Reloj para comprobar el funcionamiento del cuadro de diálogo común denominado Color. Este cuadro de diálogo devuelve un atributo de color al programa utilizando la propiedad CommonDialog1.Color. Utilizaremos esta propiedad para modificar el color del texto contenido en el cuadro de rótulo.



Adición del mandato Color del Texto en el menú Reloj

1. Pulse con el ratón sobre el elemento Fecha, situado al final de la lista de menús.
   Insertará el mandato Color del texto al final del menú Reloj.
2. Pulse sobre el botón Siguiente.
   En su pantalla aparece una nueva línea vacía al final de la lista de menús.
3. Escriba **&Color del texto...**, pulse TAB y escriba **mnuColortextoItem.**
   El mandato Color del Texto se añadirá al menú Reloj. El mandato contiene unos puntos suspensivos para indicar que se mostrará un cuadro de diálogo cuando el usuario lo seleccione. La tecla de acceso vuelve a ser la «C». En este caso, no hay ningún problema porque la otra opción con tecla de acceso «C» (mandato Cerrar) pertenece a otro menú. El sistema de tecla de acceso no funcionará correctamente si utiliza teclas duplicadas dentro del mismo nivel de un determinado menú, o se duplica a nivel de la bar de menús principal.
4. Pulse Aceptar para Cerrar el Editor de menús.

PROCEDIMIENTOS DE SUCESO QUE GESTIONAN LOS CUADROS DE DIÁLOGO COMUNES

Para mostrar un cuadro de diálogo común en un programa tendrá que llamar al objeto de diálogo común utilizando el método de objeto apropiado dentro de un procedimiento de suceso. Si fuera necesario, podrá definir una o más propiedades del cuadro de diálogo común antes de llamarlo empleando código de programa. Una vez que el usuario del programa realice sus selecciones en el cuadro de diálogo común, podrá procesar las opciones incluyendo código de programa en el procedimiento de suceso.

En el siguiente ejercicio deberá escribir el código del programa para el procedimiento de suceso denominado mnuAbrirItem_Click, es decir, la rutina que se ejecutará cuando el usuario seleccione el mandato Abrir. Tendrá que definir la propiedad Filter para el objeto CommonDialog1 para seleccionar el tipo de archivo que se mostrará en el cuadro de diálogo común Abrir (en este caso, especificaremos metaarchivos de Windows). A continuación, usaremos el método ShowOpe- para mostrar el cuadro de diálogo común Abrir. Una vez que el usuario haya sele

cionado un archivo y haya cerrado el cuadro de diálogo común, se mostrará el archivo en el objeto imagen asignando el nombre del archivo seleccionado por el usuario a la propiedad Picture del objeto Image1. Finalmente, se activará el mandato Cerrar para que el usuario pueda descargar el gráfico cuando quiera.

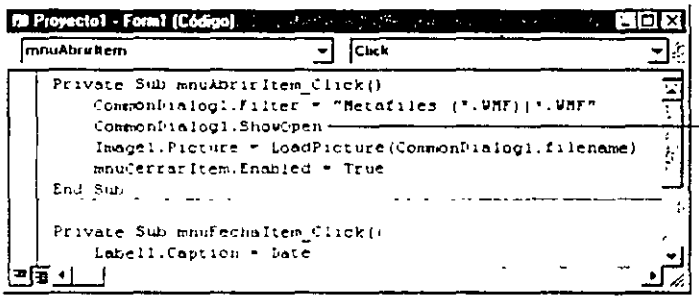
Edición del procedimiento de suceso para el mandato Abrir

Botón Ver Código

1. Pulse el botón Ver Código contenido en la ventana Proyecto.
2. Despliegue el cuadro de lista Objeto y pulse sobre el objeto mnuAbrirItem.
   En la ventana Código aparecerá al momento el procedimiento de suceso mnuAbrirItem_Click.
3. Escriba las siguientes instrucciones en el procedimiento de suceso, entre las sentencias Private Sub y End Sub.
   Desplace las instrucciones cuatro espacios hacia la derecha para diferenciar claramente estas sentencias del procedimiento de suceso al que pertenecen. Compruebe que introduce cada línea exactamente igual a como aparecen a continuación, y pulse la tecla FLECHA ABAJO después de finalizar cada una de ellas.

```
CommonDialog1.Filter = "Metafiles (*.WMF)|*.WMF"
CommonDialog1.ShowOpen
Image1.Picture = LoadPicture(CommonDialog1.FileName)
mnuCerrarItem.Enabled = True
```

Su pantalla tendrá un aspecto similar al siguiente:



La propiedad Filter define el tipo de archivo cuyo nombre se mostrara en el cuadro de diálogo Abrir

Las primeras tres líneas de este procedimiento de suceso hacen referencia a tres propiedades distintas del objeto CommonDialog1. La primera línea utiliza la propiedad Filter, que le permitirá definir una lista de archivos válidos (en este caso, la lista contiene únicamente un elemento: *.WMF). Este tema es importante para el cuadro de diálogo Abrir, ya que, tal y como aprendió en el Capítulo 2, los objetos imágenes pueden trabajar con seis tipos de archivos: mapas de bits (archivos .bmp), metaarchivos de Windows (archivos .wmf), iconos (archivos .ico), cursores (archivos .cur), formatos JPEG (archivos .jpg) y formatos GIF (archivos .gif)

(por ello, si intenta mostrar, por ejemplo, un archivo de tipo .txt en un objeto Imagen, se originará un error en tiempo de ejecución).

Si desea añadir elementos adicionales a la lista Filter podrá utilizar el carácter | para separar dos o más elementos. Por ejemplo:

```
CommonDialog1.Filter = "Bitmaps (*.BMP)|*.BMP|Metafiles
(*.WMF)|*.WMF"
```

La sentencia anterior hará que en el cuadro de objeto Abrir se muestren los nombres de archivos correspondientes a mapas de bits y a metaarchivos de Windows.

La segunda línea mostrará el cuadro de diálogo común Abrir dentro del programa que está desarrollando. Cada cuadro de diálogo común se muestra en pantalla utilizando un método de objeto distinto. El método que tendrá que utilizar para mostrar el cuadro de diálogo común Abrir es ShowOpen (consulte la tabla mostrada anteriormente en este capítulo para consultar los métodos que tendrá que utilizar para visualizar otros cuadros de diálogo comunes). Ésta es la sentencia más importante para el procedimiento de suceso. Como el nombre de la opción va a ser Abrir, el procedimiento necesita mostrar un cuadro de diálogo común del tipo Abrir (Open).

La tercera línea contenida en el procedimiento de suceso utiliza el nombre de archivo seleccionado por el usuario dentro del cuadro de diálogo. Cuando el usuario seleccione una unidad de disco, carpeta y nombre de archivo y pulse sobre botón Aceptar, el nombre de ruta completo del archivo seleccionado se comunicará al programa mediante la propiedad CommonDialog1.FileName. La función LoadPicture, una rutina que carga dibujos en formato artístico, es la encargada de copiar el metaarchivo de Windows especificado dentro del cuadro de imagen.

La última línea del procedimiento activa la opción Cerrar dentro del menú Archivo. Una vez abierto un archivo el usuario deberá poder disponer de la opción Cerrar para volver a cerrar dicho archivo.

A continuación, deberá introducir el código relacionado con el procedimiento de suceso denominado mnuColortextoItem_Click, es decir, la rutina que se ejecutará cuando el usuario seleccione el mandato Color del texto contenido en el menú Reloj.

Edición del procedimiento de suceso relacionado con el mandato Color del texto

1. Seleccione el objeto mnuColorTextoItem contenido en el cuadro de lista desplegable Objeto.

   En la ventana Código aparecerá el procedimiento de suceso relacionado con el mandato TextColor.

2. Escriba las siguientes instrucciones del programa (sangradas cuatro espacios a la derecha) en el procedimiento de suceso, entre las sentencias Private Sub y End Sub.

```
CommonDialog1.Flags = &H1&
CommonDialog1.ShowColor
Label1.ForeColor = CommonDialog1.Color
```

Control de las opciones de color utilizando banderas

La propiedad Bandera define el tipo del cuadro de diálogo Color que se mostrará en pantalla.

El procedimiento de suceso mnuColortextoItem_Click utiliza propiedades y métodos del objeto de diálogo común. La primera línea define una propiedad denominada Flags asignándola el valor &H1&, un valor hexadecimal que hace que el cuadro de diálogo común denominado Color presente una lista con las opciones estándar para el color, además de otros colores particularizados por el usuario y un color seleccionado por defecto. La siguiente tabla muestra los cuatro posibles valores para la propiedad Flag (bandera).

Flag	Significado
&H1&	Muestra un cuadro de diálogo Color estándar (con los colores personalizados como una opción) y especifica el color actual como el activo por defecto.
&H2&	Muestra un cuadro de diálogo Color personalizado y estándar.
&H4&	Muestra un cuadro de diálogo Color en el que se habrá desactivado el botón Definir colores personalizados.
&H8&	Muestra un botón de Ayuda dentro del cuadro de diálogo Color.

Podrá utilizar cualquier combinación de estos valores para preparar el cuadro de diálogo común Color antes de abrirlo. Si desea combinar dos o más valores podrá utilizar el operador Or. Por ejemplo,

```
CommonDialog1.Flags =&H1& Or &H8&
```

mostrará el mismo cuadro de diálogo Color que la sentencia anterior, pero en él se introducirá un botón de ayuda.

La segunda línea del procedimiento de suceso utiliza el método ShowColor para abrir el cuadro de diálogo común Color, mientras que la tercera línea asigna el color seleccionado a la propiedad ForeColor del objeto Label1. Anteriormente, en este mismo capítulo, ya hemos mencionado al objeto Label1 cuando desarrollamos el programa MiMenú. Se trata del cuadro de etiqueta que se utilizaba para mostrar en el formulario la hora y fecha actuales. Se utilizará el color devuelto por el cuadro de diálogo común Color para definir el color de fondo del texto contenido en la etiqueta.

Nota: Podrá utilizar el cuadro de diálogo Color para definir este atributo de cualquier elemento de la interfaz de usuario. Entre otras posibilidades podemos destacar el color de fondo, el color de las formas del formulario, y los colores de fondo y de primer plano de los objetos.

A continuación le mostraré el código que debe introducir en el procedimiento de suceso mnuCerrarItem_Click, la rutina que cierra el archivo mostrado en el objeto de imagen cuando el usuario seleccione el mandato Cerrar del menú Archivo.

Edición del procedimiento de suceso del mandato Cerrar

1. Despliegue el cuadro de lista objeto contenido en la ventana Código y seleccione el elemento mnuCerrarItem.

   El procedimiento de suceso asociado con el mandato Cerrar del menú Archivo aparece en la ventana Código.

2. Escriba las siguientes instrucciones del programa (sangradas cuatro espacios hacia la derecha) dentro del procedimiento de suceso, entre las instrucciones Private Sub y End Sub.

Utilice la función LoadPicture con comillas vacías para borrar una imagen o un cuadro de image.

```
Image1.Picture = LoadPicture("")
mnuCerrarItem.Enabled = False
```

La primera línea cierra el metaarchivo de Windows cargando una imagen vacía dentro del objeto Image1 (esta técnica se suele utilizar para borrar un cuadro de imagen o un objeto gráfico). La segunda línea desactiva el mandato Cerrar del menú Archivo, ya que no existe ningún archivo que se encuentre abierto.

A continuación, tendrá que introducir código para el procedimiento de suceso denominado mnuSalirItem_Click, la rutina que detiene la ejecución del programa cuando el usuario seleccione el mandato Salir contenido en el menú Archivo. Se trata del último procedimiento de suceso del programa.

Edición del procedimiento de suceso asociado con el mandato Salir

1. Seleccione el objeto mnuSalirItem contenido en el cuadro de lista desplegable Objeto.

   En la ventana Código aparecerá el procedimiento de suceso asociado con el mandato Salir del menú Archivo.

2. Escriba la siguiente instrucción (sangrada cuatro espacios hacia la derecha) dentro del procedimiento de suceso, entre las instrucciones Private Sub y End Sub.

```
End
```

Esta instrucción detiene el programa cuando el usuario así lo desee (a estas alturas, esta instrucción le debe resultar bastante familiar).

3. Cierre la ventana Código.

Botón Guardar Proyecto

4. Pulse el botón Guardar Proyecto contenido en la barra de herramientas para almacenar todo el proyecto en el disco.

   Visual Basic guardará los cambios realizados en los archivos de proyecto y formulario denominados MiDiálogo.

A partir de ahora estará preparado para ejecutar el programa MiDiálogo y experimentar con los menús y los cuadros de diálogo que haya creado.

Ejecución del programa MiDiálogo

Botón Iniciar

1. Pulse el botón Iniciar contenido en la barra de herramientas.

   El programa se ejecuta en el entorno de programación. Los menús Archivo y Reloj aparecen en la barra de menús.

El programa Diálogo.vbp completo se encuentra localizado en la carpeta \Vb5Sbs\Less03

2. Seleccione el mandato Abrir del menú Archivo.

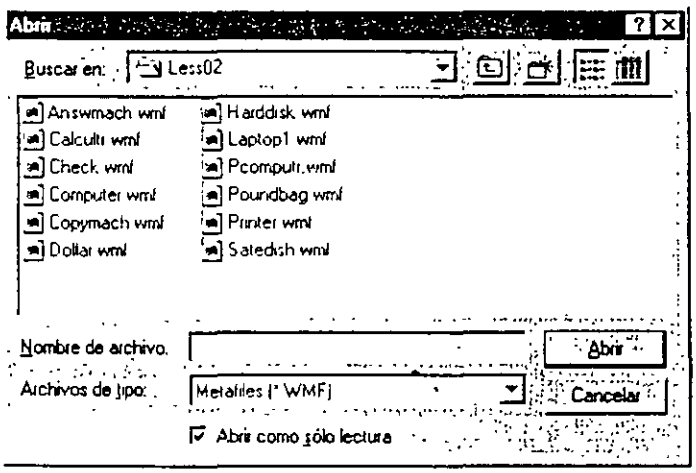
   El cuadro de diálogo común Abrir aparecerá en su pantalla. Observe que dentro del cuadro Archivos de tipo se muestran los metaarchivos (*.WMF). Esta situación se crea con la introducción de la sentencia

```
CommonDialog1.Filter = "Metafiles (*.WMF)|*.WMF"
```

   Dentro del procedimiento de suceso mnuAbrirItem_Click, la primera parte del texto encerrado entre comillas —Metafiles (*.WMF)— especifica qué elementos se van a listar dentro del cuadro Archivos de Tipo. La segunda parte —*.WMF— especifica la extensión que tendrán los nombres de los archivos que se listarán por defecto en el cuadro de diálogo.

3. Abra la carpeta \VB5Sbs\Less02 contenida en su disco fijo.

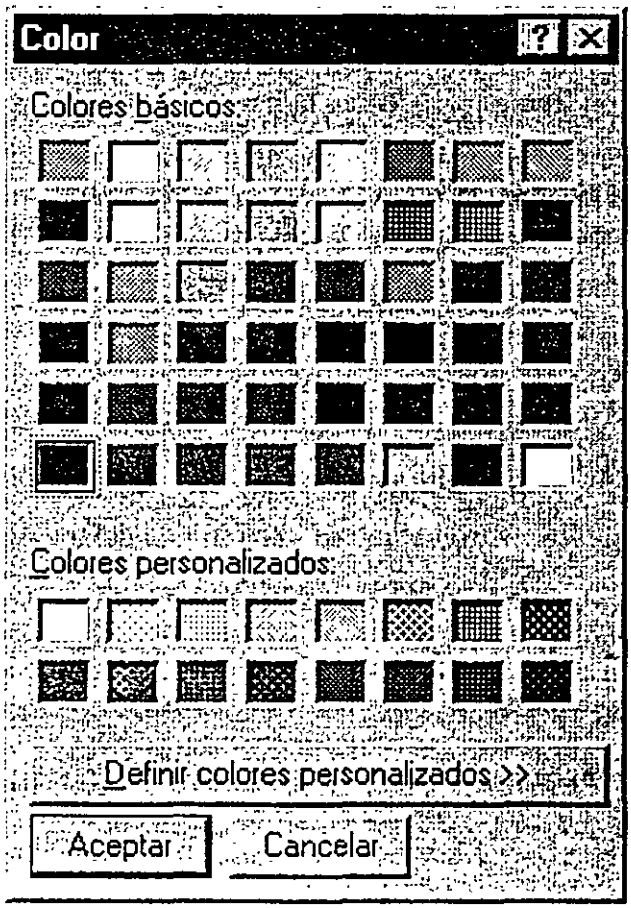
   Los metaarchivos de Windows contenidos en la carpeta Less02 aparecerán en el cuadro de lista de archivos, tal y como se muestra en la siguiente figura:



4. Realice una doble pulsación sobre el archivo pcomputr.wmf.
   En el cuadro de imagen aparecerá un dibujo de una computadora.
   Practique ahora utilizando el menú Reloj.

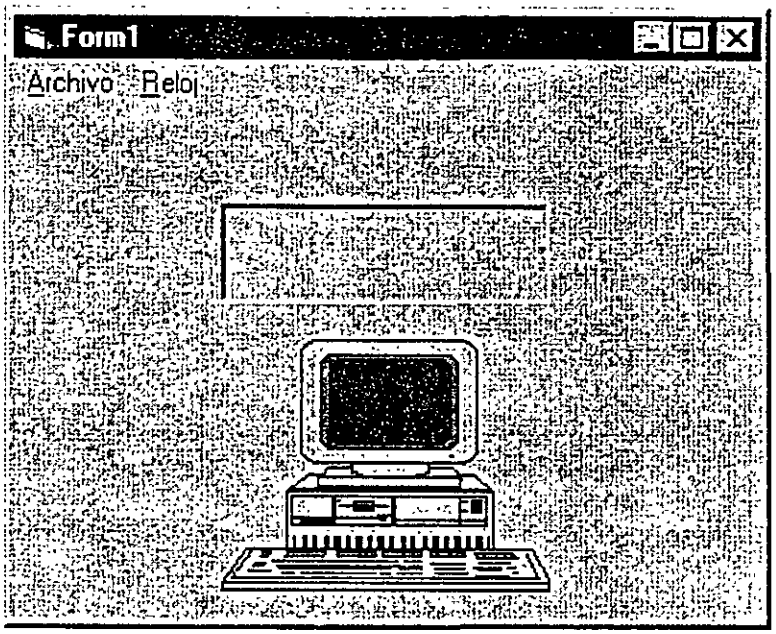
5. Seleccione el mandato Hora contenido en el menú Reloj.
   En el cuadro de etiqueta aparece la hora actual.

6. En el menú Reloj, seleccione el mandato Color del texto.
   En su pantalla aparecerá el cuadro de diálogo común Color, tal y como se muestra a continuación:



El cuadro de diálogo Color contiene elementos que le permitirán modificar el color utilizado para mostrar la hora en el programa. El color negro es el que se encuentra seleccionado.

7. Seleccione el color azul claro y, finalmente, pulse el botón Aceptar.
   Se cerrará el cuadro de diálogo común Color y el texto se mostrará ahora en azul claro.



8. En el menú Reloj seleccione el mandato Fecha.
   La fecha actual se mostrará en color azul claro. Una vez asignado un color para el texto, éste seguirá siendo el mismo hasta que el usuario vuelva a asignar otro color o hasta que termine la ejecución del programa.

9. Despliegue el menú Archivo.
   Observe que el mandato Cerrar se encuentra activado (lo activó en el procedimiento de suceso mnuAbrirItem_Click utilizando la instrucción mnuCloseItem.Enabled = True).

10. Pulse C para cerrar el metaarchivo de Windows.
    El archivo se cierra y la imagen desaparece de la pantalla.

11. Despliegue el menú Archivo.
    El mandato Cerrar se encuentra ahora desactivado, porque el cuadro de imagen no contiene ya ninguna imagen.

12. Ejecute el mandato Salir.
    El programa MiDiálogo se cierra. En su pantalla volverá a aparecer el entorno de programación de Visual Basic.

¡Eso es todo! En este apartado ha conocido el empleo de varios mandatos importantes y de las técnicas necesarias para crear menús y para introducir cuadros de diálogo en sus programas. Una vez que aprenda más acerca del código de programación, será capaz de desarrollar programas realmente complejos y vistosos con el mínimo esfuerzo.

Cómo añadir cuadros de diálogo no estándar en sus programas

¿Qué ocurre si necesita añadir un cuadro de diálogo en un programa que no sea uno de los cinco cuadros de diálogo comunes existentes? No hay ningún problema, aunque tendrá que llevar a cabo un poco más de trabajo de diseño. Como aprenderá en los próximos capítulos, un programa de Visual Basic puede utilizar más de un formulario para recibir y mostrar información. Para crear cuadros de diálogo no estándar tendrá que añadir nuevos formularios en su programa, añadir objetos de entrada y de salida, y procesar las operaciones que el usuario realice sobre los cuadros de diálogo añadiendo código de programa (todas estas técnicas serán comentadas en el Capítulo 7). En el siguiente capítulo le mostraré cómo utilizar dos cuadros de diálogo que han sido específicamente diseñados para recibir entradas textuales (InputBox) y para mostrar una salida textual (MsgBox). Estos cuadros de diálogo le ayudarán a cubrir el espacio vacío existente entre los cuadros de diálogo comunes y aquellos que tendrá que crear por su propia cuenta.

UN PASO MÁS ALLÁ: ASIGNACIÓN DE TECLAS DE ACCESO RÁPIDO A LOS MENÚS

El Editor de menús también le permite asignar teclas de acceso rápido a sus menús. Las *teclas atajo* o de acceso rápido son combinaciones de teclas que el usuario puede utilizar para activar un mandato sin tener que utilizar la barra de menús. Por ejemplo, en el típico menú de Edición de las aplicaciones para Windows (tal como Visual Basic) podrá copiar el texto seleccionado al portapapeles sin más que pulsar simultáneamente las teclas CTRL+C. A continuación le mostraré cómo asignar teclas atajo a las opciones contenidas en el menú Reloj del programa MiMenú.

Asignación de teclas de acceso directo al menú Reloj

Botón Editor de menús

1. Pulse el botón Editor de menús contenido en la barra de herramientas.
   En su pantalla aparecerá el Editor de menús.

2. Seleccione la opción Hora en el cuadro de lista de menús.
   En el cuadro de diálogo aparecerán los valores correspondientes a los atributos Caption y Name de la opción Hora. Podrá asociar a esta opción una combinación de teclas sin más que seleccionar la combinación de teclas deseada dentro del cuadro de lista desplegable denominado Shortcut (acceso directo). A continuación, asignaremos la combinación CTRL+H como atajo para este mandato.

No puede asignar una tecla atajo a un menú principal.

3. Despliegue el cuadro de lista denominado Shortcut y avance el contenido de la lista hasta encontrar la combinación de teclas CTRL+H.

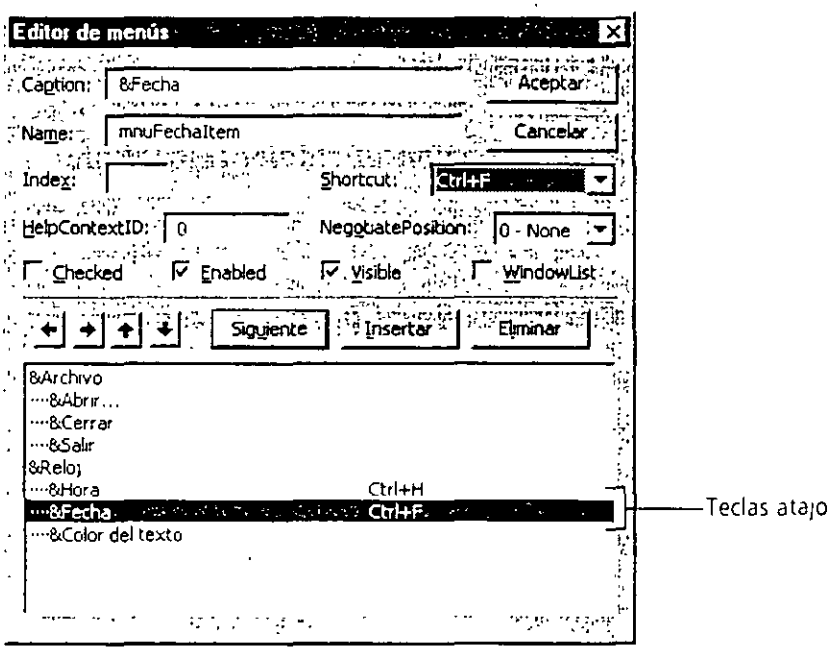
   Se asignará CTRL+H como combinación de acceso directo al mandato Hora. Esta combinación de teclas aparecerá en el cuadro de lista de menús.

4. Pulse el botón Siguiente.

   En el cuadro de diálogo aparecerán los atributos Caption y Name de la opción Fecha. A continuación le mostraré cómo asignar la combinación CTRL+F como tecla atajo para esta opción de menú.

5. Despliegue el cuadro de lista Shortcut y seleccione la combinación CTRL+F.

   Su pantalla deberá tener un aspecto similar al siguiente:



6. Pulse el botón Aceptar para cerrar el Editor de menús.

   Ahora ejecutaremos el programa y probaremos las teclas atajo.

7. Pulse el botón Iniciar contenido en la barra de herramientas.

Botón Iniciar

8. Pulse la combinación CTRL+H para ejecutar la opción Hora.

   La hora actual aparecerá en el programa.

9. Pulse la combinación CTRL+F para ejecutar la opción Fecha.

   En el programa aparecerá la fecha actual.

10. Despliegue el menú Reloj.

    Las teclas de acceso directo se muestran a la derecha de las opciones Hora y Fecha. Visual Basic añade estas combinaciones de teclas cuando se definen las teclas atajo utilizando el Editor de menús.

11. En el menú Archivo ejecute la opción Salir.

El programa se detendrá y volverá a aparecer el entorno de programación.

Botón Guardar proyecto

12. Pulse el botón Guardar proyecto contenido en la barra de herramientas para almacenar en el disco las teclas atajo que acaba de definir.

Si desea revisar sus conocimientos de programación

Consulte el ejercicio Revisión y Práctica 1 contenido en el archivo Rev1.doc que se encuentra en la carpeta \Vb5Sbs\Review de su disco fijo.

Este documento contiene ejercicios paso a paso que podrá utilizar para poner a prueba sus conocimientos de programación y para revisar el contenido de los capítulos 1 a 3. La aplicación que va a desarrollar le permitirá ver el contenido de archivos de mapas de bits.

Si desea continuar en el siguiente capítulo

➤ Mantenga en marcha Visual Basic, y pase al Capítulo 4.

Si desea salir de Visual Basic por ahora

➤ En el menú Archivo seleccione Salir.
Si en su pantalla se muestra un cuadro de diálogo Guardar, pulse Sí.

RESUMEN DEL CAPÍTULO

Para	Haga esto	Botón
Crear un elemento de menú	Pulse el botón Editor de menús y, a continuación, defina los atributos caption, nombre y posición del elemento del menú.	
Añadir una tecla de acceso a un elemento del menú	Ponga en marcha el Editor de menús, seleccione la opción de menú que desee y active el cuadro de texto denominado Caption. Escriba un carácter ampersand (&) delante de la letra que desee definir como tecla de acceso.	
Asignar una tecla atajo a una opción de menú	Ponga en marcha el Editor de menús y seleccione la opción de menú deseada. Especifique la tecla atajo que desea asociar dentro del cuadro de lista desplegable denominado Shortcut.	

Para	Haga esto	Botón
Modificar el orden en que se muestran las opciones de un menú	Ponga en marcha el Editor de menús. Seleccione la opción de menú que desee mover, y pulse el botón flecha arriba o el botón flecha abajo para desplazar dicha opción.	
Emplear un cuadro de diálogo estándar en sus programas	Añada un objeto de diálogo común a su formulario y utilice uno de los cinco métodos de diálogo comunes en el código del programa para poder mostrar el cuadro de diálogo.	
Desactivar un menú	Con el Editor de menús, elimine la marca de verificación del cuadro de verificación Enabled asociado con la opción de menú elegida.	
Activar una opción de menú utilizando código de programa	Utilice la sentencia `mnuCerrarItem.Enabled = True` pero sustituya *mnuCerrarItem* por el nombre de su opción de menú.	
Vaciar un cuadro de imagen	Utilice la sentencia de programa `Image1.Picture = LoadPicture("")`	

Sobre la ayuda en línea	En el menú Ayuda de Visual Basic, seleccione el mandato Temas de ayuda de Microsoft Visual Basic, seleccione la etiqueta Índice y a continuación
El Editor de menús	Escriba **Editor de menús**
El control CommonDialog	Escriba **CommonDialog**
Tipos y métodos de diálogo comunes	Escriba **ShowOpen**, **ShowSave**, **ShowPrinter**, **ShowFont** o **ShowColor**

AVANCE DEL SIGUIENTE CAPÍTULO

La Parte 2 (Capítulos 4 a 6) cubre las bases de la programación que podrá utilizar para obtener la máxima potencia de sus programas. En el Capítulo 4, «Variables y operadores de Visual Basic», aprenderá a crear variables en sus programas. También conocerá la forma en que Visual Basic maneja los operadores y las funciones matemáticas.

SEGUNDA PARTE

Fundamentos de programación

Capítulo

4

Variables y operadores de Visual Basic

En la primera parte del libro hemos aprendido a crear interfaces de usuario con Microsoft Visual Basic y a desarrollar y ejecutar programas en el entorno de programación de Visual Basic. En los tres capítulos siguientes aprenderemos más acerca de la programación en Visual Basic; en particular, profundizaremos en el conocimiento de las sentencias y palabras clave que forman el núcleo de un programa de Visual Basic. Una vez completada la segunda parte, estará preparado para analizar temas más avanzados.

En este capítulo aprenderemos a utilizar variables para almacenar datos y a emplear operadores matemáticos para realizar tareas como la suma y la multiplicación. También comenzaremos a utilizar funciones matemáticas con las que podremos realizar cálculos con números y utilizaremos las funciones InputBox y MsgBox para gestionar y presentar información desde cuadros de diálogo.

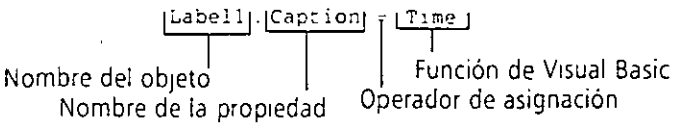
ANATOMÍA DE UNA SENTENCIA EN VISUAL BASIC

Una sentencia de programa es una instrucción válida para el compilador de Visual Basic.

Tal como comentamos en el Capítulo 1, una línea de código en un programa de Visual Basic recibe el nombre de sentencia o declaración de programa. Una *sentencia de programa* es cualquier combinación de palabras clave, propiedades, funciones, operadores y símbolos de Visual Basic que, en conjunto, constituyen una instrucción válida comprensible por el compilador de Visual Basic. Una sentencia de programa completa puede ser simplemente una palabra clave como:

```
Beep
```

que emitirá una nota a través del altavoz de la computadora, o bien puede ser una combinación de elementos como la siguiente sentencia, que asigna la hora actual del sistema a la propiedad Caption de una etiqueta:



Las reglas que se deberán seguir para la construcción de una sentencia de programa es lo que se denomina su sintaxis. Visual Basic comparte muchas reglas de sintaxis con las versiones anteriores del lenguaje Basic y con compiladores de otros lenguajes. El truco para escribir buenas sentencias de programa consiste en aprender la sintaxis de los elementos más útiles del lenguaje y después utilizar correctamente dichos elementos para procesar los datos en el programa. Afortunadamente, Visual Basic facilita mucho esta tarea, de forma que el tiempo empleado en escribir código de programa es relativamente corto y los resultados pueden volver a utilizarse en programas futuros.

En los capítulos siguientes les mostraré las palabras clave y las sentencias de programa más importantes de Visual Basic. Como podrá observar, se complementan perfectamente con las técnicas que ya hemos aprendido y le ayudarán a escribir en el futuro potentes programas. Las variables y los tipos de datos, los primeros temas, son elementos críticos de cualquier programa.

EMPLEO DE VARIABLES PARA ALMACENAR INFORMACIÓN

Una *variable* es una ubicación temporal de almacenamiento de datos dentro de un programa. En nuestro código podremos utilizar una o más variables y éstas podrán contener palabras, números, fechas o propiedades. La utilidad de las variables radica en que permiten asignar un nombre corto y fácil de recordar a los datos con los que pensamos trabajar. Las variables pueden almacenar información introducida por el usuario en tiempo de ejecución, o bien el resultado de un cálculo específico o una porción de datos que queremos mostrar en el formulario. En resumen, las variables son herramientas que podemos utilizar para manejar cualquier tipo de información.

El manejo de variables en un programa de Visual Basic es similar al uso de una mesa en un restaurante. Podemos empezar utilizando una en cualquier parte del código de programa, pero la gestión será mucho mejor si la reservamos de antemano. En las dos secciones siguientes estudiaremos el proceso de creación de reservas, o *declaración*, de una variable.

Cómo reservar espacio para las variables: La instrucción Dim

Dim reserva espacio para una variable.

Para declarar explícitamente una variable antes de utilizarla (normalmente al principio de un procedimiento de suceso) deberá escribir el nombre de la variable

trás de la sentencia Dim (Dim es la abreviatura de *Dimensión*). Esta acción reservará espacio en memoria para la variable cuando se ejecute el programa y permitirá a Visual Basic saber qué tipos de datos deberá guardar en dicha variable. Por ejemplo, la siguiente instrucción crea espacio para una variable llamada Apellido en un programa.

```
Dim Apellido
```

Después del nombre de la variable podemos especificar, de forma opcional, el tipo de la misma. (Más adelante, en este capítulo, le mostraré cuáles son los tipos básicos de variables). Visual Basic permite identificar de antemano el tipo de la variable, de forma que se puede controlar cuánta memoria dedicará el procesador a almacenarla. Por ejemplo, si la variable va a albergar un pequeño número sin cifras decimales (un entero), podrá declarar la variable como entero y ahorrar espacio en memoria. No obstante, si el programador no indica lo contrario, Visual Basic reserva por omisión espacio para un tipo de variable denominado *Variant*, que es una variable que puede contener cualquier tamaño o formato. El tipo Variant es extremadamente flexible y puede ser, perfectamente, el único tipo de variable que utilice en sus programas.

Para asignar datos en una variable deberá utilizar el operador de asignación (=).

Después de declarar una variable podrá asignar información a la misma utilizando código. Por ejemplo, la siguiente sentencia de programa asigna el nombre «Jeremías» a la variable Apellido.

```
Apellido = "Jeremías"
```

Después de esta asignación, la variable Apellido podrá utilizarse en lugar del apellido «Jeremías». Por ejemplo, la sentencia de asignación:

```
Label1.Caption = Apellido
```

mostrará la palabra *Jeremías* en la primera etiqueta (Label1) de su formulario.

Declaración de variables sin utilizar Dim

También podrá declarar variables sin utilizar la sentencia Dim; este proceso se denomina *declaración implícita*. Para declarar una variable de esta forma bastará con utilizar la variable por sí sola ignorando la sentencia Dim:

```
Nombre = "Carlos V"
```

La declaración implícita tiene la ventaja de ser más rápida, ya que no perderá tiempo en escribir la sentencia Dim, pero la gestión es a menudo peor, ya que la declaración implícita no obliga a organizar y listar las variables de antemano e impedirá que Visual Basic muestre un mensaje de error cuando posteriormente, en el código

del programa, utilice la variable declarada anteriormente pero introduciéndola c algún error ortográfico (consulte la nota siguiente). En este libro se utilizarán ambas técnicas para declarar las variables.

Nota: Si decide declarar siempre sus variables utilizando la sentencia Dim, puede interesarle introducir la sentencia OptionExplicit en la sección de declaraciones de su formulario inicial. Podrá llevar a cabo esta tarea de forma automática para cada nuevo proyecto sin más que ejecutar el mandato Opciones contenido en el menú Herramientas, pulsando el tabulador Editor y activando la opción denominada Requerir declaración de variables.

Al utilizar OptionExplicit, Visual Basic genera un mensaje de error siempre que encuentre una variable no declarada explícitamente en el código (una razón probable para que se diera esta situación podría ser un error mecanográfico en el nombre de la variable). Si le preocupan los errores mecanográficos, esta instrucción puede ser de su interés.

Empleo de variables en un programa

Las variables pueden mantener el mismo valor a lo largo de todo un programa o pueden cambiar su valor varias veces, dependiendo de sus necesidades. En el siguiente ejercicio se muestra cómo una variable llamada Apellido puede contener tanto texto como números y cómo puede asignarse su valor a las propiedades un objeto.

Modificación del valor de una variable

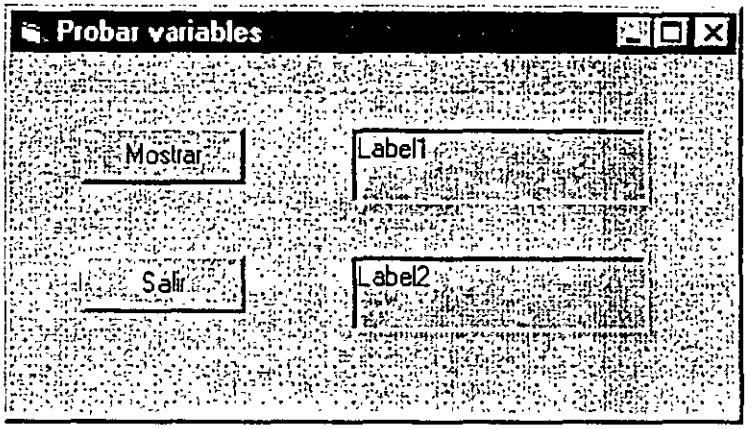
Botón Ver Objeto

1. Ponga en marcha Visual Basic

2. En el menú Archivo, seleccione la opción Abrir proyecto.
   Aparecerá el cuadro de diálogo Abrir proyecto.

3. Abra el proyecto denominado PruebaVar contenido en la carpeta \Vb5Sbs\Less04.
   El proyecto PruebaVar (Prueba de variables) se abrirá en el entorno de programación. PruebaVar es un programa esqueleto que contiene un formulario con etiquetas y botones para mostrar salidas, pero sólo cuenta con unas breves líneas de código de programa. En este ejercicio le pediré que añada dicho código.

4. Resalte el nombre del formulario en la ventana de Proyecto y pulse sobre el botón Ver Objeto contenido en dicha ventana.
   En la pantalla aparecerá el formulario Probar variables tal como se muestra a continuación:



El formulario contiene dos etiquetas y dos botones de orden. Utilizaremos variables para mostrar información en cada una de las etiquetas.

5. Pulse dos veces el botón de orden Mostrar.

   El procedimiento de suceso Command1_Click aparecerá en la ventana Código.

6. Escriba las siguientes sentencias de programa para declarar y utilizar la variable Apellido:

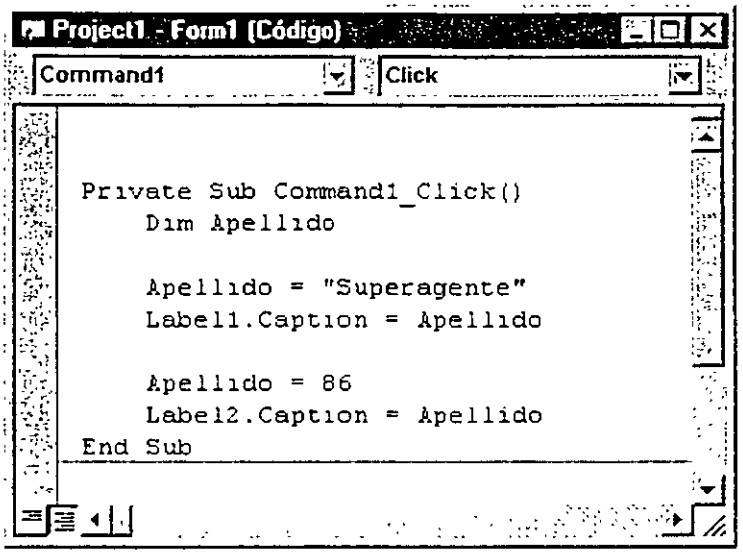
```
Dim Apellido

Apellido = "Superagente"
Label1.Caption = Apellido

Apellido = 86
Label2.Caption = Apellido
```

Las variables pueden transferir información a una propiedad

Las sentencias de programa están organizadas en tres grupos. La primera sentencia declara la variable Apellido utilizando la sentencia Dim. Como no se ha especificado ningún tipo, la variable se declara de tipo variante, es decir, un tipo genérico de variable que puede contener texto o números. Las líneas segunda y tercera asignan el nombre «Superagente» a la variable Apellido y muestran después dicho nombre en la primera etiqueta del formulario. De esta forma se muestra uno de los usos más comunes de las variables en un programa: la transferencia de información a una propiedad.

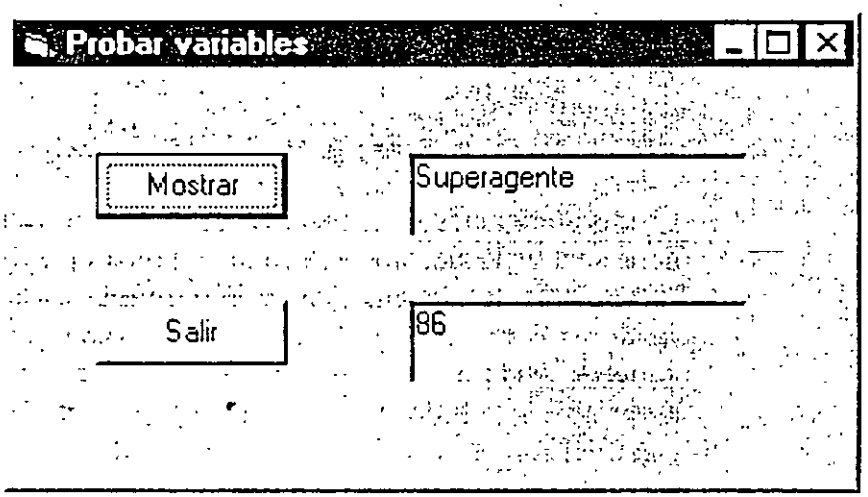
La cuarta línea asigna el número 86 a la variable Apellido. Esta operación borra de la variable el rótulo y lo reemplaza por un número. El número no se ha colocado entre comillas. Los rótulos necesitan utilizar comillas, pero los número no. (Si encierra al número entre comillas, dicho número será tratado como cadena de texto y no podrá utilizarse en fórmulas matemáticas).

Su pantalla tendrá un aspecto similar al siguiente:



Botón Iniciar

7. Pulse el botón Iniciar de la barra de herramientas para ejecutar el programa.
   El programa se ejecutará en el entorno de programación.
8. Pulse el botón de orden Mostrar.
   El programa declara la variable, le asigna datos dos veces y la copia a las dos etiquetas del formulario. El programa generará la siguiente salida:



9. Pulse el botón Salir para detener el programa.
   El programa se detendrá y volverá a aparecer el entorno de programación.
10. Guarde los cambios realizados en su formulario bajo el nombre **MiPruebaVar.frm** utilizando la opción Guardar PruebaVar.frm como.
    Guarde las modificaciones realizadas en el proyecto con el nombre **MiPruebaVar.vbp** utilizando la opción Guardar proyecto como.

Convenios de nomenclatura para variables

La nomenclatura de las variables puede ser algo complicada, ya que necesitará utilizar nombres cortos pero a su vez intuitivos y fáciles de recordar. Para evitar confusiones, utilice los siguientes convenios cuando asigne nombres a la variables.

- El nombre de las variables debe comenzar por una letra. Esto es un requisito en Visual Basic. Los nombres de variables deberán tener menos de 256 caracteres de longitud y no pueden contener puntos.
- Elija nombres descriptivos. Si es necesario puede combinar una o más palabras en el nombre de una variable. Por ejemplo, el nombre de la variable ImpuestoSobreVentas es más claro que Impuesto o Venta.
- Utilice combinaciones de mayúsculas, minúsculas y números, si así lo desea. Un convenio aceptado consiste en poner en mayúsculas la primera letra de cada palabra de una variable; por ejemplo, FechaDeNacimiento.
- No utilice palabras clave, objetos o propiedades de Visual Basic como nombre de una variable.

EMPLEO DE UNA VARIABLE PARA ALMACENAR ENTRADAS

Podrá obtener datos del usuario utilizando la función InputBox y una variable.

Un excelente uso para una variable es el de guardar la información introducida por el usuario. A menudo podremos utilizar un objeto (como un cuadro de lista de archivo o un cuadro de texto) para obtener dicha información, pero en ocasiones nos interesará manejar directamente la información introducida por el usuario y guardar la entrada en una variable, en lugar de hacerlo en una propiedad. La forma de llevar esto a cabo es utilizar la función InputBox, que mostrará en pantalla un cuadro de diálogo, y finalmente almacenar la información introducida por el usuario en una variable. En el siguiente ejemplo utilizaremos esta técnica.

Obtención de datos de entrada utilizando InputBox

1. En el menú Archivo seleccione la opción Abrir proyecto.
   Aparecerá el cuadro de diálogo Abrir proyecto.
2. Abra el proyecto CuadroEntrada contenido en la carpeta \Vb5Sbs\Less04.
   El proyecto CuadroEntrada se abrirá en el entorno de programación. Cuadro de entrada es un programa esqueleto cuyo formulario contiene dos

botones de orden y una etiqueta para mostrar la salida, pero también .u-
ye algo de código de programa.

3. Resalte el formulario en la ventana Proyecto y pulse sobre el botón Ver Objeto contenido en dicha ventana.

   El formulario Prueba de un cuadro de entrada aparecerá en la pantalla. El formulario contiene una etiqueta y dos botones de órdenes. Utilizaremos la función InputBox para obtener una entrada de información del usuario y, finalmente, mostraremos dicha entrada en la etiqueta del formulario.

4. Pulse dos veces el botón de orden Introducir.

   El procedimiento de suceso Command1_Click aparecerá en la ventana Código.

5. Escriba las siguientes sentencias de programa para declarar dos variables e invocar a la función InputBox:

```
Dim Mensaje, Nombre
Mensaje = "Por favor, escriba su nombre."

Nombre = InputBox$(Mensaje)
Label1.Caption =Nombre
```

   En esta ocasión, habrá declarado dos variables mediante el uso de una única sentencia Dim: Mensaje y Nombre. La segunda línea del procedimiento de suceso asigna un grupo de caracteres o una *cadena de texto* a la variable Mensaje.

   Este mensaje se utilizará como argumento de texto para la función InputBox. (Un *argumento* es un valor o expresión pasado a un subprocedimiento o a una función.) La siguiente línea llama a la función InputBox y asigna el resultado de la llamada (la cadena de texto introducida por el usuario) a la variable Nombre. InputBox es una función especial de Visual Basic que muestra en pantalla un cuadro de diálogo y pide al usuario que introduzca un valor. Además de mostrar en pantalla un rótulo, la función InputBox permite otros argumentos que podrá utilizar ocasionalmente. Si desea obtener más detalles relacionados con este tema consulte la ayuda interactiva de Visual Basic.

   Después de que InputBox haya devuelto una cadena de texto al programa, la cuarta sentencia del procedimiento asigna el nombre del usuario a la propiedad Caption del objeto Label1, que es la encargada de mostrarlo en el formulario.

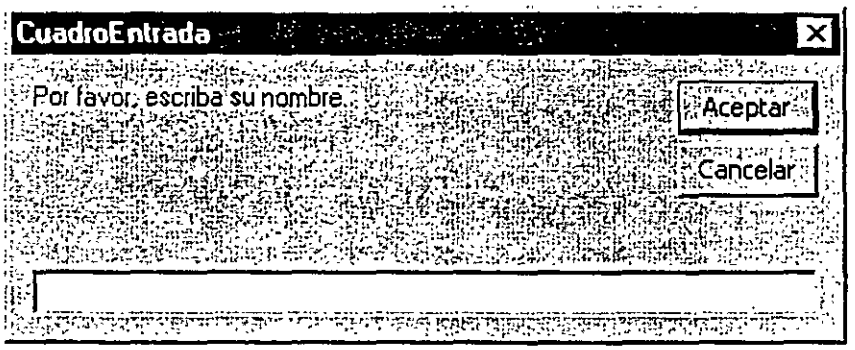
Botón Iniciar

6. Pulse el botón Iniciar de la barra de herramientas para ejecutar el programa.

   El programa se ejecutará en el entorno de programación.

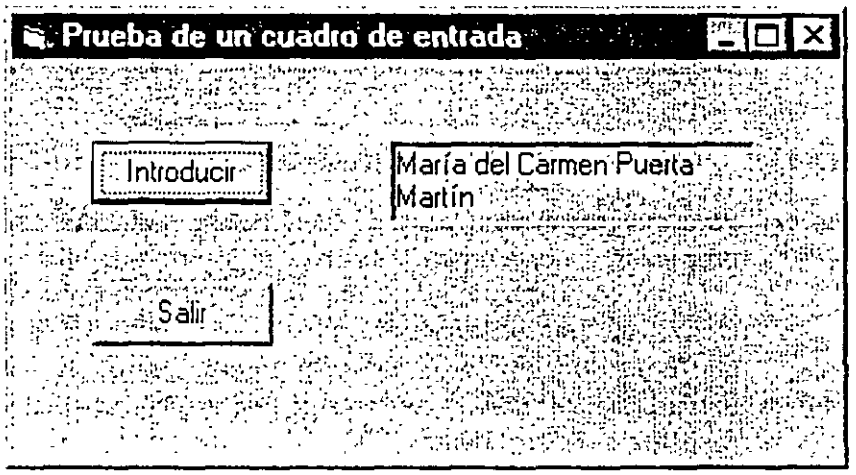
7. Pulse el botón Introducir.

   Visual Basic ejecutará el procedimiento de suceso Command1_C[illegible] y el cuadro de diálogo denominado Cuadroentrada aparecerá en panta



8. Escriba su nombre completo y pulse Aceptar.

   La función InputBox devolverá al programa su nombre y lo colocará en la variable Nombre. El programa utilizará después dicha variable para mostrar su nombre en el formulario:



Podrá utilizar la función InputBox siempre que desee solicitar información al usuario. Podrá utilizarla en combinación con otros controles de entrada para regular el flujo de datos de entrada y salida de un programa. En el siguiente ejercicio le mostraré cómo utilizar una función similar para mostrar texto en un cuadro de diálogo.

9. Pulse el botón Salir para detener el programa.

   El programa se detendrá y volverá a aparecer el entorno de programación.

10. Guarde los cambios en su formulario y en su proyecto con el nombre **MiCuadroEntrada**.

¿Qué es una función?

InputBox es una palabra clave especial de Visual Basic conocida como función. Una *función* es una sentencia que realiza una determinada tarea (por ejemplo, la petición de información al usuario o el cálculo de una ecuación) y después devuelve un resultado al programa. El valor devuelto por una función puede asignarse a una variable tal como hemos hecho en el programa MiCuadroEntrada o bien puede asignarse a una propiedad o a otra sentencia o función. Las funciones de Visual Basic utilizan, a menudo, uno o más argumentos para definir su actividad. Por ejemplo, la función InputBox que acaba de utilizar usaba la variable Mensaje para mostrar al usuario ciertas instrucciones dentro del cuadro de diálogo. Cuando una función utilice dos o más argumentos, dichos argumentos se separarán por comas y el grupo completo de argumentos se encierra entre paréntesis. La siguiente sentencia muestra una llamada a una función que utiliza dos argumentos:

```
Nombre = InputBox$(Mensaje, Título)
```

Nombre de la variable
Operador de asignación
Nombre de la función
Argumentos

EMPLEO DE UNA VARIABLE PARA SALIDA DE INFORMACIÓN

La función MsgBox utiliza rótulos para mostrar la salida en un cuadro de diálogo Con ella podrá utilizar una serie de argumentos opcionales.

Para mostrar el contenido de una variable podemos asignar dicha variable a una propiedad (por ejemplo, la propiedad Caption de un objeto de etiqueta) o pasar la variable como argumento a una función de cuadro de diálogo. Una de estas funciones, de gran utilidad para mostrar una salida, es la función MsgBox. Al igual que InputBox, utiliza uno o más argumentos como entrada y el resultado de la llamada a la función puede asignarse a una variable. La sintaxis de la función MsgBox es la siguiente:

```
BotónPulsado = MsgBox(Mensaje, NúmeroDeBotones, Título)
```

donde *Mensaje* es el texto que aparecerá en la pantalla. *NúmeroDeBotones* es un número de estilo de botón (entre 1 y 5) y *Título* es el texto que se mostrará en la barra de título del cuadro de mensajes. La variable BotónPulsado recibe el resultado devuelto por la función, que indica qué botón se ha pulsado en el cuadro de diálogo.

Si únicamente desea mostrar un mensaje en MsgBox, tanto el operador de asignación (=) como la variable BotónPulsado y el argumento *NúmeroDeBotones* serán opcionales. En el siguiente ejercicio no los utilizaremos; si desea obtener más información sobre los mismos (incluyendo los distintos tipos de botones que se pueden utilizar en MsgBox) busque *MsgBox* en la ayuda interactiva de Visual Basic.

A continuación le mostraré cómo añadir una función MsgBox al programa MiCuadroEntrada para visualizar el nombre que el usuario introduce en el cuadro de diálogo InputBox.

Cómo mostrar un mensaje utilizando MsgBox

1. Pulse dos veces el botón Introducir contenido en el formulario MiCuadroEntrada.

   El procedimiento de suceso Command1_Click aparecerá en la ventana Código (Deberá contener el código introducido en el último ejercicio).

2. Utilice el ratón para seleccionar la siguiente sentencia del procedimiento de suceso (la última línea).

```
Label1.Caption = Nombre
```

   Ésta es la sentencia que muestra en la etiqueta el contenido de la variable Nombre.

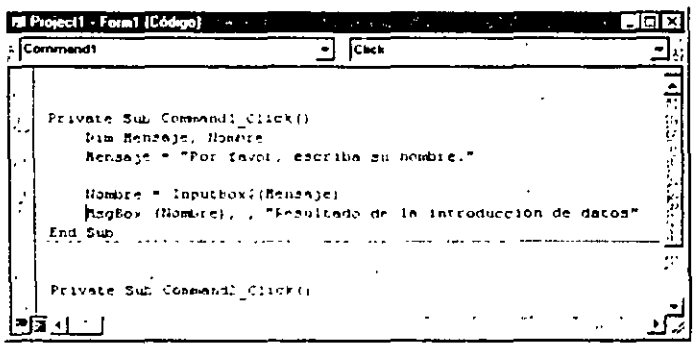
3. Pulse SUPR para borrar la línea.

   La instrucción desaparecerá de la ventana Código.

4. Escriba la siguiente línea en el procedimiento de suceso para reemplazar a la anterior:

```
MsgBox(Nombre), , "Resultado de la introducción de datos"
```

   Esta nueva sentencia llamará a la función MsgBox, mostrará el contenido de la variable Nombre en el cuadro de diálogo y presentará las palabras *Resultado de la introducción de datos* en la barra de títulos. (En el presente ejemplo se han omitido el argumento *NúmeroDeBotones* y la variable BotónPulsado). El contenido del procedimiento de suceso deberá ser similar al mostrado a continuación:



Nota: Cuando no se incluye la variable BotónPulsado, los paréntesis sólo rodean al primer argumento.

Botón Iniciar

5. Pulse el botón Iniciar de la barra de herramientas.

6. Pulse el botón Introducir, escriba su nombre en el cuadro de entrada y pulse después Aceptar.

   La entrada se guardará en el programa dentro de la variable Nombre y después se mostrará en un cuadro de mensaje tal y como se muestra en la siguiente figura:

7. Pulse Aceptar para cerrar el cuadro de mensajes y después pulse el botón Salir para cerrar el programa.

   El programa se cerrará y volverá a aparecer el entorno de programació

8. Guarde el formulario y el proyecto con el nombre **MiSalida** para mantener una copia de su programa.

MANEJO DE TIPOS ESPECÍFICOS DE DATOS

Si una variable va a contener siempre un mismo tipo de dato podrá mejorar el funcionamiento del programa declarando la variable como de dicho tipo

En la mayoría de los casos, el tipo de dato denominado Variant será el único que necesite. Las variables del tipo Variant pueden almacenar cualquier tipo de dato utilizado en Visual Basic (predefinidos) y cambiar de formato automáticamente. Las variables del tipo Variant también son sencillas de utilizar y no tendrá que pensar demasiado en el tamaño de la variable cuando sea declarada. Sin embargo, si desea crear un código especialmente rápido y conciso, deberá utilizar tipos de datos específicos cuando así lo requiera el programa.

Por ejemplo, si una variable siempre va a contener valores enteros de pequeña cuantía (números sin decimales), podrá ahorrar espacio en memoria cuando ejecute su programa declarando la variable como entera, en lugar de hacerlo como variante. Una variable entera (Integer) también acelerará las operaciones aritméticas, de forma que obtendrá un pequeño incremento en la velocidad del programa cuando sea utilizada.

En el cuadro que vemos a continuación se muestran los tipos de datos fundamentales existentes en Visual Basic. En el siguiente ejercicio veremos cómo fu cionan la mayoría de estos tipos de datos.

El tamaño de almacenamiento de las variables se mide en bytes, es decir, la cantidad de espacio necesario para almacenar ocho bits de información (aproximadamente un carácter)

Tipo de datos	Tamaño	Rango	Ejemplo de uso
Integer (entero)	2 bytes	-32.768 a 32.767	Dim Pájaros% Pájaros%=37
Long Integer (entero largo)	4 bytes	-2.147.483.648 a 2.147.483.647	Dim Ingresos& Ingresos& =350.000
Single Precision (Coma flotante de simple precisión)	4 bytes	-3,042823E38 a 3,042823E38	Dim Precio! Precio! =899,99
Double Precision (Coma flotante de doble precisión)	8 bytes	-1,79769313486232D308 a 1,79769313486232D308	Dim Pi# Pi# =3,1415926535
Currency (Monetario)	8 bytes	-922337203685477,5808 a 922337203685477,5807	Dim Deuda@ Deuda@ =7600300,50
String (Cadena)	1 byte por carácter	0 a 65.535 caracteres	Dim PerroS PerroS =«Cocker»
Boolean (Buleano)	2 bytes	True o False	Dim Bandera as Boolean Bandera =True
Date	8 bytes	1 Enero 100 hasta	Dim Aniversario as Date
(Fecha)		31 diciembre 9999	Aniversario=#3-1-63#
Variant	16 bytes (Con números) 22 bytes + 1 byte por carácter (con cadenas)	Todo tipo de rangos	Dim Total Total = 289,13

Truco: Podrá especificar algunos tipos de datos fundamentales añadiendo al nombre de la variable un carácter de declaración de tipo. Por ejemplo, podrá declarar una variable de tipo entero añadiendo el carácter de porcentaje al final de su nombre. De esta forma, en Visual Basic las dos declaraciones siguientes son equivalentes:

```
Dim I As Integer
Dim I%
```

Uso de los tipos de datos fundamentales en el código

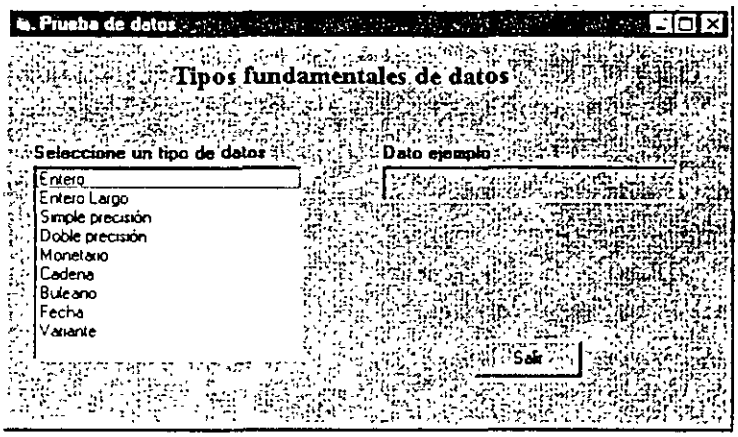
El programa Datos muestra el empleo de los tipos de datos más importantes de Visual Basic

1. En el menú Archivo seleccione la opción Abrir proyecto.
   Aparecerá el cuadro de diálogo Abrir proyecto.
2. Abra el proyecto Datos contenido en la carpeta \Vb5Sbs\Less04.
   El proyecto Datos se abrirá en el entorno de programación. Datos es un programa completo de Visual Basic que muestra el funcionamiento

de varios tipos de datos fundamentales. Ejecute el programa para ver el aspecto que tienen los tipos de datos. Finalmente, analizaremos el modo en que se declaran las variables y cómo se utilizan en el código de programa.

Botón Iniciar

3. Pulse el botón Iniciar contenido en la barra de herramientas.
   Aparecerá la siguiente ventana de aplicación:



El programa Datos le permitirá probar nueve tipos de datos, incluyendo Entero, Coma flotante de precisión simple y Fecha. El programa mostrará un ejemplo de cada uno de los tipos elegidos cuando seleccione su nombre en el cuadro de lista.

4. Seleccione el tipo Entero en el cuadro de lista.
   En el cuadro Dato Ejemplo aparecerá el número 37.

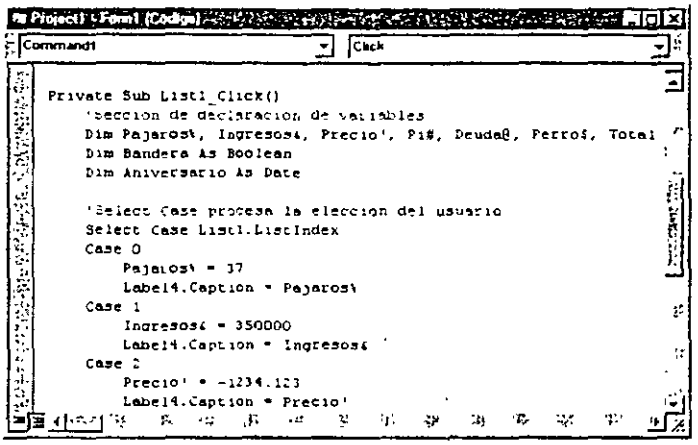
5. Seleccione el tipo Fecha en el cuadro de lista.
   En el cuadro Dato Ejemplo aparecerá la fecha martes, 19 noviembre, 1963.

6. Pulse cada uno de los tipos de datos de cuadro de lista para ver cómo los muestra Visual Basic en el cuadro Dato Ejemplo.

7. Pulse el botón Salir para detener programa.
   Examine ahora la declaración de los tipos de datos fundamentales y su uso en el procedimiento de suceso List1_Click.

8. Resalte el formulario en la ventana Proyecto, y pulse el botón Ver Objeto.

9. Pulse dos veces el objeto cuadro de lista del formulario y aumente el tamaño de la ventana Código para ver todo el código posible.
   El procedimiento de suceso List1_Click aparecerá en la ventana Código.



Las primeras líneas del procedimiento han sido utilizadas para declarar variables con tipos de datos específicos. Dichas variables serán *locales* para el procedimiento: no tendrán ningún significado en otros procedimientos de suceso contenidos en el programa. Algunas de las variables han sido declaradas utilizando caracteres especiales que sirven para especificar los tipos de datos, por ejemplo, %, # y @. Estos caracteres identifican cada una de las variables como un tipo de dato fundamental y marcan las variables para hacerlas comprensibles tanto por el compilador de Visual Basic como para cualquiera que lea el código del programa.

La siguiente sección del procedimiento de suceso es una estructura de decisión Select Case. En el siguiente capítulo estudiaremos cómo selecciona una opción este grupo de sentencias de programa. De momento, puede observar cómo en cada una de las secciones de la estructura Select Case se asigna un valor a cada una de las variables de tipo de dato fundamental. Finalmente, se asigna la variable al rótulo del objeto Label4 contenido en el formulario. Podrá utilizar cualquiera de estas dos técnicas para manipular los tipos de datos fundamentales en sus propios programas.

El tipo de datos Date (fecha) resulta especialmente útil si piensa trabajar habitualmente con valores de fecha y hora. La fecha se asigna a la variable Aniversario encerrada por caracteres almohadilla (#) y se le asigna formato mediante la función Format.

Nota: Las variables también pueden ser globales, *es decir, estar disponibles (ser públicas) para todos los procedimientos y módulos de un programa. (Los* módulos *son archivos especiales que contienen declaraciones y procedimientos no asociados con un formulario en particular). Para que una variable tenga este rango o* alcance *tiene que ser declarada en un módulo estándar. Si desea obtener más información sobre la creación de una variable global en módulos estándar consulte el capítulo 9, «Empleo de módulos y procedimientos».*

---

10. Desplácese por la ventana Código y examine detenidamente cada una de las asignaciones de variables.

    Si lo desea, puede cambiar los datos en algunas de las sentencias de asignación de variables y volver a ejecutar el programa para ver el resultado obtenido.

11. Cuando haya terminado, cierre la ventana Código.

    Si ha realizado alguna modificación que desee guardar en disco, pulse el botón Guardar proyecto de la barra de herramientas.

Tipos de datos definidos por el usuario

Visual Basic también le permitirá crear sus propios tipos de datos. Esta posibilidad le será de utilidad cuando trabaje con un grupo de datos que se ajusten naturalmente entre ellos, pero que pertenezcan a diferentes tipos de datos. Para crear un tipo de dato definido por el usuario deberá utilizar la palabra clave Type y podrá crear variables de este nuevo tipo utilizando la sentencia Dim. (La sentencia Type deberá encontrarse en la sección Declaraciones de un módulo estándar; si desea obtener más información acerca de los módulos estándar deberá buscar la palabra *Módulos* en la ayuda interactiva de Visual Basic). Por ejemplo, la siguiente declaración crea un tipo de datos definido por el usuario denominado Empleado, que puede almacenar el nombre, la fecha de nacimiento y la fecha de alta asociada con un empleado:

```
Type Empleado
    Nombre As String
    Nacimiento As Date
    FechaAlta As Date
End Type
```

Después de crear un nuevo tipo de datos podrá utilizarlo en el código de programa. Las siguientes sentencias hacen uso del nuevo tipo de datos denominado Empleado. La primera sentencia crea una variable denominada DirectorProducto, de tipo Empleado, y la segunda sentencia asigna el nombre «Jorge Enrique» a la componente Nombre de dicha variable:

```
Dim DirectorProducto As Empleado
DirectorProducto.Nombre = "Jorge Enrique"
```

La forma de proceder resultará similar a asignar un valor a una propiedad ¿Verdad? Visual Basic utiliza la misma notación para la relación entre objetos y propiedades que para la relación entre tipos de datos definidos por el usuario y variables componentes.

Constantes: variables que no cambian

Si una variable de su programa contiene un valor que no va a cambiar (tal como π, que es un número real fijo) puede interesarle guardar dicho valor como constante en lugar de hacerlo como variable. Una *constante* es un nombre significativo que ocupa el lugar de un número o una cadena de texto que no cambia. Las constantes son útiles porque facilitan la legibilidad del código de programa, ahorran memoria y facilitan la realización posterior de modificaciones globales. Las constantes funcionan de forma muy similar a las variables, con la diferencia de que no podrá modificar su valor en tiempo de ejecución. Se declaran con la palabra clave Const, tal como se muestra en el siguiente ejemplo:

```
Const Pi = 3,14159265
```

La sentencia anterior crea una constante llamada Pi, que podrá utilizarse en lugar del valor de π en el código de programa. Para crear una constante que esté disponible en todos los procedimientos de un programa deberá crearla en un módulo estándar colocándole delante la palabra clave Public. Por ejemplo:

```
Public Const Pi = 3,14159265
```

Nota: Si desea obtener más información sobre los módulos estándar consulte el Capítulo 9 «Empleo de módulos y procedimientos».

El siguiente ejercicio le mostrará cómo utilizar una constante en un procedimiento de suceso.

Empleo de una constante en un procedimiento de suceso

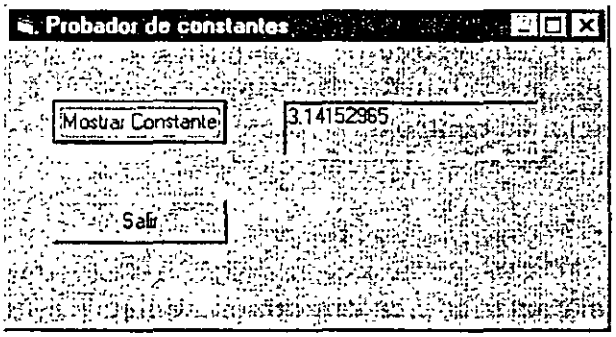
1. En el menú Archivo seleccione la opción Abrir proyecto.
   Aparecerá el cuadro de diálogo Abrir proyecto.
2. Abra el proyecto Constante de la carpeta \Vb5Sbs\Less04.
3. Pulse el botón Ver objeto contenido en la ventana Proyecto.
   El formulario asociado con el programa Constante aparecerá en la pantalla. Constante es un esqueleto de programa. La interfaz de usuario está terminada, pero deberá escribir el código de programa.
4. Pulse dos veces el botón Mostrar constante del formulario.
   En la ventana Código aparecerá el procedimiento de suceso Command1_Click.
5. Escriba las siguientes sentencias en el procedimiento de suceso:

```
Const Pi = 3.14159265
Label1.Caption = Pi
```

Botón Iniciar

6. Pulse el botón Iniciar para ejecutar el programa.
7. Pulse el botón Mostrar constante del formulario.

   El valor de la constante Pi aparecerá en el cuadro de etiqueta tal y como se muestra a continuación:



8. Pulse el botón Salir para detener el programa.

   Si desea almacenar una copia del programa Constante modificado, guarde el formulario y el proyecto con el nombre **MiConstante**.

Las constantes resultan muy útiles en el código del programa, especialmente cuando se utilizan en fórmulas matemáticas, como por ejemplo Área = $2\ \pi r^2$. En siguiente apartado se muestra el modo de utilizar operadores y variables para escribir fórmulas similares a ésta.

MANEJO DE LOS OPERADORES DE VISUAL BASIC

Los operadores de Visual Basic se utilizarán exclusivamente en fórmulas.

Una *fórmula* es una sentencia que combina números, variables, operadores y palabras clave, o varios de estos elementos, para crear un nuevo valor. Visual Basic cuenta con bastantes elementos de lenguaje diseñados para ser utilizados en fórmulas. En este apartado practicará en el manejo de los operadores matemáticos, es decir, los símbolos utilizados para enlazar las partes de una fórmula. Salvo algunas excepciones, los símbolos matemáticos que mostraré aquí son los que habitualmente utilizará en su vida diaria y sus operaciones son realmente intuitivas. En los siguientes ejercicios le mostraré un ejemplo del uso de cada uno de ellos.

Visual Basic proporciona los siguientes operadores:

Operador	Operación matemática
+	Suma
–	Sustracción (resta)
*	Multiplicación

(contin.

Operador	Operación matemática *(continuación)*
/	División
\	División entera
Mod	Resto de la división entera
^	Exponenciación (elevar a una potencia)
&	Concatenación de cadenas (combinación)

Matemáticas básicas: operadores +, –, * y /

Los operadores de suma, resta, multiplicación y división son realmente sencillos y pueden utilizarse en cualquier fórmula en la que se empleen números o variables numéricas. En el siguiente ejercicio se muestra su uso en un programa.

Trabajo con operadores básicos

1. En el menú Archivo seleccione la opción Abrir proyecto.
2. Abra el proyecto OpBásicos de la carpeta \Vb5Sbs\Less04.
3. Seleccione el formulario dentro de la ventana Proyecto y pulse el botón Ver Objeto.

   En la ventana aparecerá el formulario del programa OpBásicos (Operadores básicos). El programa OpBásicos muestra el funcionamiento de los operadores de suma, resta, multiplicación y división cuando el usuario introduce números por el teclado. También muestra cómo se pueden utilizar cuadros de texto, botones de opción y botones de orden para procesar la entrada introducida por el usuario en un programa.
4. Pulse el botón Iniciar contenido en la barra de herramientas.

   El programa OpBásicos se ejecutará en el entorno de programación. El programa mostrará dos cuadros de texto en los que deberá introducir valores numéricos, un grupo de botones de opción con operadores matemáticos, un cuadro de resultado y dos botones de órdenes.

Los objetos de cuadro de texto son herramientas de gran utilidad para obtener una entrada de datos del usuario a través del teclado.

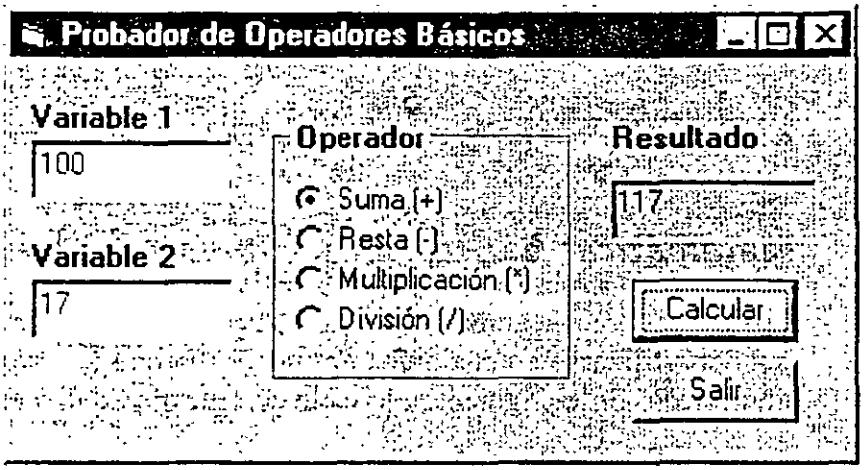
5. Escriba **100** en el cuadro de texto Variable 1 y pulse después TAB.

   El cursor se desplazará al segundo cuadro de texto.
6. En el cuadro de texto Variable 2 escriba **17**.

   Ahora podrá aplicar cualquiera de los operadores matemáticos mostrados a los valores contenidos en estos cuadros de texto.

7. Pulse el botón de opción Suma y después pulse el botón de orden Calcular.

   Este operador se aplicará a los dos valores y el número 117 aparecerá en el cuadro Resultado, tal como se muestra en la siguiente figura:



8. Practique utilizando los operadores de resta, multiplicación y división contenidos en este formulario (pulse Calcular para calcular cada uno de los resultados).

   Los resultados aparecerán en el cuadro Resultado. No dude en introducir otros números en los cuadros de texto variables. (Si lo desea, utilice números con decimales).

9. Cuando haya terminado de realizar cálculos, pulse el botón Salir.

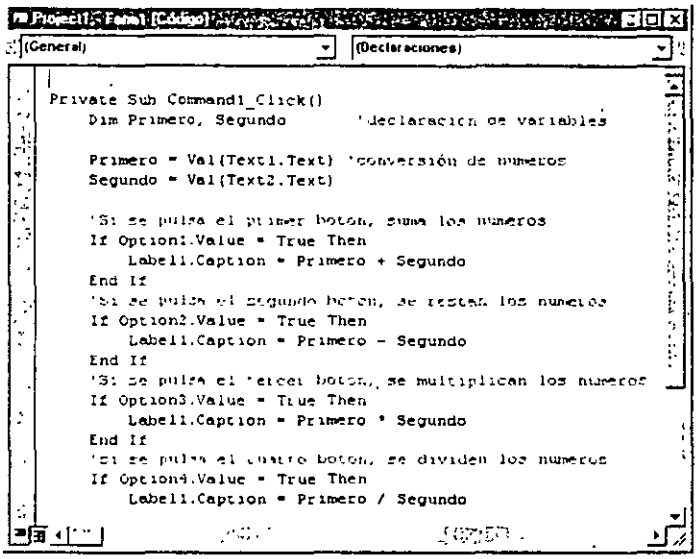
   El programa se detendrá y volverá el entorno de programación.

Eche ahora un vistazo al código de programa para ver cómo se han calculado los resultados. El programa OpBásicos utiliza algunos de los controles de entrada estándar que vimos en el Capítulo 2 y un procedimiento de suceso que utiliza variables y operadores para calcular sencillas fórmulas matemáticas. El procedimiento utiliza también la función Val para convertir en números los rótulos introducidos en los cuadros de texto.

Examen del código del programa Operadores Básicos

1. Realice una doble pulsación sobre el botón Calcular contenido en el formulario.

   El procedimiento de suceso Command1_Click aparecerá en la ventana Código tal como se muestra a continuación:



La primera sentencia del procedimiento declara dos variables de propósito general de tipo variante. Estas dos variables se utilizarán para albergar los valores introducidos en los dos cuadros de texto y serán lo suficientemente flexibles como para manejar cualquier tipo de datos numéricos que queramos utilizar. Las dos sentencias siguientes asignan los datos contenidos en los cuadros de texto a las variables y convierten las cadenas de texto en números utilizando la función Val:

```
Primero = Val(Text1.Text) 'conversión de números
Segundo = Val(Text2.Text)
```

La función Val convierte valores de texto en valores numéricos.

La función Val es una rutina especial que convierte un argumento de texto en un valor numérico. Esta conversión es necesaria para que la operación de suma funcione correctamente en este programa. Por defecto, el tipo de dato que devuelve un cuadro de texto es texto. Esto no es un problema para tres de los operadores mostrados. Los operadores –,* y / sólo trabajan con números, de forma que cuando el usuario seleccione uno de estos tres operadores en el programa, Visual Basic convertirá automáticamente en números los valores asignados a las variables Primero y Segundo.

Sin embargo, el operador + funciona tanto con números como con cadenas de texto. Como el tipo de dato devuelto por el cuadro de texto es texto, Visual Basic tratará automáticamente a los valores Primero y Segundo como texto cuando utilice el operador +. Visual Basic combinaría o *concatenaría* los dos valores en lugar de realizar su suma aritmética (en otras palabras: «100 + 17» generaría «10017»).

En el siguiente ejercicio le mostraré más detalles acerca de la concatenación de cadenas. Por ahora, sólo tendrá que recordar que aunque el tipo de datos Variante pueda albergar cualquier tipo de datos fundamental, ten-

drá que utilizarlo con especial cuidado en sus programas. En caso contrario, puede que los resultados no sean los esperados.

Importante: Siempre conviene comprobar todos los cálculos contenidos en un programa para verificar que el programa completo funciona correctamente. No basta con comprobar una parte del programa.

---

2. Desplácese por la ventana Código y examine las cuatro fórmulas que utilizan operadores matemáticos básicos.

   La primera fórmula del procedimiento utiliza el operador Suma (+) dentro de una estructura de decisión If...Then

```
'Si se pulsa el primer botón, suma los números
If Option1.Value = True Then
   Label1.Caption = Primero + Segundo
End If
```

   Si la propiedad Value del primer botón de opción se define como True (es decir, si se ha pulsado el botón), entonces las dos variables se sumarán entre sí mediante el operador + y el resultado se asignará a la etiqueta. Las tres fórmulas restantes tienen una lógica similar, cada una de las cuales utiliza una estructura de decisión If...Then y la propiedad Caption del objeto Label1. Las estructuras de decisión similares a If...Then son extremadamente útiles a la hora de determinar la opción seleccionada por un usuario dentro de un programa en el caso de que haya varias opciones disponibles. En el siguiente Capítulo aprenderemos más sobre If...Then.

3. Cierre la ventana Código.

Ya hemos terminado con el programa Operadores Básicos.

Funciones matemáticas en Visual Basic

Siempre será de interés experimentar un poco con los programas mostrados aquí. Puede que le interese convertir un valor a un tipo diferente, calcular una expresión matemática compleja o introducir números aleatorios en sus programas. Las siguientes funciones de Visual Basic le permitirán llevar a cabo con rapidez ciertas operaciones matemáticas complejas. Al igual que con cualquier otra función, las funciones matemáticas necesitan utilizarse dentro de una sentencia de programa y devolverán un valor al programa. En la siguiente tabla, el argumento (*n*) representa el número, variable o expresión que será evaluado por la función.

(continúa)

Funciones matemáticas en Visual Basic *(continuación)*

Función	Propósito
Abs(*n*)	Calcula el valor absoluto de *n*.
Atn(*n*)	Calcula el arcotangente de *n* en radianes.
Cos(*n*)	Calcula el coseno del ángulo *n*. El ángulo *n* se expresa en radianes.
Exp(*n*)	Calcula la constante e elevada a *n*.
Rnd(*n*)	Genera un número aleatorio entre 0 y 1.
Sgn(*n*)	Devuelve –1 si *n* es menor que cero, 0 si *n* es cero y +1 si *n* es mayor que cero.
Sin(*n*)	Calcula el seno del ángulo *n*. El ángulo *n* se expresa en radianes.
Sqr(*n*)	Calcula la raíz cuadrada de *n*.
Str(*n*)	Convierte un valor numérico en una cadena.
Tan(*n*)	Calcula la tangente del ángulo *n*. El ángulo *n* se expresa en radianes.
Val(*n*)	Convierte una cadena a valor numérico.

Empleo de operadores avanzados: \, Mod, ^ y &

Además de los cuatro operadores matemáticos básicos, Visual Basic incluye cuatro operadores avanzados que realizan división entera (\), resto de la división entera (Mod), exponenciación (^) y concatenación de cadenas (&). Estos operadores resultan útiles en fórmulas matemáticas de propósito especial y en aplicaciones de procesamiento de textos. La siguiente utilidad (una pequeña modificación del programa OpBásicos) muestra el funcionamiento de cada uno de estos operadores dentro del programa.

Manejo de operadores avanzados

1. En el menú Archivo seleccione la opción Abrir proyecto.
2. Abra el proyecto OpAvdos (Operadores Avanzados) contenido en la carpeta \Vb5Sbs\Less04.
3. Resalte el formulario contenido en la ventana Proyecto y pulse el botón Ver Objeto.

   En la pantalla aparecerá el formulario del programa OpAvdos. El programa Operadores Avanzados es idéntico al programa Operadores Básicos, con la excepción de que los operadores mostrados en los botones de opción son distintos.

4. Pulse el botón Iniciar de la barra de herramientas.

   El programa mostrará dos cuadros de texto en los que podrá introducir valores numéricos; también muestra un grupo de botones de opción con distintos operadores matemáticos asociados, un cuadro de resultados y dos botones de órdenes.

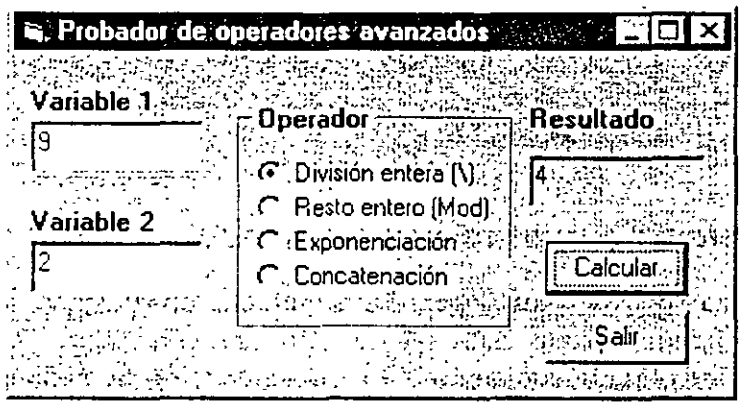
5. En el cuadro de texto Variable 1 escriba **9** y después pulse TAB.

6. En el cuadro de texto Variable 2 escriba **2**.

   Ahora podrá aplicar cualquiera de los Operadores avanzados a los valores contenidos en los cuadros de texto.

7. Pulse el botón de opción denominado División entera y después pulse el botón Calcular.

   El operador se aplicará a los dos valores, y en el cuadro de resultados aparecerá el número 4, tal como se muestra a continuación:



La división entera da como resultado la parte entera del cociente resultante de la división. Aunque 9 dividido entre 2 es igual a 4,5, la operación de división entera sólo devuelve la parte entera (el número 4). Este tipo de operación puede resultarle de utilidad si trabaja con cantidades que carecen de sentido si se expresan de forma decimal, por ejemplo, el número de personas que caben en un coche.

8. Pulse el botón de opción Resto y después pulse el botón Calcular.

   En el cuadro Resultado aparecerá el número 1. El resto de la división entera (módulo aritmético) devuelve el resto (la parte que no se divide) resultante de la división entera de dos números. Como 9 dividido entre 2 es igual a 4 con un resto de 1 ($2 \times 4 + 1 = 9$), el resultado producido por el operador Mod es 1. Además de añadir a sus programas un cierto sabor a las canciones de los años sesenta, el operador Mod puede ayudarle a calcular las partes enteras sobrantes en ciertos cálculos, como saber el número de pasteles sobrantes cuando se reparten equitativamente 14 pasteles entre comensales (cada uno tocaría a tres y sobrarían dos).

9. Seleccione la opción Exponenciación y después pulse el botón Calcular.

   En el cuadro Resultado aparecerá el número 81. El operador exponenciación (^) eleva un número a una potencia. Como 9 al cuadrado es 81, el resultado producido por el operador ^ es 81. En una fórmula de Visual Basic, 9 al cuadrado se escribe 9 ^ 2.

10. Seleccione el botón Concatenación y después pulse el botón Calcular.

    En el cuadro Resultado aparecerá la cadena «92». El Operador de concatenación de cadenas (&) combina dos cadenas en una fórmula. El resultado (en este caso «92») no es un número, es una combinación de los caracteres 9 y 2. La concatenación de cadenas sólo puede unir variables de texto, cadenas delimitadas por comillas y variables del tipo Variante. Como las variables utilizadas en este programa son variantes, se han convertido automáticamente a texto para poder realizar la operación. Para ver cómo funciona este método con rótulos, escriba alguna palabra en cada uno de los cuadro de entrada y, finalmente, vuelva a pulsar el botón Calcular.

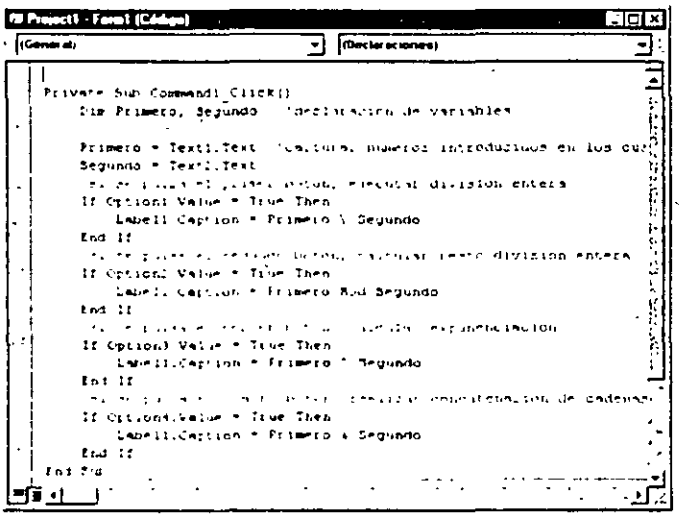
11. Pulse el botón Salir para detener el programa.

    El programa se detendrá y volverá a aparecer el entorno de programación.

    Eche ahora un vistazo al procedimiento de suceso Command1_Click para ver cómo se han utilizado los operadores.

12. Pulse dos veces el botón Calcular en el formulario.

    El procedimiento de suceso aparecerá en la ventana Código tal como se muestra en la siguiente figura:



El procedimiento Command1_Click presenta un aspecto similar al procedimiento Command1_Click del programa Operadores Básicos. En el código se declaran dos variables variantes, se asignan datos a las variables

desde los cuadros de texto y se calcula la fórmula seleccionada con estructuras de decisión Inf...Then.

No obstante, existe una importante diferencia: este procedimiento de suceso no utiliza la función Val para convertir los datos a tipo numérico cuando los lee de los cuadros de texto. La conversión no es necesaria para los operadores avanzados, ya que, a diferencia del operador +, cada uno de los operadores avanzados trabaja sólo con un tipo de datos: \, Mod y ^, sólo trabajan con números; & sólo trabaja con texto. Debido a que en este caso no hay ambigüedad, las variables variantes pueden convertirse adecuadamente en las cadenas devueltas por los cuadros de texto en las operaciones que requieran el uso de los números.

13. Pulse la ventana Código.

    Ya hemos terminado de trabajar con el programa Operadores Avanzados.

Prioridad de los operadores

En los dos últimos ejercicios hemos manejado siete operadores matemáticos y un operador de cadenas. Visual Basic permite mezclar en una fórmula tantos operadores matemáticos como se quiera siempre que las variables numéricas y expresiones estén separadas entre sí por operadores. Por ejemplo, ésta es una fórmula aceptable en Visual Basic:

```
Total = 10 + 15 * 2 / 4 ^ 2
```

La fórmula procesa varios valores y asigna el resultado a una variable llamada Total. Pero ¿cómo se evalúa dicha expresión en Visual Basic? En otras palabras, ¿qué operaciones realizará en primer lugar Visual Basic? Puede que no se haya dado cuenta, pero el orden de evaluación tiene gran importancia en este ejemplo.

Deberá tener en cuenta el orden de evaluación de los operadores cuando esté creando fórmulas matemáticas.

Visual Basic resuelve este dilema estableciendo un *orden de prioridad* específico para las operaciones matemáticas. Esta lista de reglas indica a Visual Basic qué operadores deberá utilizar primero cuando evalúe una expresión que contenga más de un operador. En la siguiente tabla se muestran los operadores matemáticos listados en orden de prioridad (los operadores de igual nivel en esta tabla se evalúan de izquierda a derecha, según aparecen en la expresión).

Operador(es)	Orden de prioridad
()	Los valores entre paréntesis se evalúan siempre los primeros.
^	La exponenciación (elevar un número a una potencia) es siempre el segundo.
–	La negación (creación de un número negativo) es el tercero.

(contin.

Operador(es)	Orden de prioridad *(continuación)*
* /	La multiplicación y la división comparten el cuarto lugar.
\	La división entera es el quinto operador.
Mod	El resto es el sexto.
+ –	Los últimos son la suma y la resta.

Teniendo en cuenta el orden de prioridad mostrado en la tabla anterior, la expresión:

```
Total = 10 + 15 * 2 / 4 ^ 2
```

se evaluará de la siguiente forma. (Las negritas indican el orden de evaluación y el resultado):

```
Total = 10 + 15 * 2 / 4 ^ 2
Total = 10 + 15 * 2 / 16
Total = 10 + 30 / 16
Total = 10 + 1,875
Total = 11,875
```

UN PASO MÁS ALLÁ: EMPLEO DE PARÉNTESIS EN LAS FÓRMULAS

Los paréntesis clarifican e influyen en el orden de la evaluación de una fórmula.

Podemos utilizar uno o más pares de paréntesis en una fórmula para clarificar el orden de prioridad. Por ejemplo, Visual Basic calcularía la fórmula

```
Número = ( 8 - 5 * 3 )^ 2
```

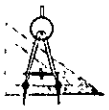
realizando en primer lugar la operación contenida entre los paréntesis (–7) antes de realizar la exponenciación, aunque la exponenciación tenga un orden de prioridad mayor que la resta y la multiplicación. Podemos refinar aún más el cálculo colocando paréntesis anidados en la fórmula. Por ejemplo:

```
Número = ( ( 8 - 5 ) * 3 ) ^ 2
```

hace que Visual Basic calcule en primer lugar la diferencia contenida en el conjunto de paréntesis más interno, posteriormente trabaja con los paréntesis externos para calcular, finalmente, el exponente. El resultado producido por las dos fórmulas es diferente: la primera fórmula da como resultado 49 y la segunda 81. Los paréntesis pueden modificar los resultados de una operación matemática además de facilitar su lectura.

Si desea continuar con el siguiente capítulo

➤ No salga de Visual Basic y pase al Capítulo 5.



Si desea salir de Visual Basic por ahora

➤ En el menú Archivo seleccione Salir.
Si en su pantalla aparece un cuadro de diálogo que le permite almacenar los cambios, seleccione Sí.

RESUMEN DEL CAPÍTULO

Para	Haga esto

Declarar una variable	Escriba **Dim** seguido del nombre de la variable en el código de programa. (También puede especificar un tipo de datos; el tipo implícito será variante). Por ejemplo:
	`Dim Número%  'tipo Integer`
	`Dim Almacenamiento  'tipo variante`

Cambiar el valor de una variable	Asigne un nuevo valor con el operador de asignación (=). Por ejemplo: `País = "Japón"`
Obtener una entrada mediante un cuadro de diálogo	Utilice la función InputBox y asigne el resultado a una variable. Por ejemplo: `NombreUsuario = InputBox("¿Cómo se llama?")`

Mostrar una salida en un cuadro de diálogo	Utilice la función MsgBox. (La cadena que se mostrará en el cuadro de diálogo puede guardarse en una variable.) Por ejemplo:
	`Previsión = "Lloverá en ambas mesetas"`
	`MsgBox(Previsión), ,"Previsión climatológica para España"`

Declarar una variable de un tipo especial de datos	Escriba **Dim** seguido del nombre de la variable y del carácter de tipo
	o bien
	Escriba **Dim** seguido del nombre de la variable, la palabra clave **As** y uno de los ocho tipos de datos fundamentales. Por ejemplo:
	`Dim Aniversario As Date 'tipo fecha`
	`Dim Precio! ' coma flotante simple precisión`

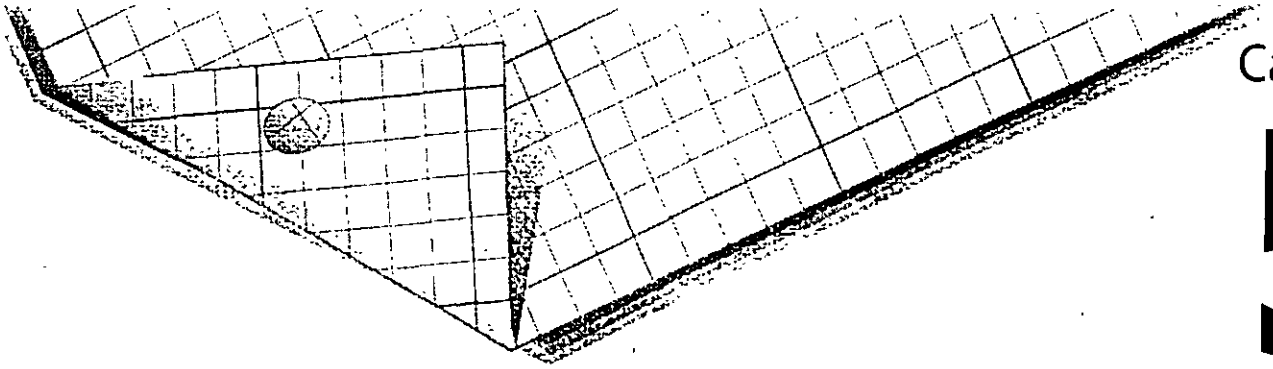
Para	Haga esto
Crear una constante	Escriba la palabra reservada **Const** seguida por el nombre de la constante, el operador de asignación (=) y el valor asignado. Por ejemplo: `Const EdadPepe = 39`
Crear una fórmula	Enlace las variables numéricas o los números mediante alguno de los siete operadores aritméticos y asigne después el resultado a una variable o una propiedad. Por ejemplo: `Resultado = 1 ^ 2 * 3 \ 4 'esto equivale a 0`
Combinar cadenas de texto	Utilice el operador de concatenación (&). Por ejemplo: `Mensaje = "Hola" & "," & "amigos"`
Convertir caracteres textuales en caracteres numéricos	Utilice la función Val. Por ejemplo: `Pi = Val("3.14159265358979 32")`
Empleo de funciones matemáticas	Añada a la fórmula la función y los argumentos que necesite. Por ejemplo: `Hipotenusa = Sqr(x ^ 2 + y ^ 2)`

Control del orden de evaluación en una fórmula	Utilice paréntesis en la fórmula. Por ejemplo:
	`Resultado = 1 + 2 ^ 3 \ 4 'esto equivale a 3`
	`Resultado = ( 1 + 2 ) ^ ( 3 \ 4 ) 'esto equivale a 1`

Sobre la ayuda en línea	En el menú Ayuda de Visual Basic, seleccione el mandato Temas de ayuda de Microsoft Visual Basic, seleccione la etiqueta Índice y a continuación
Declaración de variables	Escriba **Variables, declarar**
Tipos de datos fundamentales	Escriba los símbolos de los tipos de datos que desee examinar.
Tipos de datos definidos por el usuario	Escriba **tipos de datos, definidos por el usuario**
Empleo de la función InputBox	Escriba **InputBox**
Empleo de la función MsgBox	Escriba **MsgBox**
Operadores y prioridad	Escriba el operador que desee examinar
Funciones matemáticas	Escriba el nombre de la función que desee examinar

AVANCE DEL SIGUIENTE CAPÍTULO

En el siguiente capítulo, «Empleo de estructuras de decisión», aprenderá a controlar el flujo del programa en sus procedimientos de suceso. Aprenderá a utilizar las sentencias If...Then y Select Case y a ejecutar diferentes bloques de código dependiendo de las diferentes condiciones del programa. También aprenderá a localizar los errores lógicos y los errores sintácticos en un programa y a corregirlos.



Capítulo

5

Empleo de estructuras de decisión

En los capítulos anteriores hemos utilizado varias herramientas de Microsoft Visual Basic para procesar la entrada introducida por el usuario. Se han utilizado menús, objetos y cuadros de diálogo para mostrar opciones al usuario y se ha procesado la entrada utilizando propiedades y variables. En este capítulo le mostraré cómo puede su programa ejecutar una serie de instrucciones u otras dependiendo de la entrada introducida por el usuario. Le mostraré cómo evaluar una o más propiedades o variables mediante el uso de expresiones condicionales y a ejecutar, posteriormente, una o más sentencias de programa basándose en los resultados. También aprenderá a detectar y corregir errores de programación en su código utilizando el modo de Depuración.

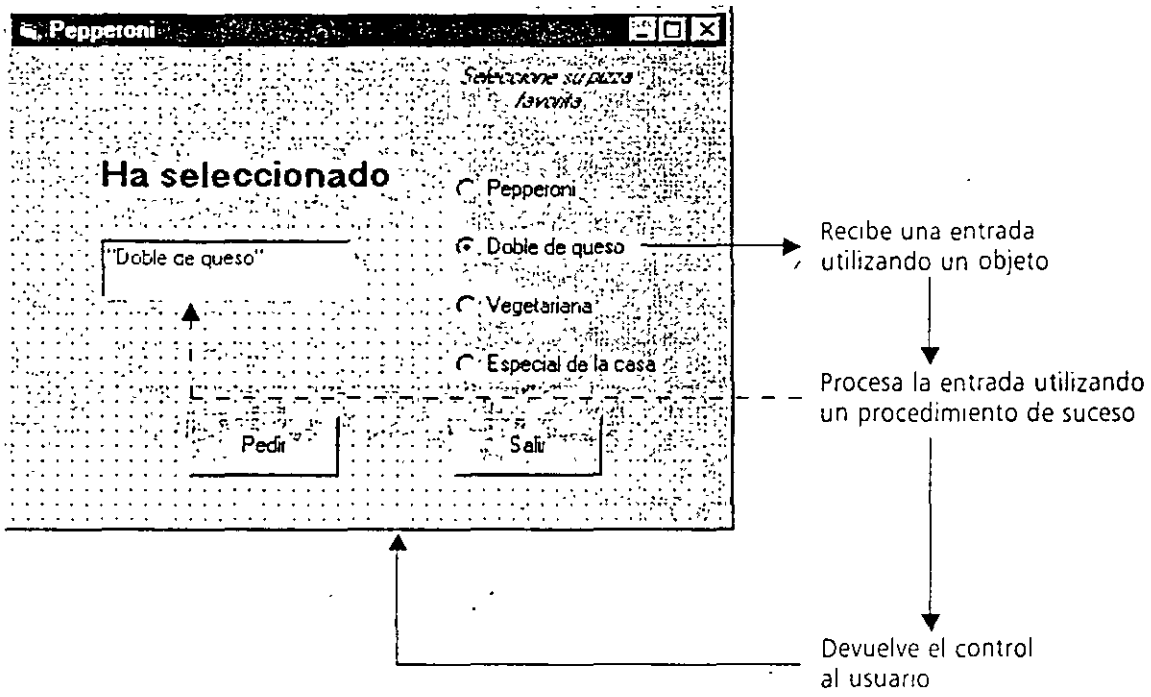
PROGRAMACIÓN ORIENTADA A SUCESO

Los programas de Visual Basic están orientados a suceso.

Los programas que ha escrito hasta ahora muestran menús, objetos y cuadros de diálogos en la pantalla y, además, dejan a los usuarios que libremente escojan el orden en que desean manipular dichos objetos.

Los programas se relacionan con el usuario, esperan pacientemente una respuesta y después procesan la entrada de forma predecible. En los ambientes de programación esta forma de trabajar se conoce como *programación orientada a suceso.* Los programas se crean partiendo de un grupo de objetos «inteligentes» que saben cómo responder cuando el usuario interactúa con ellos y, finalmente, se procesa las entradas realizadas por el usuario mediante procedimientos de suceso-

asociados con dichos objetos. En el siguiente diagrama se muestra cómo funciona un programa dirigido por sucesos en Visual Basic.

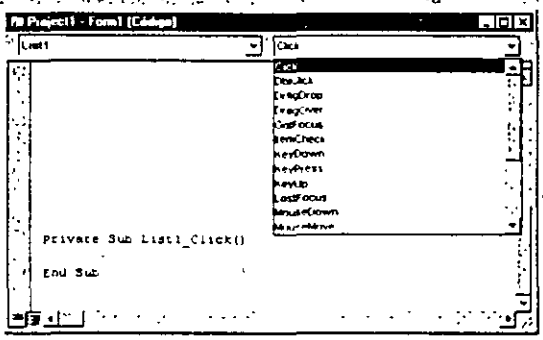


La entrada al programa también puede venir de la propia computadora. Por ejemplo, el programa podría advertir la llegada de un mensaje de correo electrónico o del momento exacto en que haya pasado una cierta cantidad de tiempo en el reloj del sistema. Estos sucesos son activados por la computadora, no por el usuario. Independientemente de quién active un suceso, Visual Basic reacciona llamando al procedimiento de suceso asociado con el objeto que reconoce dicho suceso. Aunque por ahora sus programas sólo han manejado sucesos del tipo pulsación del ratón, los objetos de Visual Basic pueden responder a muchos tipos de sucesos diferentes.

La orientación a suceso de los programas desarrollados en Visual Basic se traduce en que la mayoría de las operaciones llevadas a cabo por sus programas serán ejecutadas mediante procedimientos de suceso. Estos bloques de código procesarán las entradas, calcularán nuevos valores, mostrarán la salida y llevarán a cabo otras tareas. En el capítulo anterior le mostré cómo utilizar variables, operadores y fórmulas matemáticas para realizar cálculos en los procedimientos de suceso. En este capítulo aprenderá a utilizar *estructuras de decisión* para comparar variables, propiedades y valores y para saber cómo ejecutar una o más sentencias basándose en los resultados. En el siguiente capítulo aprenderá a utilizar *bucles* para ejecutar un grupo de sentencias una y otra vez hasta que se cumpla una condición. Conjuntamente, estas estructuras de control de flujo le permitirán crear procedimientos que puedan responder ante cualquier situación.

Sucesos controlados por los objetos de Visual Basic

Cada objeto de Visual Basic tiene un conjunto predefinido de sucesos a los que puede responder. Estos sucesos se listan en el cuadro de lista desplegable Proc (procedimiento) de la ventana Código para cada uno de los objetos contenidos en el formulario. Podrá escribir un procedimiento de suceso para cada uno de estos sucesos, y si alguno de estos sucesos resulta activado por el programa, Visual Basic ejecutará el procedimiento de suceso asociado con el mismo. Por ejemplo: un objeto cuadro de lista permite los sucesos Click, DblClick, DragDrop, DragOver, GotFocus, ItemCheck, KeyDown, KeyPress, KeyUp, LostFocus, MouseDown, MouseMove y MouseUp, OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag y Scroll. Aunque probablemente no necesite programar más de uno o dos de estos sucesos en la mayor parte de sus aplicaciones, resulta interesante saber que cuenta con un gran número de opciones a la hora de crear elementos en su interfaz de usuario. En la siguiente ilustración se muestra en la ventana de Código una lista parcial de los sucesos asociados con un objeto cuadro de lista:



EMPLEO DE EXPRESIONES CONDICIONALES

Las expresiones condicionales requieren respuestas del tipo Verdadero o Falso.

Una de las herramientas más útiles para procesar información en un procedimiento de suceso es la expresión condicional. Una *expresión condicional* forma parte de una sentencia de programa que realiza preguntas del tipo Verdadero o Falso sobre una propiedad, una variable o algún otro tipo de datos en el código del programa. Por ejemplo, la expresión condicional

```
Precio < 100
```

dará como resultado Verdadero (True) si la variable Precio contiene un valor menor que 100 y dará como resultado Falso (False) si la variable Precio contiene un

valor mayor o igual que 100. En cualquier expresión condicional podrá utilizar los operadores de comparación que se muestra a continuación:

Operador de comparación	Significado
=	Igual a
<>	Distinto que
>	Mayor que
<	Menor que
> =	Mayor o igual que
< =	Menor o igual que

Nota: Las expresiones que pueden evaluarse como Verdadero o Falso se conocen también como expresiones booleanas y los resultados Verdadero o Falso (True o False) se pueden asignar a una variable o propiedad booleana. Los valores booleanos se pueden asignar a ciertas propiedades de objetos, variables variantes o variables booleanas que hayan sido creadas mediante el uso de la sentencia Dim y de las palabras clave As Boolean.

En la siguiente tabla se muestran algunas expresiones condicionales y sus resultados. En el siguiente ejercicio trabajará con los operadores que aparecen en la tabla.

Expresión condicional	Resultado
10 < > 20	Verdadero (10 es distinto de 20)
Puntuación < 20	Verdadero si Puntuación es menor que 20; en otro caso, Falso.
Puntuación = Label1.Caption	Verdadero si la propiedad Caption del objeto Label1 contiene el mismo valor que la variable Puntuación; en caso contrario, Falso.
Text1.Text = «Guillermo»	Verdadero si la palabra *Guillermo* está en el primer cuadro de texto; en otro caso, Falso.

LA ESTRUCTURA DE DECISIÓN IF...THEN

Las estructuras de decisión If...Then permiten añadir lógica a los programas.

Las expresiones condicionales pueden controlar el orden en el que se ejecutan las sentencias, cuando se usan en un bloque especial de sentencias denominado *estructura de decisión*. Las estructuras de decisión If...Then le permitirán evaluar una condición en el programa y llevar a cabo una serie de acciones basándose

el resultado de dicha condición. En su forma más simple, una estructura de decisión If...Then cabe en una única línea

```
If condición Then sentencia
```

donde *condición* es una expresión condicional y *sentencia* es una sentencia válida de Visual Basic. Por ejemplo:

```
If Marcador >= 20 Then Label1.Caption = "¡Ha ganado!"
```

es una estructura de decisión If..Then que utiliza la expresión condicional

```
Marcador >= 20
```

para determinar si el programa deberá asignar a la propiedad Caption del objeto Label1 el valor «¡Ha ganado!». Si la variable Marcador contiene un valor mayor o igual que 20, Visual Basic asignará el valor de la propiedad Caption; en caso contrario, ignorará la sentencia de asignación y ejecutará la siguiente línea del procedimiento de suceso. Este tipo de comparación genera siempre como resultado un valor Verdadero o Falso. Una expresión condicional nunca produce otro resultado (por ejemplo, «Tal vez» o «Quizás»).

Verificación de varias condiciones en una estructura de decisión If...Then

Visual Basic cuenta también con una estructura de decisión If...Then que le permitirá manejar simultáneamente varias expresiones condicionales. Este bloque de sentencias puede ocupar varias líneas y utiliza las palabras clave ElseIf, Else y End If.

Las cláusulas ElseIf y Else permiten incluir preguntas adicionales en una estructura If. .Then.

```
If condición1 Then
    sentencias ejecutadas si condicion1 es Verdadera
ElseIf condición2 Then
    sentencias ejecutadas si condición2 es Verdadera
(Aquí se pueden poner más cláusulas ElseIf y más sentencias)
Else
    sentencias ejecutadas si ninguna de las condiciones es Verdadera
End If
```

En esta estructura, *condición1* se examina en primer lugar. Si esta expresión condicional resulta ser Verdadera (True), se ejecutará el bloque de sentencias que hay debajo, una a una (podrá incluir una o más sentencias de programa). Si la primera condición no es Verdadera, se evaluará la segunda expresión condicional *(condición2)*. Si la segunda condición es Verdadera, se ejecutará el segundo bloque de sentencias (podrá añadir condiciones y sentencias ElseIf adicionales si desea evaluar más condiciones). Por último, si ninguna de las expresiones condicio-

nales resulta Verdadera, se ejecutarán las sentencias que siguen a la palabra clave Else. La estructura completa se cierra finalmente con las palabras clave End If.

Las cláusulas ElseIf y Else múltiples son muy adecuadas para manejar valores que pueden pertenecer a diferentes rangos, tales como los ingresos percibidos y las tablas de retención de Hacienda

En el siguiente ejemplo se muestra una estructura If...Then de varias líneas que podría utilizarse para determinar los impuestos que debe pagar un determinado contribuyente en un sistema fiscal progresivo (los valores corresponden al Ministerio de Hacienda de los EE.UU. para declaraciones individuales en el año 1995).

```
If Ingresos <= 23350 Then           '15 % retención
      Impuestos = Ingresos * 0.15
ElseIf Ingresos <= 56550 Then       '28 % retención
      Impuestos = 3502 + ((Ingresos - 23350) * 0.28)
ElseIf Ingresos <= 117950 then      '31 % retención
      Impuestos = 12798 + ((Ingresos - 56550) * 0.31)
ElseIf Ingresos <=256500 then       '36 % retención
      Impuestos = 31832 + ((Ingresos - 117950) * 0.36)
Else                                '39.6 % retención
      Impuestos = 81710 + ((Ingresos - 256500) * 0.396)
End If
```

En esta estructura de decisión se va evaluando la variable Ingresos desde el primer nivel hasta encontrar un rango de ingresos que haga que la comparación resulte Verdadera. Cuando esto ocurra se determinará los impuestos que ha de pagar el contribuyente. Esta estructura de decisión, a pesar de su simplicidad, puede resultar bastante útil. Puede utilizarse para calcular los impue que debe pagar un contribuyente en un sistema de tasas progresivas (como el de los EE.UU.). El programa calculará el valor total del impuesto a pagar suponiendo que los valores de los rangos hayan sido actualizados correctamente y que el valor de la variable Ingresos que está siendo utilizada sea correcto. Si los extremos de los rangos se ven modificados bastará con actualizar las expresiones condicionales.

Importante. El orden de las expresiones condicionales en las cláusulas If...Then y ElseIf es crítico. Si invierte el orden en que se evalúan las expresiones condicionales en el ejemplo anterior (es decir, si coloca los porcentajes en orden inverso) los contribuyentes que pertenecieran a las bandas salariales correspondientes a los porcentajes 15, 28 y 31 se evaluarían como si pertenecieran a la banda del porcentaje 36, puesto que es cierto que todos ellos tienen unos ingresos inferiores a 256.500 dólares (Visual Basic se detiene en la primera expresión condicional que sea Verdadera, aunque haya más expresiones que también lo sean). Como todas las expresiones condicionales de este ejemplo evalúan la misma variable, deberán listarse en orden ascendente para poder agrupar a los contribuyentes en su categoría exacta. Moraleja: cuando utilice más de una expresión condicional, tenga cuidado con el orden.

En el siguiente ejercicio utilizará una estructura de decisión If...Then para validar a los usuarios que intenten trabajar con un determinado programa. Podrá utilizar una lógica de programación similar si desea desarrollar una aplicació red.

Validación de usuarios utilizando If...Then

1. Inicie Visual Basic
   Si Visual Basic ya está en ejecución, abra un nuevo proyecto.

Control CommandButton

2. Utilice el control CommandButton para introducir un botón de orden en la esquina superior izquierda del formulario.
3. Asigne el valor «Acceso» a la propiedad Caption del botón de orden que acaba de introducir.
4. Pulse dos veces el botón Acceso.
   El procedimiento de suceso Command1_Click aparecerá en la ventana Código.

Como norma, las sentencias comprendidas entre sentencias If...Then, ElseIf y Else se encuentran desplazadas hacia la derecha con respecto al margen izquierdo

5. Escriba las siguientes sentencias de programa en el procedimiento:

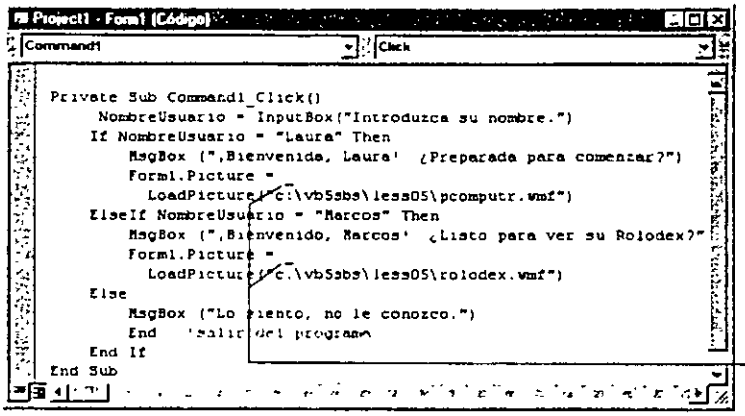
```
NombreUsuario = InputBox("Introduzca su nombre.")
If NombreUsuario = "Laura" Then
    MsgBox ("¡Bienvenida, Laura!  ¿Preparada para comenzar?")
    Form1.Picture = _
      LoadPicture("c:\vb5sbs\less05\pcomputr.wmf")
ElseIf NombreUsuario = "Marcos" Then
    MsgBox ("¡Bienvenido, Marcos!  ¿Listo para ver su
      Rolodex?")
    Form1.Picture = _
      LoadPicture("c:\vb5sbs\less05\rolodex.wmf")
Else
    MsgBox ("Lo siento, no le conozco.")
    End   'salir del programa
End If
```

(El carácter subrayado que se utiliza después de las propiedades Form1.Picture permite la división de las sentencias largas del programa en tantas líneas como sea necesario para poder representarlas en este libro. Si lo desea puede escribir cada una de estas sentencias en una única línea; la ventana Código las desplazará hacia la derecha.)

Nota: Las líneas de programa pueden tener una longitud máxima de 1.023 caracteres dentro de la ventana Código de Visual Basic, pero suele ser más sencillo trabajar con líneas de una longitud menor o igual a 80 caracteres. Las sentencias de programa que resulten de mayor tamaño podrán ser divididas en varias líneas introduciendo el carácter de continuación de líneas (_) al final de cada línea que forme la sentencia, a excepción de la última. (No obstante, no podrá utilizar un carácter de continuación de línea para partir una cadena que se encuentre entre comillas).

Cuando haya finalizado, su pantalla presentará un aspecto similar al siguiente:

El programa Acceso completo está disponible en la carpeta \Vb5Sbs\Less05 del disco fijo



Botón Iniciar

6. Pulse el botón Iniciar de la barra de herramientas.

   El programa se ejecutará en el entorno de programación. En su pantalla aparecerá un formulario en blanco con un botón. Acceso en la esquina superior izquierda.

7. Pulse el botón Acceso.

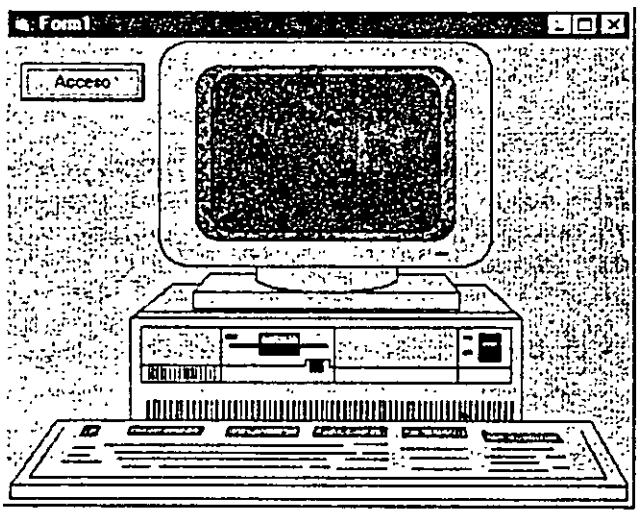
   La función InputBox del procedimiento de suceso Command1_Click mostrará un cuadro de diálogo que le pedirá que introduzca su nombre.

8. Escriba **Laura** y pulse INTRO.

   La estructura de decisión If...Then comparará el nombre que ha escri con el texto «Laura» contenido en la primera expresión condicional. Si ha escrito *Laura*, la expresión dará como resultado Verdadero y la sentencia If...Then mostrará un mensaje de bienvenida utilizando la función MsgBox.

9. Pulse Aceptar en el cuadro de mensaje:

   El cuadro del mensaje se cerrará y en el formulario se cargará un metaarchivo de Windows con la imagen de una PC, tal como se muestra en la siguiente ilustración:



En este programa, el metaarchivo de Windows se carga directamente en el formulario utilizando la propiedad Picture. (Los formularios cuentan con la propiedad Picture, al igual que los objetos Imagen y que los objetos cuadro de imagen). Cuando se carga un gráfico en un formulario éste aparecerá siempre en el segundo plano del formulario. Por ello, cualquier control que exista previamente en el formulario aparecerá por encima del gráfico.

10. Pulse el botón Acceso, escriba **Marcos** y pulse Aceptar.

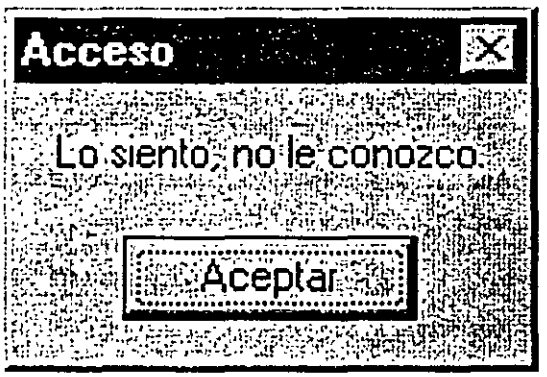
    Esta vez, la estructura de decisión seleccionará la cláusula ElseIf y admitirá a Marcos en el programa. En la pantalla volverá a aparecer un mensaje de bienvenida mostrado por la función MsgBox.

11. Pulse Aceptar para mostrar el dibujo del Rolodex.

    En el formulario se cargará el correspondiente metaarchivo de Windows.

12. Pulse el botón de Acceso, escriba **Federico** y pulse Aceptar.

    Se ejecutará la cláusula Else de la estructura de decisión y en el objeto MsgBox aparecerá el siguiente mensaje.



13. Pulse Aceptar para cerrar el cuadro de mensajes.

    Tanto el cuadro de mensajes como el programa se cerrarán. Se ha prohibido la entrada al programa de un usuario no autorizado.

14. Guarde el formulario como **MiAcceso.frm** y el proyecto como **MiAcceso.vbp**.

Empleo de operadores lógicos en las expresiones condicionales

Los operadores lógicos permiten añadir comprobaciones a las expresiones.

Visual Basic le permitirá comprobar más de una expresión condicional en sus cláusulas If...Then y ElseIf en el caso de que quiera incluir más de un criterio de selección en sus estructuras de decisión. Las condiciones extra se enlazarán mediante el uso de uno o más de los siguientes operadores lógicos:

Operador lógico	Significado
And	Si ambas expresiones condicionales son Verdaderas, el resultado es Verdadero.
Or	Si alguna de las dos expresiones es Verdadera, el resultado es Verdadero.
Not	Si la expresión condicional es Falsa, el resultado es Verdadero. Si la expresión condicional es Verdadera, el resultado es Falso.
Xor	Si una, y sólo una, de las expresiones condicionales es Verdadera, el resultado es Verdadero. Si ambas son Verdaderas o Falsas, el resultado es Falso.

Nota: Cuando el programa evalúa expresiones complejas que mezclen diferentes tipos de operadores, primero evaluará los operadores matemáticos, después los operadores de comparación y, por último, los operadores lógicos.

La tabla que se muestra a continuación lista algunos ejemplos de operadores lógicos en funcionamiento. En las expresiones se ha supuesto que la variable Vehículo contiene el valor «Moto» y que la variable Precio contiene el valor 200000.

Expresión lógica	Resultado
Vehículo = «Moto» And Precio < 300000	Verdadero (ambas expresiones son Verdaderas)
Vehículo = «Coche» Or Precio < 500000	Verdadero (una condición es Verdadera)
Not Precio < 100000	Verdadero (la condición es Falsa)
Vehículo = «Moto» Xor Precio < 300000	Falso (ambas condiciones son Verdaderas)

En el siguiente ejercicio le mostraré cómo modificar el programa MiAcceso para solicitar al usuario que introduzca una contraseña durante el proceso de validación. La contraseña introducida por el usuario se captará mediante un cuadro de entrada. Deberá modificar las cláusulas If...Then y ElseIf de la estructura de decisión para añadir el operador And y poder verificar así la contraseña introducida.

Empleo de una contraseña mediante el uso del operador And

1. Abra el procedimiento de suceso Command1_Click en la ventana Código.
2. Inserte la siguiente sentencia entre las instrucciones InputBox e If...Then del procedimiento (entre las líneas primera y segunda):

```
Paso = InputBox("Introduzca su contraseña.")
```

3. Modifique la sentencia If...Then tal y como se muestra a continuación:

```
If NombreUsuario = "Laura" And Contr = "May17" Then
```

La sentencia incluye ahora un operador lógico And que verificará el nombre del usuario y su contraseña, antes de admitir a «Laura» en el programa.

El programa Contraseña completo se encuentra en el disco fijo en la carpeta \Vb5Sbs\Less05

4. Modifique la sentencia ElseIf tal y como se muestra a continuación:

```
ElseIf NombreUsuario = "Marcos" And Contr = "trek" Then
```

El operador lógico And añadirá una comprobación para verificar la contraseña de la cuenta de Marcos.

5. Guarde el formulario como **MiContraseña.frm** y el proyecto como **MiContraseña.vbp**

Botón Iniciar

6. Pulse el botón Iniciar de la barra de herramientas.
   El programa se ejecutará en el entorno de programación.
7. Pulse el botón Acceso, escriba **Laura** y pulse Aceptar.
   El programa le pedirá una contraseña.
8. Escriba **May17** y pulse Aceptar.
   La expresión condicional And dará como resultado Verdadero y Laura podrá entrar en el programa.
9. Pulse Aceptar para cerrar el cuadro de mensajes.

Botón Terminar

10. Pulse el botón Terminar de la barra de herramientas para salir del programa.
    El programa se detendrá y aparecerá el entorno de programación.

Nota: Si está escribiendo una versión completa del programa MiContraseña deberá plantearse la posibilidad de utilizar un objeto cuadro de texto para recibir la entrada de la contraseña en el programa. Los objetos cuadro de texto incluyen la propiedad PasswordChar, que permiten mostrar un carácter marcador, como un asterisco (*), cuando el usuario escribe, y la propiedad MaxLength, que le permitirá limitar el número máximo de caracteres que se pueden introducir.

ESTRUCTURAS DE DECISIÓN SELECT CASE

Las estructuras de decisión Select Case basan las decisiones de bifurcación en una variable clave

Visual Basic también le permitirá controlar la ejecución de sentencias en los programas mediante el uso de estructuras de decision Select Case. En este libro ya hemos utilizado las estructuras Select Case cuando escribimos procedimientos de suceso para procesar cuadros de lista, cuadros combo y opciones de elementos de menú. Una estructura Select Case es similar a una estructura If...Then...ElseIf, pero resulta más eficiente cuando la bifurcación depende de una variable clave o *caso*

de prueba. También podrá utilizar las estructuras Select Case para hacer que el código de programa sea más fácilmente legible y mejorar su eficiencia.

La sintaxis de una estructura Select Case es la siguiente:

```
Select Case variable
Case valor1
    se ejecutan las sentencias de programa si valor1 coincide con
      variable
Case valor2
    se ejecutan las sentencias de programa si valor2 coincide con
      variable
Case valor3
    se ejecutan las sentencias de programa si valor3 coincide con
      variable
.
.
.
End Select
```

Todas las estructuras Select Case comienzan con las palabras clave Select Case y finalizan con las palabras clave End Select. Deberá reemplazar *variable* por la variable, propiedad o expresión que vaya a ser valor clave o caso de prueba en la estructura. Asimismo, deberá reemplazar *valor1, valor2* y *valor3* por números, cadenas o cualquier otro valor relacionado con la prueba que se está llevando a cabo. Si alguno de los valores coincide con la variable, se ejecutarán las sentencias que aparezcan debajo de la cláusula Case correspondiente y, finalmente, Visual Basic continuará ejecutando el código de programa que aparezca a continuación de la sentencia End Select. En una estructura Select Case se puede incluir cualquier número de cláusulas Case y más de un valor en cada cláusula Case. Si desea mostrar varios valores detrás de un caso, deberá separarlos por comas.

El ejemplo siguiente muestra un posible uso de la estructura Select Case para imprimir un mensaje apropiado acerca de la edad de una persona en un programa. Si la variable Edad coincide con alguno de los valores utilizados en las sentencias Case, aparecerá un mensaje apropiado utilizando una etiqueta.

```
Select Case Edad
Case 16
    Label1.Caption = "Termina ya la ESO, chaval."
Case 18
    Label1.Caption = "Ya tienes edad para votar."
Case 21
    Label1.Caption = "Ya podrás tomar vino en tus comidas."
Case 65
    Label1.Caption = "Ya es hora de disfrutar de tu jubilación."
End Select
```

Las estructuras del tipo Select Case resultan más sencillas de interpretar que en el caso de utilizar una estructura If...Then equivalente

Las estructuras Select Case también permiten el empleo de cláusulas Case Else, que podrá utilizar para ejecutar una acción en el caso de que no se den ninguno de los casos contemplados por las cláusulas Case. A continuación se muestra su funcionamiento en el ejemplo Edad:

```
Select Case Edad
Case 16
    Label1.Caption = "Termina ya la ESO, chaval."
Case 18
    Label1.Caption = "Ya tienes edad para votar."
Case 21
    Label1.Caption = "Ya podrás tomar vino en tus comidas."
Case 65
    Label1.Caption = "Ya es hora de disfrutar de tu jubilación."
Case Else
    Label1.Caption = "Bonita edad, disfrútala."
End Select
```

Empleo de los operadores de comparación dentro de una estructura Select Case

Las estructuras Select Case permiten el empleo de los mismos operadores de comparación que los utilizados en las estructuras If...Then.

En Visual Basic podrá utilizar operadores de comparación para abarcar un rango de valores en cada prueba de una estructura Select Case. Los operadores de comparación de Visual Basic que se pueden utilizar son =,< >, >, <, >= y <=. Para poder utilizar los operadores de comparación deberá incluir la palabra clave Is o la palabra clave To en la expresión para identificar la comparación que se está llevando a cabo. La palabra clave Is indica al compilador que compare la variable de prueba con la expresión que aparece después de la palabra clave Is. La palabra clave To identifica un rango de valores. La siguiente estructura utiliza Is, To y varios operadores de comparación para comprobar la variable Edad y para mostrar uno de entre cinco mensajes:

```
Select Case Edad
Case Is < 13
    Label1. Caption = "Todavía eres un chaval."
Case 13 To 19
    Label1.Caption = "Eres un cándido adolescente."
Case 21
    Label1.Caption = "Ya podrás tomar vino en las comidas."
Case Is > 100
    Label1.Caption = "¡Enhorabuena, abuelo!"
Case Else
    Label1.Caption = "Tienes una bonita edad."
End Select
```

Si el valor de la variable Edad es menor que 13, aparecerá el mensaje «Todavía eres un chaval». Para edades entre 13 y 19, el mensaje será «Eres un cándido adolescente»; etc.

Una estructura de decisión Select Case, normalmente, es más sencilla de leer que la estructura If...Then y resulta más eficiente cuando se realizan tres o más decisiones de bifurcación basándose en una variable o propiedad. No obstante, cuando se lleven a cabo dos o menos comparaciones, o cuando trabaje con varias variables distintas, probablemente preferirá utilizar una estructura de decisión If...Then.

El siguiente ejercicio le mostrará cómo utilizar una estructura Select Case para procesar la entrada obtenida a través de un cuadro de lista. En él se utilizarán las propiedades List1.Text y List1.ListIndex para recoger la entrada y, finalmente, se utilizará una estructura Select Case para mostrar un mensaje en uno de los posibles cuatro idiomas.

Empleo de una estructura Select Case para procesar un cuadro de lista

1. En el menú Archivo, seleccione la opción Nuevo proyecto y cree una nueva aplicación estándar.

   Aparecerá un formulario en blanco en el entorno de programación.

Control Label

2. Pulse el control Label contenido en el cuadro de herramientas y después cree un cuadro grande en la parte superior central del formulario para mostrar el título del programa.

Control ListBox

3. Pulse el control ListBox del cuadro de herramientas y cree un cuadro de lista debajo del cuadro de título introducido anteriormente.
4. Cree una pequeña etiqueta encima del objeto cuadro de lista y, después, cree dos pequeñas etiquetas debajo del cuadro de lista para mostrar la salida del programa.

Control CommandButton

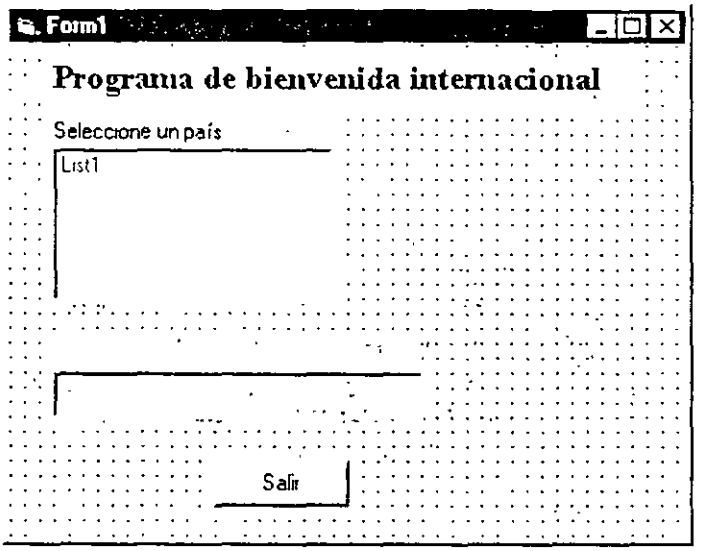
5. Pulse el control CommandButton contenido en el cuadro de herramientas y cree un pequeño botón de orden en la parte inferior central del formulario.

Botón Ventana Propiedades

6. Pulse el botón Ventana Propiedades de la barra de herramientas y asigne las siguientes propiedades a los objetos contenidos en el formulario:

Objeto	Propiedad	Valor
Label1	Caption	«Programa de bienvenida internacional»
	Font	Times New Roman, Negrita,14
Label2	Caption	«Seleccione un país»
Label3	Caption	(Vacío)
Label4	Caption	(Vacío)
	BordeStyle	1-Fixed Single
	ForeColor	Rojo oscuro(&H00000080&)
Command1	Caption	«Salir»

Cuando haya terminado de establecer las propiedades, su formulario presentará un aspecto similar al siguiente:



Introduzca ahora el código de programa para inicializar el cuadro de lista.

7. Pulse dos veces en el formulario.
   En la ventana código aparecerá el procedimiento de suceso Form_Load.

Para introducir valores en un cuadro de lista deberá emplear el método AddItem

8. Escriba el siguiente código de programa para inicializar el cuadro de lista:

```
List1.AddItem "Inglaterra"
List1.AddItem "Alemania"
List1.AddItem "España"
List1.AddItem "Italia"
```

Estas líneas utilizan el método AddItem del cuadro de lista para añadir entradas al cuadro de lista del formulario.

9. Abra el cuadro de lista desplegable Objeto y seleccione el objeto List1.
   En la ventana Código aparecerá el procedimiento de suceso List1_Click.

10. Escriba las siguientes líneas para procesar la selección del cuadro de lista realizado por el usuario:

La propiedad ListIndex contiene el número del elemento seleccionado de la lista.

```
Label3.Caption = List1.Text
Select Case List1.ListIndex
Case 0
    Label4.Caption = "Hello, programmer"
Case 1
    Label4.Caption = "Hallo, Programmierer"
Case 2
    Label4.Caption = "Hola, programador"
Case 3
    Label4.Caption = "Ciao, programmatori"
End Select
```

La primera línea copia el nombre del elemento del cuadro de lista seleccionado en el rótulo de la tercera etiqueta del formulario. La propiedad más importante utilizada en la sentencia es List1.Text, que contiene el texto exacto del elemento seleccionado en el cuadro de lista. Las sentencias restantes forman parte de la estructura de decisión Select Case. La estructura utiliza la propiedad List1.ListIndex como variable de prueba y la compara con diferentes valores. La propiedad ListIndex también contiene el número del elemento seleccionado en el cuadro de lista; el primer elemento de la lista tiene asociado el número cero, el segundo el uno, el siguiente el dos, etc. Mediante el uso de ListIndex, la estructura Select Case puede identificar rápidamente la opción seleccionada por el usuario y mostrar el mensaje correcto en el formulario.

11. Abra el cuadro de lista desplegable Objeto y seleccione el objeto Command1 en el cuadro de lista.

    En la ventana Código aparecerá el procedimiento de suceso Command1_Click.

12. Escriba **End** en el procedimiento de suceso y después cierre la ventana Código.

13. Guarde el formulario en disco con el nombre **MiCaso.frm** y después guarde el proyecto en disco con el nombre **MiCaso.vbp**

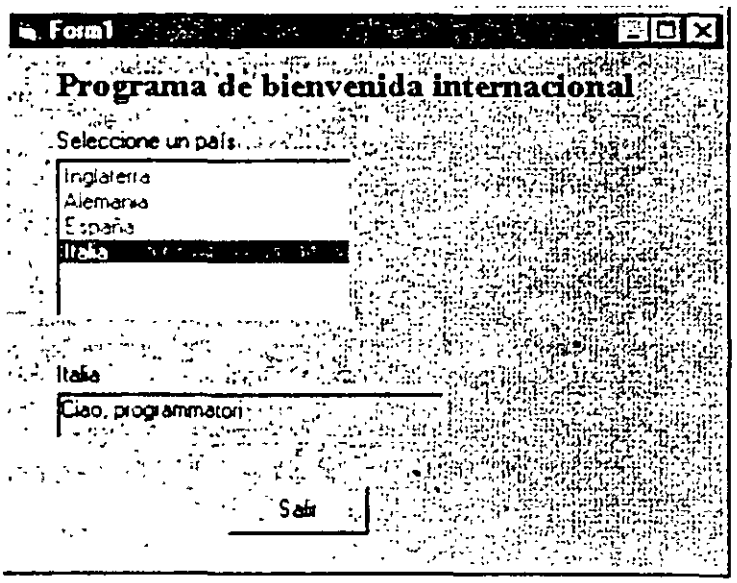
Botón Iniciar

14. Pulse el botón Iniciar para ejecutar el programa MiCaso.

15. Pulse cada uno de los nombres de país del cuadro de lista «Seleccione un país».

    El programa mostrará un mensaje de salutación para cada uno de los países listados. En la siguiente figura se muestra el mensaje correspondiente a Italia.

El programas Caso completo se encuentra almacenado en el disco fijo en la carpeta \Vb5Sbs\Less05



16. Pulse el botón Salir para detener el programa.

    El programa se detendrá y volverá a aparecer el entorno de programación. Por el momento, hemos terminado con las estructuras Select Case.

BÚSQUEDA Y CORRECCIÓN DE ERRORES

El proceso de localización y corrección de errores en programas se denomina depuración

Los errores que haya encontrado hasta ahora en sus programas habrán sido siempre errores de mecanografía o errores de sintaxis. Ahora bien, ¿qué ocurre si descubre un problema más complicado en su programa, uno que no pueda corregir con sencillez simplemente revisando las propiedades de los objetos y las sentencias del programa? El entorno de programación de Visual Basic contiene bastantes herramientas que podrá utilizar para detectar y corregir errores, o *bugs*, en sus programas. Estas herramientas no le impedirán que cometa errores pero, a menudo, le facilitarán la resolución del problema cuando aparezca.

Examine la siguiente estructura de decisión If...Then que evalúa dos expresiones condicionales y después muestra un mensaje basándose en el resultado:

```
If Edad > 13 AND Edad < 20 Then
    Text2.Text = "Eres un adolescente."
Else
    Text2.Text = "No eres un adolescente."
End If
```

¿Ha podido detectar el problema en esta estructura de decisión? En general, se considera como adolescente a cualquier persona cuya edad esté comprendida entre 13 y 19 años, ambos inclusive, por lo que la estructura falla, ya que no permitirá identificar como adolescentes a las personas cuya edad sea exactamente 13 años (para esta edad, la estructura muestra erróneamente el mensaje «No eres un adolescente»). Este tipo de error no es un error de sintaxis (las sentencias siguen las reglas de Visual Basic); es un error de planteamiento o *error lógico*. La estructura de decisión correcta deberá contener la siguiente sentencia If...Then:

```
If Edad >= 13 AND Edad < 20 Then
```

Lo crea o no, este tipo de errores es el problema más frecuente en un programa de Visual Basic. El código funciona bien la mayoría de las veces, aunque no siempre. Por tanto, es el más difícil de detectar y corregir.

Tres tipos de errores

En un programa de Visual Basic pueden ocurrir tres tipos de errores: errores de sintaxis, errores en tiempo de ejecución y errores lógicos:

- Un *error de sintaxis* (o *error de compilación*) es un error de programación que viola las reglas de Visual Basic (como una propiedad mal

(continúa)

Tres tipos de errores *(continuación)*

escrita o una palabra clave mal escrita). Visual Basic es capaz de detectar muchos tipos de errores a medida que vaya escribiendo las sentencias de programa y no permitirá la ejecución del programa hasta que se corrijan todos los errores de sintaxis.

- Un *error en tiempo de ejecución* es un error que provoca la detención no esperada de un programa durante su ejecución. Los «errores en tiempo de ejecución» hacen referencia a cualquier error, normalmente un suceso externo o un error de sintaxis no descubierto inicialmente por el compilador, que fuerzan la interrupción de la ejecución del programa. Dos ejemplos de condiciones que pueden provocar un error en tiempo de ejecución son un nombre de archivo mal escrito en una función LoadPicture o una unidad de disco abierta.
- Un *error lógico* es un error humano —un error de programación que obliga al código de programa a producir resultados erróneos—. La mayor parte del esfuerzo de depuración suele estar dedicado a la detección de este tipo de errores introducidos inadvertidamente por el programador.

Utilice la ayuda interactiva de Visual Basic cuando detecte mensajes de error producidos por errores de sintaxis o errores en tiempo de ejecución. Si aparece un cuadro de diálogo relacionado con un error en tiempo de ejecución, pulse el botón Ayuda.

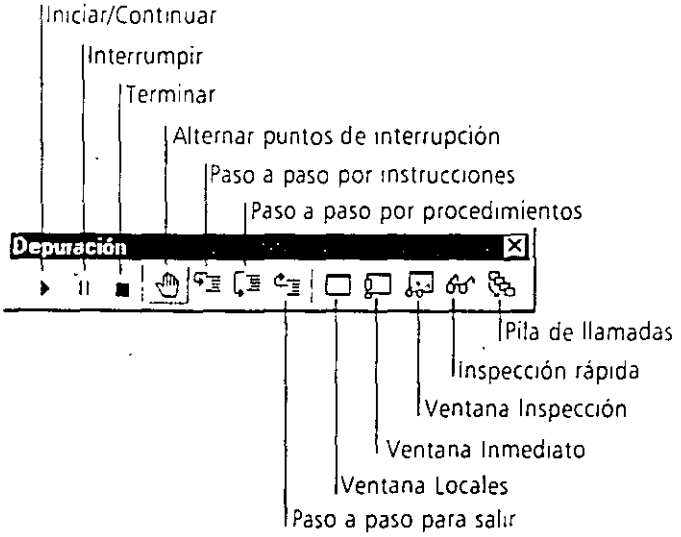
Empleo del modo paso a paso

El modo paso a paso le permite ver cómo se ejecuta el programa.

Un método para identificar un error lógico consiste en ejecutar el código del programa línea a línea y examinar el valor de una o más variables o propiedades según éstas cambien. Para ello podrá entrar en *modo ruptura* mientras el programa se esté ejecutando y ver el código en la ventana Código. El modo ruptura le proporcionará una vista más próxima de su programa durante la ejecución paso a paso del mismo. Es como colocar otro sillón justo detrás de los asientos del piloto y copiloto de una aeronave y ver cómo realizan las maniobras. Pero, en este caso, podrá tocar los controles.

Cuando esté verificando sus aplicaciones podrá abrir una barra de herramientas nueva, la denominada Depuración, que contiene botones dedicados exclusivamente a localizar y depurar errores. Quizás también sea de su interés abrir la ventana denominada Inspección, donde se mostrarán los valores de las variables críticas que le interesen. También podrá utilizar la ventana Inmediato para introducir instrucciones de programa y ver su efecto inmediato.

La siguiente figura muestra el contenido de la barra de Depuración que podrá ver desplegando el submenú Barras de herramientas contenido en el menú Ver y ejecutando la opción Depuración.



En el siguiente ejercicio utilizará el modo interrupción para localizar y corregir los errores lógicos descubiertos anteriormente en la estructura If...Then (el error forma parte de un programa real). Para aislar el problema utilizará el botón Paso a paso por instrucciones contenido en la barra de herramientas Depuración para ejecutar una a una las instrucciones contenidas en el programa. También utilizará el botón Inspección rápida para mirar los cambios que se produzcan en el valor de la variable Edad. Preste mucha atención a esta estrategia de depuración. Podrá utilizarla para corregir muchos tipos de errores en sus programas.

Depuración del programa Errores

Botón Abrir proyecto

1. Pulse el botón Abrir proyecto contenido en la barra de herramientas.
2. Abra el proyecto Errores de la carpeta \VB5Sbs\Less05.
3. Si el formulario no está visible, resalte el formulario Errores en la ventana Proyecto y pulse el botón Ver Objeto.

   Aparecerá el formulario del programa Errores. Este programa solicita al usuario que introduzca su edad. Cuando el usuario pulse el botón Prueba, el programa comprobará si el usuario es o no un adolescente. El programa sigue teniendo el problema de los 13 años que hemos identificado antes. A continuación le mostraré cómo abrir la barra de herramientas Depuración y entre en el modo Interrupción para localizar el problema.
4. En el menú Ver, despliegue el submenú Barras de herramientas y seleccione la opción Depuración, si todavía no se encuentra seleccionada.

En pantalla aparecerá la barra de herramientas Depuración (puede aparecer fijada a la derecha de la barra de herramientas Estándar.

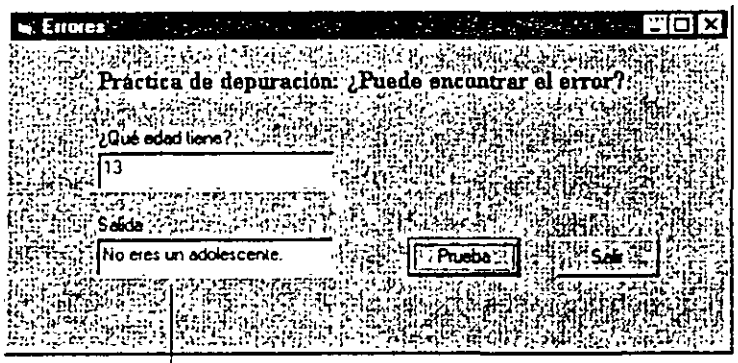
5. Sitúe la barra de herramientas Depuración justo debajo del formulario Errores para que le sea más fácil de manejar.


Botón Iniciar

6. Pulse el botón Iniciar de la barra de herramientas Depuración.
7. El programa se ejecutará. Elimine el 0 del cuadro de texto Edad, escriba **14** y después pulse el botón Prueba.

   El programa mostrará el mensaje «Eres un adolescente». En este caso, el programa muestra el resultado correcto.
8. Escriba **13** en el cuadro de texto ¿Qué Edad tienes? y pulse el botón Prueba.

   El programa mostrará el mensaje «No eres un adolescente», tal como se muestra en la siguiente figura:



Este resultado es un error

Esta respuesta es incorrecta y deberá analizar el código del programa para localizar el problema. En lugar de abandonar el programa y analizar el código de programa por su cuenta, puede pedir ayuda a Visual Basic.


Botón Interrumpir

9. Pulse el botón Interrumpir contenido en la barra de herramientas Depuración (el botón Interrumpir está justo a la derecha del botón Iniciar).

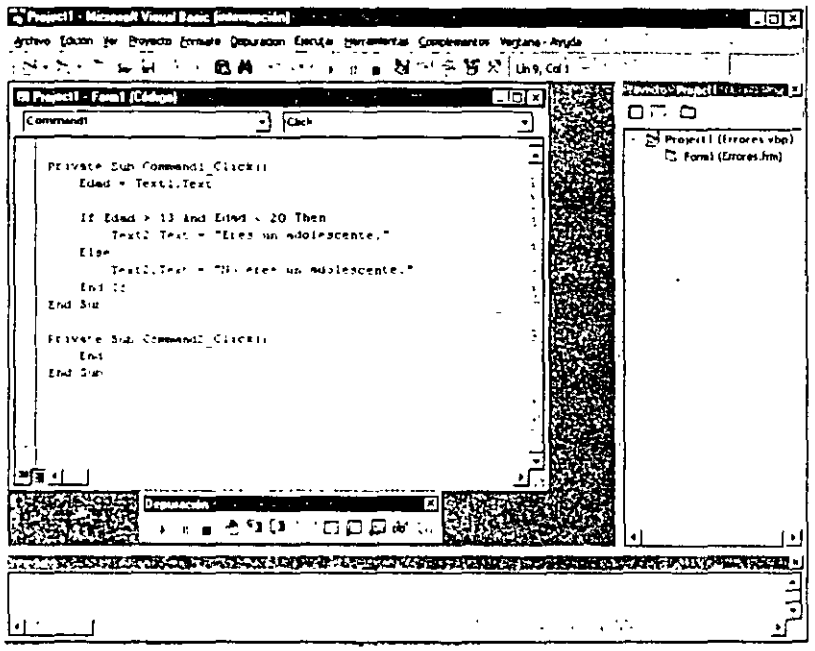
   El programa hará una pausa y Visual Basic mostrará la ventana Código que muestra el código que está ejecutando Visual Basic. Su pantalla será similar a la figura de la página siguiente.


Botón Paso a paso por instrucciones

10. Pulse el botón Paso a paso por instrucciones contenido en la barra de herramientas Depuración para ejecutar la siguiente sentencia del programa.

    Visual Basic devolverá el control al formulario del programa y esperará la entrada.
11. Pulse el formulario Errores en la barra de tareas de Windows, verifique que el 13 sigue estando en el cuadro de texto y pulse el botón Prueba.

    Como Visual Basic está en modo ruptura, ocurre algo inusual. Visual Basic abrirá la ventana Código y mostrará el procedimiento de su



Command1_Click (el código de programa que va a ser ejecutado por el compilador). La primera instrucción contenida en este procedimiento se muestra resaltada en amarillo. De esta forma, se le brinda la oportunidad de ver cómo funciona la lógica del programa.

12. Vuelva a pulsar el botón Paso a paso por instrucciones para ejecutar la primera sentencia del procedimiento.

    La sentencia Sub se ejecutará y la sentencia que contiene la variable Edad aparecerá resaltada. Edad es la variable de prueba crítica en este programa, por lo que deberá introducirla en la ventana de Inspección para ver cómo se va modificando su valor durante la ejecución del programa.

Truco: Cuando su programa se encuentre en modo Interrupción, podrá verificar el valor de una variable contenida en la ventana Código manteniendo el puntero del ratón sobre ella.

Botón Inspección rápida

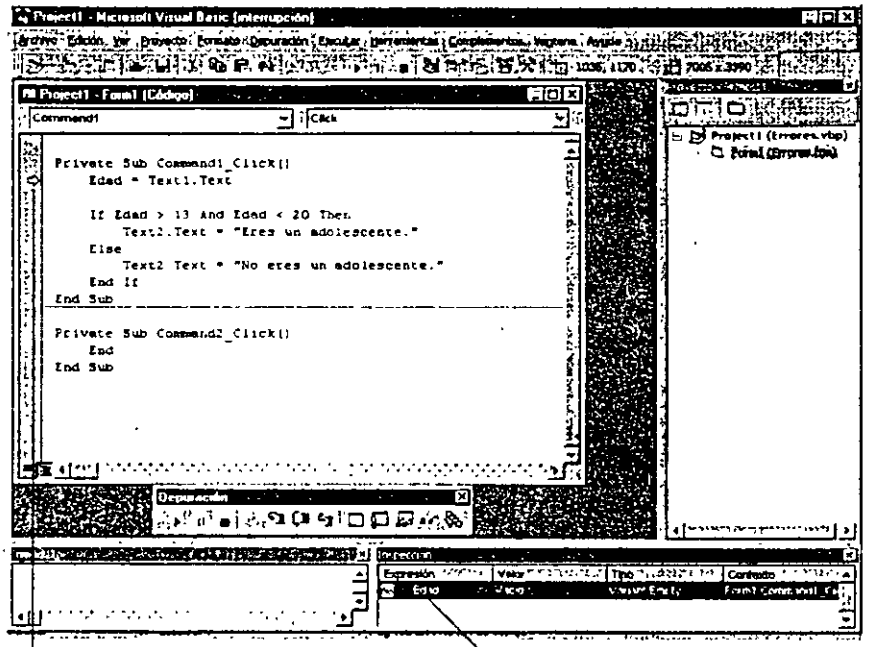
La ventana Inspección muestra variables añadidas mediante el empleo de la orden Inspección Instantanea.

13. Seleccione la variable Edad utilizando el ratón y después pulse el botón Inspección rápida contenido en la barra de herramientas Depuración.

    En la ventana aparecerá un cuadro de diálogo mostrando el contexto, el nombre y el valor de la variable Edad del programa. También podrá ver este cuadro de diálogo si selecciona la opción Inspección rápida del menú Depuración.

14. Pulse el botón Agregar contenido en el cuadro de diálogo Inspección rápida. La ventana de Inspección aparecerá fijada en la parte inferior de la

pantalla (tal vez necesite aumentarla de tamaño para ver completamente su contenido).



En este momento la variable Edad no tiene ningún valor, ya que aún no ha sido utilizada en el procedimiento de suceso (como la variable Edad no ha sido declarada globalmente en el programa, se utilizará como variable local en el procedimiento y se reinicializa cada vez que se llame a dicho procedimiento).

Nota: Para eliminar una variable de inspección de la ventana Inspección, pulse el nombre de la variable en la ventana de Inspección y presione la tecla SUPR.

---

15. Pulse el botón Paso a paso por instrucciones para ejecutar la siguiente sentencia.

    Visual Basic pasará el número 13 desde el cuadro de texto a la variable Edad y la variable Edad se actualizará en la ventana Inspección. Visual Basic resaltará ahora la primera sentencia de la estructura If...Then, la instrucción más importante del programa y la que contiene el error que andamos buscando. Como con 13 años cualquier niño es ya un adolescente, Visual Basic debería ejecutar la cláusula Then después de evaluar esta instrucción.

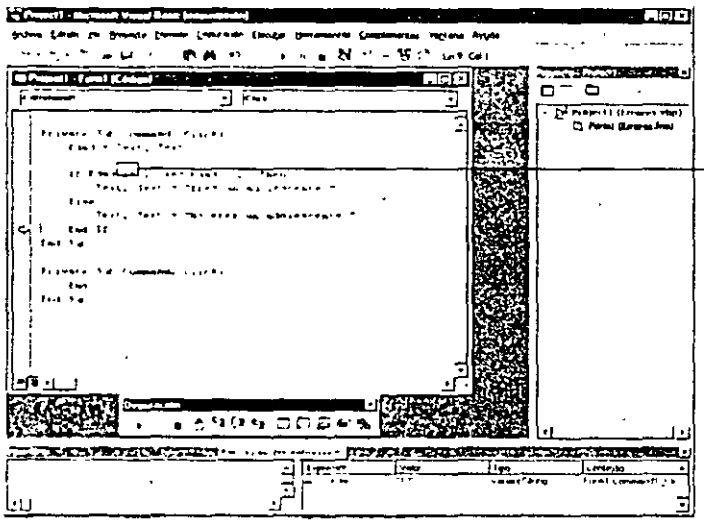
Cuando una estructura de decisión bifurca incorrectamente, deberá buscar el error en la expresión condicional y corregirlo, si es posible.

16. Pulse de nuevo el botón Paso a paso por instrucciones.

    Visual Basic resaltará ahora la cláusula Else en la estructura If...Th
    Como puede ver, la prueba falla con este valor de la variable Edad. Por lo

tanto, deberá localizar y corregir el problema ahora, si es posible. Visual Basic le ha ayudado a detectar el problema, será su obligación reconocerlo y corregirlo. Esta vez ya conoce la solución. En la primera comparación deberá utilizar el operador > =.

17. Pulse detrás del operador > de la sentencia If...Then y escriba =. Su pantalla presentará un aspecto similar al siguiente:



Visual Basic le permite corregir errores mientras se encuentre en modo ruptura, pero si los cambios se realizan en sentencias que ya se han ejecutado, las correcciones no tendrán efecto hasta la siguiente vez que se ejecuten dichas sentencias. Para comprobar que la corrección que acaba de efectuar funciona adecuadamente deberá pulsar de nuevo el botón Prueba.

18. Pulse tres veces el botón Paso a paso por instrucciones.
    Visual Basic finalizará la ejecución de la estructura de decisión.

19. Pulse el botón Iniciar (ahora llamado Continuar) de la barra de herramientas Depuración para realizar la ejecución completa del programa.
    El formulario Errores reaparecerá.

20. Pulse el botón Prueba para comprobar el error corregido, verifique el mensaje «Eres un adolescente» en el cuadro Salida y después pulse el botón Salir para detener el programa.

21. Pulse el botón Cerrar contenido en la barra de herramientas Depuración y la ventana Inspección desaparecerá de su pantalla.

¡Enhorabuena!, ya ha utilizado satisfactoriamente el modo Interrupción (ruptura) para localizar y corregir un error lógico en un programa. A medida que siga

trabajando con Visual Basic, no dude en utilizar el modo ruptura y la barra de herramientas Depuración para analizar su código.

UN PASO MÁS ALLÁ: USO DE UNA INSTRUCCIÓN STOP PARA ENTRAR EN EL MODO RUPTURA

Podrá entrar en el modo ruptura utilizando una instrucción Stop.

Si desea interrumpir la ejecución del programa en un lugar determinado y comenzar allí la depuración, deberá colocar la sentencia Stop en dicho lugar del código. De esta forma, se detendrá la ejecución del programa y se mostrará la ventana Código. Por ejemplo, como alternativa a la pulsación del botón Interrumpir, podría haber activado el modo ruptura en el ejercicio anterior mediante la inserción de una sentencia Stop al principio del procedimiento de suceso Command1_Click, tal y como se muestra a continuación:

```
Private Sub Command1_Click()
    Stop     'entra en modo ruptura
    Edad = Text1.Text

    If Edad > 13 AND Edad < 20 Then
        Text2.Text = "Eres un adolescente."
    Else
        Text2.Text = "No eres un adolescente."
    End If
End Sub
```

Cuando se ejecute un programa que incluya una sentencia Stop, Visual Basic entrará en el modo ruptura tan pronto como se localice la sentencia Stop. Mientras esté en modo ruptura, podrá utilizar la ventana Código y la barra de herramientas Depuración en la misma forma que en el caso de que hubiera entrado el modo ruptura manualmente. Cuando finalice la depuración, elimine la instrucción Stop.

Si desea continuar con el siguiente capítulo

➤ No salga de Visual Basic y pase al Capítulo 6.

Si desea salir de Visual Basic por ahora

➤ En el menú Archivo seleccione Salir.

Si en su pantalla aparece un cuadro de diálogo que le permite almacenar l cambios, seleccione Sí.

RESUMEN DEL CAPÍTULO

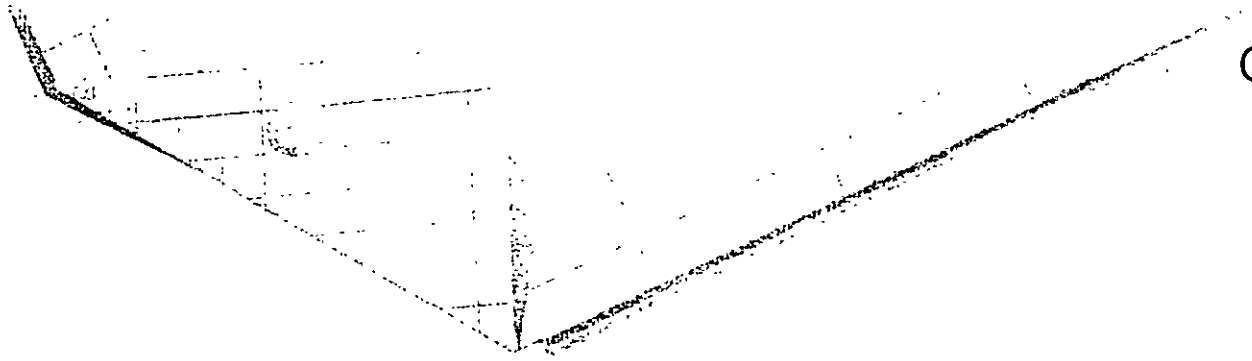
Para	Haga esto	Botón
Esribir una expresión condicional	Utilice un operador de comparación entre dos valores.	
Utilizar una estructura de decisión	Utilice una sentencia If...Then o Select Case y las expresiones y palabras clave necesarias.	
Realizar dos comparaciones en una expresión condicional	Utilice un operador lógico (And, Or, Not o Xor) entre las comparaciones.	
Mostrar la barra de herramientas Depuración	En el menú Ver, despliegue el submenú Herramientas y ejecute el mandato Depuración.	
Entrar en modo ruptura para corrección de errores	Pulse el botón Interrumpir en la barra de herramientas Depuración	
	o	
	Introduzca una sentencia Stop donde desee provocar la interrupción.	
Ejecutar una línea de código en la ventana Código	Pulse el botón Paso a paso por instrucciones	
	o	
	En el menú Depuración, ejecute el mandato Paso a paso por instrucciones.	
Examinar una variable en la ventana Código	Résalte la variable que desee examinar y pulse el botón Inspección rápida contenido en la barra de herramientas Depuración o seleccione la opción Inspección rápida del menú Depuración.	
Eliminar una expresión de inspección	Seleccione la expresión en la ventana Inspección y pulse Eliminar.	

Sobre la ayuda en línea	**En el menú Ayuda de Visual Basic, seleccione el mandato Temas de ayuda de Microsoft Visual Basic, seleccione la etiqueta Índice y a continuación**
Operadores de comparación	Escriba **Operadores de comparación**
Operadores lógicos	Escriba Operadores lógicos
Estructuras de decisión If...Then	Escriba **If**

Sobre la ayuda en línea	En el menú Ayuda de Visual Basic, seleccione el mandato Temas de ayuda de Microsoft Visual Basic, seleccione la etiqueta Índice y a continuación
Estructuras de decisión Select Case	Escriba **Select (Instrucción)**
Depuración	Escriba **Depurar**

AVANCE DEL SIGUIENTE CAPÍTULO

En el siguiente capítulo, «Empleo de bucles y temporizadores», aprenderá a manejar acciones repetitivas en sus programas. Conocerá el funcionamiento de las instrucciones For...Next, Do y While y cómo utilizar el control Timer.



Capítulo

6

Empleo de bucles y del control Timer

En el Capítulo 5 se analizaron las estructuras de decisión If...Then y Select Case que podrán utilizarse para seleccionar qué sentencias se desea ejecutar en un programa. En este capítulo le mostraré cómo ejecutar un bloque de sentencias una y otra vez mediante el uso de un *bucle*. Utilizará un bucle For...Next para ejecutar sentencias un número exacto de veces y empleará un bucle Do para ejecutar un grupo de sentencias hasta que la expresión condicional del bucle se evalúe como Verdadero. También aprenderá a utilizar el método Print para mostrar textos y números en un formulario y a utilizar un objeto temporizador para ejecutar código a ciertos intervalos temporales en sus programas.

ESCRITURA DE BUCLES FOR...NEXT

Un bucle For...Next le permite ejecutar un grupo específico de sentencias de programa una serie concreta de veces dentro de un procedimiento de suceso. Esta forma de proceder puede resultar útil si está realizando cálculos relacionados entre sí, si está trabajando con elementos en la pantalla o si se encuentra procesando varias piezas de información introducidas por el usuario. Un bucle For...Next es, en realidad, una forma abreviada de escribir una larga lista de sentencias de programa. Como cada grupo de sentencias de la lista haría esencialmente lo mismo, Visual Basic le permite definir un grupo de sentencias y hacer que dichas sentencias se ejecuten tantas veces como desee.

La sintaxis de un bucle For...Next es la siguiente:

En un bucle For...Next, las variables inicio y fin determinan las veces que se repetirá el bucle

```
For variable = inicio To fin
      sentencias que se van a repetir
Next variable
```

En este tipo de sentencias, For, To y Next son palabras clave imprescindibles, mientras que el operador = (igual) también es necesario. La palabra *variable* indica una variable numérica que se encargará de llevar la cuenta del número de veces que se ha ejecutado el bucle, mientras que las palabras *inicio* y *fin* son valores numéricos que representan los puntos inicial y final de ejecución del bucle. La línea o líneas que se encuentren entre las sentencias For y Next serán las instrucciones que se repetirán cada vez que se ejecute el bucle.

Por ejemplo, el siguiente bucle For..Next emite cuatro pitidos en una sucesión rápida a través del altavoz de la computadora:

```
For i = 1 To 4
      Beep
Next i
```

Este bucle es el equivalente funcional a escribir cuatro veces la sentencia Beep en el mismo procedimiento. Para un compilador sería lo mismo que escribir:

```
Beep
Beep
Beep
Beep
```

La variable utilizada en el bucle es «i», una letra que, por convenio, se utiliza como primer contador de enteros en un bucle For...Next. Cada vez que se ejecute el bucle, la variable contadora se incrementará una unidad (la primera vez que se ejecute el bucle, la variable contadora tendrá el valor 1, o valor de *inicio*; la última vez, la variable valdrá 4, o valor *final*). Como le mostraré en los ejemplos siguientes, esta variable contadora se puede utilizar con diversos propósitos de cálculo dentro del bucle.

Visualización de una variable contadora utilizando el método Print

El método Print envía la salida al formulario o a la impresora

Una variable contadora es idéntica a cualquier otra variable dentro del procedimiento de suceso. Se puede asignar a propiedades, se puede utilizar en cálculos o se puede mostrar en un programa. Una de las técnicas más sencillas utilizadas para visualizar una variable contadora consiste en utilizar el método Print, una sentencia especial que muestra la salida en el formulario o la imprime en una impresora conectada. El método Print tiene la siguiente sintaxis:

```
Print expresión
```

donde *expresión* es una variable, propiedad, valor de texto o valor numérico del procedimiento. En el siguiente ejercicio utilizará el método Print para mostrar la salida de un bucle For...Next en un formulario.

Truco: Si tiene pensado minimizar un formulario que contenga la salida obtenida mediante un método Print, asigne el valor True a la propiedad AutoRedraw del formulario. De esta forma, Visual Basic podrá recrear automáticamente la salida cuando vuelva a mostrar en pantalla el formulario. A diferencia de los objetos contenidos en un formulario, que se redibujan de forma automática, el texto mostrado con el método Print sólo reaparecerá si define como True la propiedad AutoRedraw.

Cómo visualizar información utilizando un bucle For...Next

1. Ponga en marcha Microsoft Visual Basic.

   Si Visual Basic ya está en ejecución, abra un nuevo proyecto EXE estándar.

2. Aumente el tamaño del formulario con el puntero del ratón para aumentar el espacio disponible para mostrar la salida.

Control CommandButton

3. Utilice el control CommandButton para crear un botón de orden en la parte inferior central del formulario.

4. Abra la ventana Propiedades y asigne a la propiedad Caption del botón de orden la palabra «Bucle».

Botón Propiedades

5. Abra el cuadro de lista desplegable objeto situado en la parte superior de la ventana Propiedades y pulse el nombre de objeto Form1.

   En la ventana Propiedades aparecerán las propiedades del formulario.

6. Cambie la propiedad Font a Times New Roman.

   La propiedad Font controla el tipo de letra que utilizará el programa para representar al texto contenido en el formulario. Podrá utilizar cualquier fuente para este cometido; sin embargo, las fuentes True Type son las que mejor funcionan, ya que pueden visualizarse en distintos tamaños y su aspecto es idéntico tanto en la pantalla como en la impresora.

7. Cambie la propiedad AutoRedraw a True.

   Si se oculta el formulario, la propiedad AutoRedraw volverá a mostrar cualquier texto visualizado previamente por el método Print.

8. Pulse dos veces el botón Bucle del formulario.

   En la ventana Código aparecerá el procedimiento de suceso Command1_Click.

9. Introduzca las siguientes sentencias de programa en dicho procedimiento:

```
For i = 1 To 10
    Print "Línea"; i
Next i
```

Este bucle For...Next utiliza el método Print para mostrar 10 veces en el formulario la palabra *Línea*, seguida del contador del bucle. El punto y coma (;) de la sentencia Print hace que Visual Basic muestre la variable contador junto a la cadena «Línea», sin espacio adicional en medio. Sin embargo, cuando ejecute el programa podrá ver un espacio en blanco entre «Línea» y la variable contadora. Cuando se imprimen valores numéricos, el método Print reserva un espacio en blanco para el signo menos, aunque dicho signo no sea necesario en todas las ocasiones.

El programa BucleFor completo se encuentra en la carpeta \Vb5Sbs\Less06 del disco

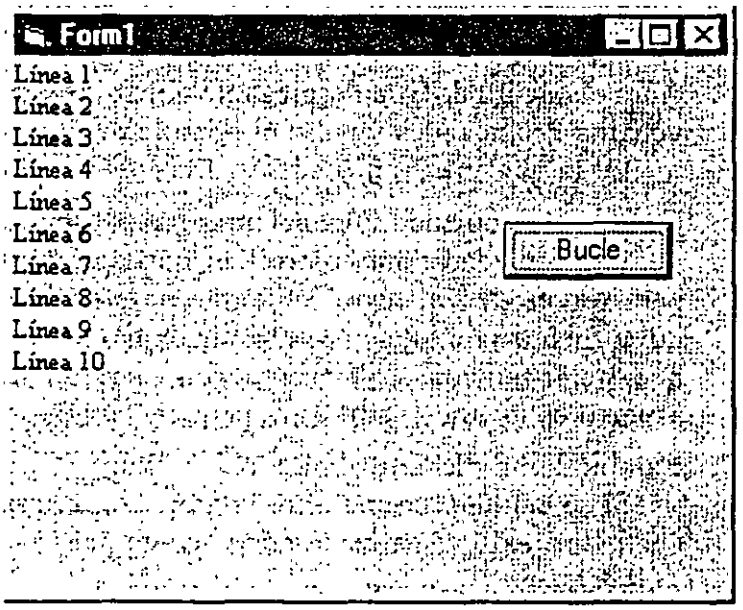
Nota: *El método Print permite utilizar los símbolos punto y coma (;) y coma (,) para separar los elementos contenidos en una lista. El punto y coma coloca los elementos uno al lado del otro, mientras que la coma separa cada par de elementos mediante un tabulador. Podrá utilizar cualquier combinación de comas y puntos y comas para separar los elementos de una lista.*

Ahora ya puede ejecutar el programa.

Botón Iniciar

10. Pulse el botón Iniciar contenido en la barra de herramientas.
11. Pulse el botón Bucle.

    El bucle For...Next imprimirá 10 líneas en el formulario, tal como se muestra a continuación:



12. Vuelva a pulsar el botón Bucle.

    El bucle For...Next imprimirá otras 10 líneas en el formulario (o tantas como quepan). Cada vez que se imprime una línea, el cursor se desplaza una línea hacia abajo, hasta que el cursor llegue al borde inferior del formulario.

Botón Terminar

13. Pulse el botón Terminar de la barra de herramientas para detener el programa.

Botón Guardar proyecto

14. Pulse el botón Guardar proyecto de la barra de herramientas. Guarde el formulario como **MiBucleFor.frm** y guarde el proyecto como **MiBucleFor.vbp**.

    Guarde los archivos en la carpeta \Vb5Sbs\Less06.

Modificación de una propiedad de un bucle For...Next

La propiedad FontSize modifica el tamaño de las fuentes utilizadas en el formulario

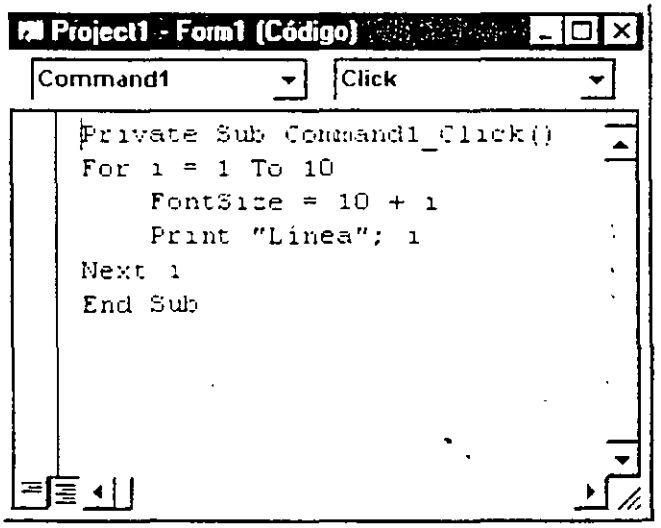
Visual Basic le permitirá modificar propiedades y actualizar variables clave en un bucle. En el siguiente ejercicio modificará el programa MiBucleFor para cambiar la propiedad FontSize utilizando un bucle For...Next. La propiedad FontSize especifica el tamaño del texto mostrado en un formulario; podrá utilizarla como alternativa a la modificación del tamaño con la propiedad Font.

Modificación de la propiedad FontSize

1. Abra el procedimiento de suceso Command1_Click si es que no está abierto. En la ventana Código aparecerá el bucle For...Next.
2. Inserte la siguiente sentencia directamente debajo de la sentencia For:

```
FontSize = 10 + i
```

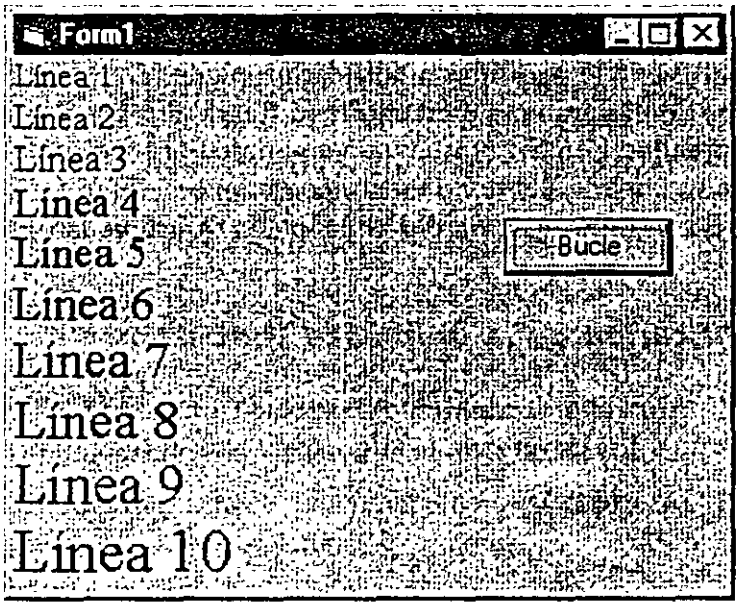
Esta sentencia asigna a la propiedad FontSize del formulario el valor 10 puntos más el valor que tenga en cada pasada el contador del bucle. La primera vez que se atraviese el bucle, el tamaño de fuente será 11 puntos, la siguiente vez será 12, y así hasta la última vuelta del bucle, cuando el tamaño de la fuente será de 20 puntos. Cuando haya terminado, el bucle For...Next será similar al que se muestra en la figura siguiente:



Botón Iniciar

3. Pulse el botón de Iniciar de la barra de herramientas para ejecutar el programa.
4. Pulse el botón Bucle.

   El bucle For...Next mostrará la siguiente salida en el formulario:



Botón Terminar

   Cada vez que se incrementa el contador del bucle, el tamaño del tipo de letra también se incrementa en el formulario.

5. Pulse el botón Terminar para detener el programa.

   El programa se detendrá y volverá a aparecer el entorno de programación.

6. En el menú Archivo, pulse la orden Guardar MiBucleFor.frm como. Guarde el formulario modificado con el nombre **MiAumentoFuente.frm.**

El programa AumentoFuente completo se encuentra en la carpeta \Vb5Sbs\Less06

7. En el menú Archivo, pulse la orden Guardar proyecto como. Guarde el proyecto Como **MiAumentoFuente.vbp**.

Importante: Un bucle For...Next puede ahorrar bastante espacio en un programa. En el ejemplo anterior, un bucle For...Next de cuatro líneas ha procesado el equivalente a 20 sentencias de programa.

Bucles For...Next complejos

La variable contadora de un bucle For...Next puede ser una potente herramienta en sus programas. Con un poquito de imaginación podrá utilizarla para cr distintas secuencias útiles de números en sus Bucles. Para crear un bucle con u...

Con la palabra clave Step podrá crear otras secuencias o sucesiones de números para la variable contadora del bucle For...Next.

sucesión distinta a la de 1, 2, 3, 4, etc., podrá especificar un valor distinto para el *inicio* del bucle y utilizar la palabra clave Step para incrementar el contador en intervalos distintos a la unidad. Por ejemplo, el bucle

```
For i = 5 To 25 Step 5
      Print i
Next i
```

imprimirá en el formulario la siguiente secuencia de números:

```
5
10
15
20
25
```

Podrá utilizar la palabra clave Step con valores decimales.

En un bucle también se pueden especificar valores decimales. Por ejemplo, el bucle For...Next

```
For i = 1 To 2.5 Step 0.5
      Print i
Next i
```

imprimirá en un formulario los siguientes números:

```
1
1.5
2
2.5
```

Además de mostrar el valor asociado al contador podrá utilizar este valor para asignar propiedades, calcular valores o procesar archivos. El siguiente ejercicio muestra cómo podrá utilizar el contador para abrir iconos de Visual Basic que se encuentran almacenados en distintos archivos del disco, con números en sus nombres. El programa muestra también cómo podrá utilizar un bucle For...Next para trabajar con varios objetos de imagen en modo de grupo. Para organizar los objetos de imagen de forma que se puedan procesar de forma eficiente, se introducirán en un contenedor denominado array de control.

Cómo abrir archivos utilizando el bucle For...Next

1. En el menú Archivo, seleccione la opción Nuevo proyecto y pulse Aceptar.
2. Pulse el control Image en el cuadro de herramientas y cree después un pequeño cuadro de imagen junto a la esquina superior izquierda del formulario.

Control Image

3. En el menú Edición seleccione la opción Copiar.

   En el portapapeles de Microsoft Windows se almacenará una copia del cuadro de imagen. Esta copia se utilizará para crear en el formulario tres cuadros de imagen adicionales.

Creará un array de control copiando y pegando objetos

4. En el menú Edición, seleccione la opción Pegar.

   Visual Basic mostrará un mensaje preguntándole si desea crear un array de control en su programa. Un *array de control* es un grupo de objetos idénticos que se mostrarán en la interfaz del programa. Cada uno de los objetos del grupo comparte el mismo nombre del objeto, de forma que se puede seleccionar y definir simultáneamente el grupo entero de objetos. Sin embargo, los objetos de un array de control también pueden utilizarse individualmente, por lo que el programador tiene control completo sobre cada uno de los elementos de la interfaz de usuario.

5. Pulse Sí para crear un array de control.

   Visual Basic crea un array de control de cuadros de imagen y pega el segundo cuadro de imagen en la esquina superior izquierda del formulario. El nuevo objeto estará seleccionado.

6. Arrastre el segundo cuadro de imagen hasta situarlo a la derecha del primer cuadro de imagen.

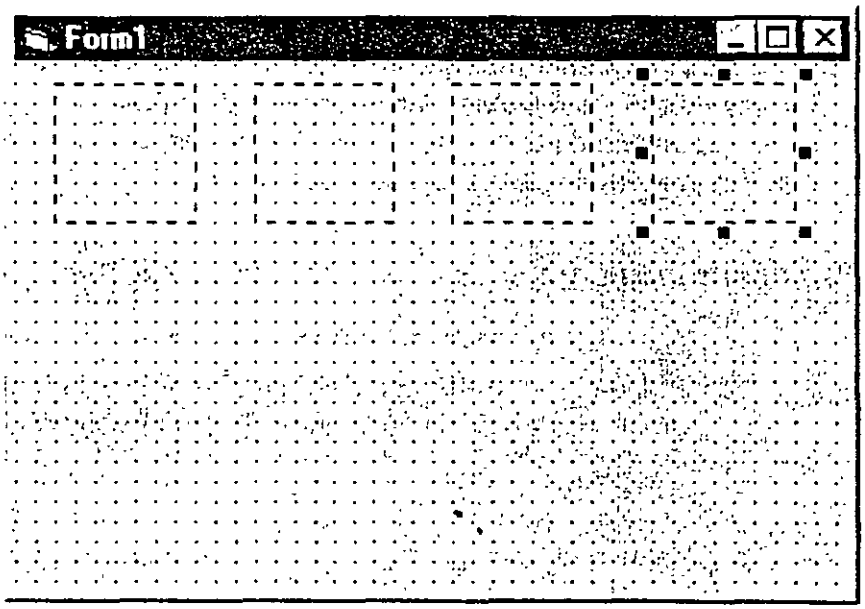
Nota: Después de seleccionar un objeto podrá arrastrarlo hacia cualquier parte del formulario.

---

7. En el menú Edición, vuelva a seleccionar la opción Pegar y después arrastre el tercer cuadro de imagen a la derecha del segundo.

8. Vuelva a ejecutar la opción Pegar y coloque el cuarto cuadro de imagen a la derecha del tercero.

   Cuando haya terminado de pegar estos cuatro objetos, su pantalla ofrecerá un aspecto similar al de la siguiente ilustración:



Control CommandButton

9. Pulse el control CommandButton de la barra de herramientas y cree un botón de orden en la parte inferior del formulario.

   Asigne ahora las propiedades a los objetos contenidos en el formulario. En primer lugar, asignará las propiedades correspondientes a los cuadros de imagen del array de control tratándolos como grupo.

Podrá trabajar con todos los objetos de un array de control seleccionándolos como grupo

10. Seleccione el primer cuadro de imagen, pulse y mantenga pulsada la tecla MAYÚS y después seleccione los cuadros de imagen segundo, tercero y cuarto.

    Los cuadros de imagen del array de control aparecerán seleccionados en el formulario.

11. Abra la ventana Propiedades y asigne las propiedades que se muestran en la siguiente tabla. (Después de establecer las propiedades de los cuadros de imagen, pulse el botón de orden para asignar sus propiedades).

Objeto	Propiedad	Valor
Array de control Image1	BorderStyle	1 - Fixed Single
	Stretch	True
Command1	Caption	«Mostrar iconos»

12. Pulse dos veces el botón Mostrar Iconos del formulario para ver en pantalla el procedimiento de suceso de este botón de orden.

    En la ventana Código aparecerá el procedimiento Command1_Click.

13. Aumente el tamaño de la ventana Código y escriba el siguiente bucle For..Next.

```
For i = 0 To 3
    Image1(i).Picture = _
      LoadPicture("c:\vb5sbs\less06\misc0" & i & ".ico")
Next i
```

Nota: *En este procedimiento de suceso, la función LoadPicture es demasiado larga para que aparezca en una sola línea en este libro, por lo que se ha dividido en dos líneas utilizando el carácter de continuación de línea de Visual Basic (_). Este carácter se puede utilizar en cualquier parte del programa, exceptuando en el interior de un rótulo o cadena alfabética.*

El bucle utiliza la función LoadPicture para cargar cuatro archivos que contienen la imagen de cuatro iconos y que están almacenados en la carpeta Vb5Sbs Less06 del disco duro. La parte más importante de este bucle es la sentencia

```
Image1(i).Picture = _
  LoadPicture("c:\vb5sbs\less06\misc0" & i & ".ico")
```

que carga los archivos desde el disco fijo. La primera parte de la sentencia,

```
Image1(i - 1).Picture
```

accede a la propiedad Picture de cada uno de los cuatro cuadros de imagen del array de control. Se hace referencia a cada uno de los elementos del array de control mediante sus índices, por lo que podrá nombrarlos de la siguiente forma: Image1(0), Image1(1), Image1(2) e Image1(3). El número entre paréntesis es el valor del índice en el array; en este ejemplo, el valor correcto del índice se calcula restando 1 a la variable contadora.

El nombre de archivo se crea utilizando la variable contadora y el operador de concatenación que fue comentado en el capítulo anterior. La instrucción

```
LoadPicture("c:\vb5sbs\less06\misc0" & i & ".ico")
```

combina una vía de acceso (ruta), el nombre de un archivo y la extensión .ico para crear cuatro nombres válidos para los archivos de icono contenidos en el disco duro. En este ejemplo, los archivos que está cargando en los cuadros de imágenes tienen los siguientes nombres: Misc01.ico, Misc02.ico, Misc03.ico y Misc04.ico. Esta sentencia funciona debido que en la carpeta \Vb5Sbs\Less06 hay almacenados varios archivos cuyo nombre concuerda con el modelo: Miscxx.ico. La existencia de este modelo es lo que nos permite crear un bucle For...Next para cargar los cuatro nombres de archivo.

Botón Guardar proyecto

14. Pulse el botón Guardar proyecto de la barra de herramientas

    Guarde el formulario en disco con el nombre **MiArrayControl.frm** y guarde después el proyecto con el nombre **MiArrayControl.vbp**.

Botón Iniciar

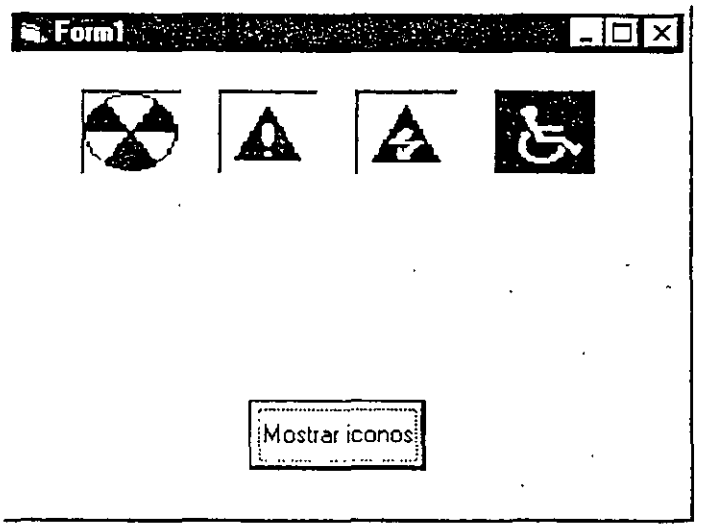
15. Pulse el botón Iniciar contenido en la barra de herramientas para poner en marcha el programa y pulse el botón Mostrar Iconos.

    El bucle For...Next cargará los iconos desde el disco a los cuadros de imagen.

El programa ArrayControl completo se encuentra en la carpeta \Vb5Sbs\Less06.

Truco: Si Visual Basic muestra un mensaje de error, compruebe que el código introducido es correcto para localizar posibles errores de mecanografía. Verifique posteriormente que los archivos de icono se encuentran en la vía de acceso especificada en el programa. Si ha instalado los archivos de prácticas del libro Aprenda Visual Basic Ya *en una carpeta distinta de la propuesta por defecto por el programa de instalación, o si ha movido los archivos de icono a otra ubicación, puede que la vía de acceso utilizada en el procedimiento de suceso no sea la correcta.*

El programa mostrará la siguiente salida:



16. Pulse el botón Cerrar de la barra de títulos para abandonar el programa.
    El programa se detendrá y volverá a aparecer el entorno de programación.

Empleo de la palabra clave Step en el programa MiArrayControl

Imagine que la carpeta \Vb5Sbs\Less06 está repleta de archivos con nombres que siguen el modelo Misexx.ico. Intente ahora utilizar la palabra clave Step para mostrar algunos iconos nuevos en los cuadros de imagen. Para cargar archivos cuyos nombres sigan un modelo distinto bastará con cambiar los números que aparecen detrás de la sentencia For y modificar el código que crea el índice del array de control y el nombre de archivos de icono. También deberá cambiar los índices de los cuadros de imagen para que coincidan con los nuevos valores del contador que esté utilizando.

Modificación del programa MiArrayControl

1. Seleccione el primer cuadro de imagen del formulario y después abra la ventana Propiedades.
   En la ventana Propiedades aparecerán las propiedades asociadas con el cuadro de imagen Image1(0).
2. Cambie el valor de la propiedad Index a 22.
   El primer elemento de un array de control suele tener el valor de índice 0. Sin embargo, podrá cambiar el valor de este índice si ello le facilita el manejo del array de control. En este ejercicio, los archivos que va a abrir son Misc22.ico, Misc24.ico, Misc26.ico y Misc28.ico, por lo que deberá cambiar los índices de los objeto a 22, 24, 26 y 28 para facilitar la referencia a los objetos.

3. Abra el cuadro de lista desplegable de objetos de la ventana Propiedades y seleccione el nombre de objeto Image1(1).
4. Asigne a la propiedad Index del segundo cuadro de imagen el valor 24.
5. Abra el cuadro de lista desplegable y pulse el nombre de objeto Image1(2).
6. Asigne a la propiedad Index del tercer cuadro de imagen el valor 26.
7. Abra el cuadro de lista desplegable objeto y pulse el nombre de objeto Image1(3).
8. Asigne a la propiedad Index del cuarto cuadro de imagen el valor 28.

   A continuación le mostraré cómo modificar el código del bucle For...Next.
9. Realice una doble pulsación sobre el botón de orden Mostrar Iconos.

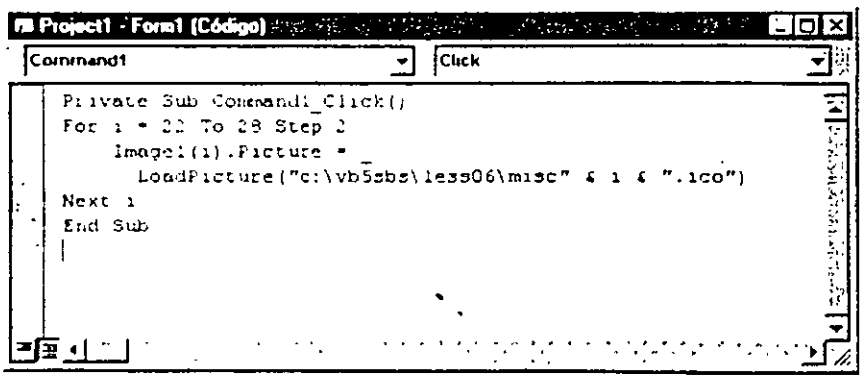
   En la ventana Código aparecerá el procedimiento de suceso Command1_Click.
10. Cambie la primera sentencia For por la siguiente:

```
For i = 22 To 28 Step 2
```

    Este código asigna a la variable contador el valor 22 en el primer bucle, el valor 24 en el segundo y los valores 26 y 28 en los bucles siguientes. Actualice ahora la función LoadPicture del bucle.
11. Cambie el índice del array de control de Image1(i-1) a image1(i).

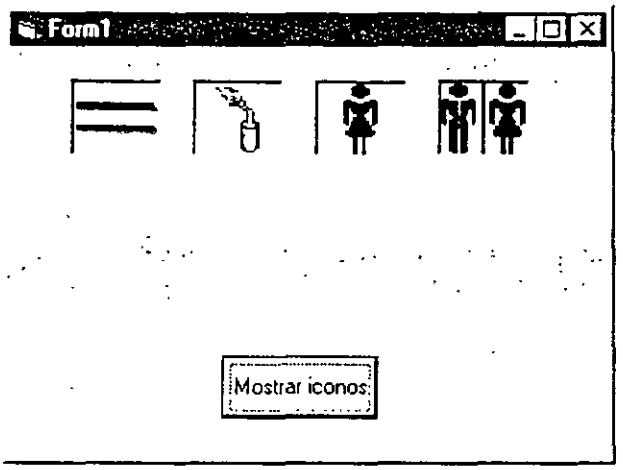
    Como el índice coincide ahora exactamente con el bucle y con los nombres de los objetos imágenes, no hará falta realizar ningún cálculo para determinar el índice.
12. Borre el segundo cero (0) de la vía de acceso (ruta) dentro de la función LoadPicture.

    Los nombres de los archivos (Miscxx.ico) ya no contendrán ceros. Cuando haya terminado, su procedimiento de suceso deberá ser similar al siguiente:



13. Pulse el botón Iniciar de la barra de herramientas y pulse después el botón Mostrar Iconos.

    El bucle For...Next cargará ahora cuatro nuevos iconos en los cuadros de imagen. Su pantalla deberá tener un aspecto similar a la mostrada en la figura siguiente.



Nota: En las versiones comerciales de Visual Basic, la carpeta \Archivos de programa\DevStudio\VB\Iconos\Misc contiene docenas de iconos similares a los que se han mostrado en estas páginas. Esta carpeta no se encuentra instalada en su disco fijo por defecto. Si no consigue localizar esta carpeta tendrá que volver a ejecutar el programa de instalación de Visual Basic 5, seleccionar Agregar/Quitar, y seleccionar la opción Gráficos/Biblioteca de iconos. A partir de ese momento, podrá utilizar libremente dichos iconos como botones o como imágenes gráficas en todos sus programas.

14. Pulse el botón Cerrar de la barra de títulos.

    El programa se detendrá y volverá a aparecer el entorno de programación.

15. Guarde el formulario y el proyecto modificados en disco con el nombre **MiBucleStep**.

El programa BucleStep completo se encuentra en la carpeta \Vb5Sbs\Less06

Sentencias Exit For

La sentencia Exit For le permitirá salir de un bucle For...Next antes de que el bucle haya terminado su ejecución. De esta forma podrá dar respuesta a un suceso específico que suceda antes de que el bucle se haya ejecutado el número de veces previsto. Por ejemplo, en el siguiente bucle For...Next:

(continúa)

Sentencias Exit For *(continuación)*

```
For i = 1 To 10
    Nombre = InputBox("Escriba su nombre o Introduzca Fin para
             salir.")
    If Nombre = "Fin" Then Exit For
    Print Nombre
Next i
```

el bucle solicita al usuario diez nombres y los irá imprimiendo en el formulario, a menos que introduzca antes la palabra *fin* (en cuyo caso el programa saltará a la primera sentencia que siga a la instrucción Next). Normalmente, la sentencia Exit For va unida a sentencias If. Esta estructura le resultará de gran utilidad para gestionar casos especiales que puedan suceder en un bucle, tal como detenerlo antes de que se alcance un límite predefinido.

ESCRITURA DE BUCLES DO

Los bucles Do ejecutan código hasta que se cumpla una condición específica

Como alternativa a los bucles For...Next podrá utilizar un bucle Do para ejecutar un grupo de sentencias hasta que cierta condición del bucle sea Verdadera. Los bucles Do son útiles especialmente en aquellas ocasiones en las que no se sabe de antemano cuántas veces hay que repetir la ejecución del bucle. Por ejemplo, t vez desee desarrollar una aplicación que permita al usuario introducir nombres en una base de datos hasta que escriba la palabra *Fin* en un cuadro de Entrada. En ese caso podrá utilizar un bucle Do para repetir el bucle indefinidamente hasta que se introduzca la cadena de texto «Fin».

Los bucles Do pueden tener diferentes formatos, dependiendo de dónde y bajo qué condiciones se evalúen. La sintaxis más habitual es:

```
Do While condicion
      bloque de sentencias a ejecutarse
Loop
```

Por ejemplo, las siguientes sentencias forman un bucle Do que procesará la entrada hasta que se introduzca la palabra «*Fin*»:

```
Do While Nombre <> "Fin"
      Nombre = InputBox("Escriba su nombre o introduzca Fin para
               salir.")
      If Nombre <> "Fin" Then Print Nombre
Loop
```

La situación de la condición afecta a la forma en que se ejecuta el bucle Do

La sentencia condicional de este bucle es Nombre <> «Fin», sentencia que el compilador de Visual Basic traducirá como «Repite el bucle mientras la variable Nombre no tenga el valor *Fin*». Este hecho nos recuerda una característica interesante de los bucles Do: si la condición que encabeza el bucle no es Verdade

cuando se evalúa por primera vez la sentencia Do, el bucle nunca llegará a ejecutarse. En este ejemplo, si la variable Nombre *tuviera* el valor «Fin» antes de iniciarse el bucle (probablemente por una asignación anterior en el procedimiento de suceso), Visual Basic ignoraría las sentencias incluidas dentro del bucle y continuará ejecutando la línea siguiente a la palabra Loop. Observe que este tipo de bucle requiere una estructura If...Then adicional para evitar que el valor de salida se muestre cuando el usuario lo escribe.

Si desea ejecutar el bucle al menos una vez deberá colocar la prueba condicional al final del bucle. Por ejemplo, el bucle

```
Do
      Nombre = InputBox("Escriba su nombre o introduzca Fin para
               salir.")
      If Nombre <> "Fin" Then Print Nombre
Loop While Nombre <> "Fin"
```

es esencialmente el mismo que el anterior pero la condición del bucle se verifica después de haber recibido un nombre de la función InputBox. Este modelo tiene la ventaja de actualizar la variable Nombre antes de llevarse a cabo la prueba condicional del bucle, por lo que un valor «Fin» anterior no hará que el programa ignore el bucle. Introduciendo al final la comprobación de la condición del bucle se asegurará de que el bucle se ejecute al menos una vez, aunque a menudo tendrá que añadir unas cuantas sentencias adicionales para procesar los datos.

Cómo evitar los bucles sin fin

Asegúrese de que cada bucle cuenta con una condición de salida válida

Debido a la naturaleza de los bucles Do es muy importante diseñar las condiciones de prueba de forma que todos los bucles tengan un punto de salida. Si un bucle nunca da como resultado el valor Falso, dicho bucle se ejecutará indefinidamente y el programa dejará de responder a las entradas del usuario. Considere el siguiente ejemplo:

```
Do
    Número = InputBox("Escriba un número para elevarlo al cuadrado;
             -1 para salir.")
    Número = Número * Número
    Print Número
Loop While Número >= 0
```

En este bucle, el usuario introducirá número tras número y el programa calculará su cuadrado y lo imprimirá en el formulario. Lamentablemente, cuando el usuario quiera terminar no podrá salir del programa, ya que la condición de salida del bucle prevista no funciona. Cuando el usuario introduce el valor –1, el programa calcula su cuadrado y asigna el valor 1 a la variable Número (el problema podría resolverse creando una condición de salida distinta). Los bucles sin fin pueden ser un serio problema cuando escriba bucles Do. Por fortuna, son fáciles de evitar si revisa sus programas con cuidado.

El siguiente ejercicio muestra el empleo de un bucle Do para convertir temperaturas Fahrenheit a temperaturas Celsius. Este sencillo programa solicita al usuario que introduzca un valor de temperaturas por medio de la función InputBox, convierte la temperatura a grados centígrados y, por último, muestra la salida en un cuadro de mensaje. El programa le enseñará también a ocultar un formulario asignando a la propiedad Visible el valor False.

Conversión de temperaturas utilizando un bucle Do

1. En el menú Archivo, seleccione la opción Nuevo proyecto y pulse Aceptar.
   Visual Basic mostrará un nuevo formulario en el entorno de programación.

2. Abra la ventana Propiedades y asigne el valor False a la propiedad Visible de Form1.

   Puede hacer invisible un formulario en tiempo de ejecución definiendo como False su propiedad Visible.

   Cuando asigne el valor False a la propiedad Visible de un formulario Visual Basic ocultará el formulario durante la ejecución del programa. Con ello, básicamente dejará invisible toda la interfaz de usuario en tiempo de ejecución (no podrá mostrar ningún objeto). Probablemente, ésta es una operación que no desee realizar con demasiada frecuencia, pero resulta una técnica útil cuando se desea que parte o todo el programa trabaje en segundo plano. Como este programa recibe únicamente temperaturas Fahrenheit y las devuelve como temperaturas Celsius; no es mala idea ocultar su formulario. Podrá gestionar la entrada utilizando la función InputBox, mientras que la función MsgBox le mostrará el valor de salida.

3. Realice una doble pulsación sobre el formulario.

   El procedimiento de suceso Form_Load se ejecuta cuando se pone en marcha un programa

   El contenido del procedimiento de suceso Form_Load aparecerá en la ventana Código. En este programa, todo el código se encuentra dentro de este procedimiento.

4. Escriba las siguientes sentencias:

```
Prompt = "Introduzca una temperatura Fahrenheit."
Do
    TempF = InputBox(Prompt, "Fahrenheit a Celsius")
    If TempF <> "" Then
       Celsius = Int((TempF - 40) * 5 / 9 - 40)
       MsgBox (Celsius), , "Temperatura en Celsius"
    End If
Loop While TempF <> ""
```

Estas nueve líneas gestionan los cálculos de la conversión. La primera línea asigna una cadena de texto a la variable Prompt (indicador), que se utilizará después para mostrar un mensaje en el cuadro de entrada. El bucle Do irá solicitando al usuario que repetidamente introduzca temperatur Fahrenheit, convertirá el número a Celsius y mostrará el resultado en

pantalla a través de la función MsgBox. El bucle se ejecutará hasta que el usuario pulse el botón Cancelar, que devolverá un valor vacío o *nulo* a la variable TempF. El bucle comprobará el valor nulo utilizando una instrucción While contenida al final del bucle. Por último, la sentencia de programa

```
Celsius = Int((TempF - 40) * 5 / 9 - 40)
```

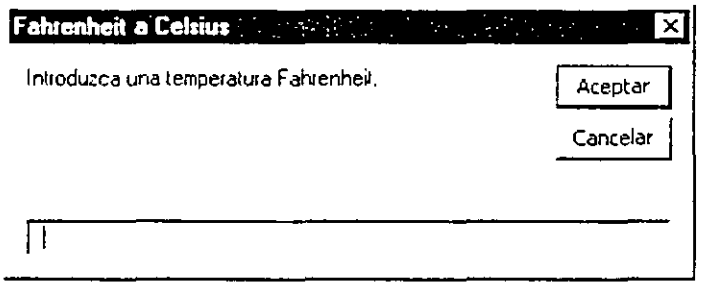
realiza la conversión de Fahrenheit a Celsius en el programa. Esta sentencia utiliza una fórmula de conversión estándar, pero a la hora de devolver el valor en grados centígrados, la variable Celsius utiliza la función Int para evitar las cifras decimales. (Descartará todas las cifras decimales a la derecha de la coma decimal). Aunque de esta forma se sacrifica precisión de cálculo, se evita enfrentarse a números excesivamente largos e inmanejables del tipo 21.111111111111, valor Celsius equivalente a 70 grados Fahrenheit.

Ejecute ahora el programa.

5. Pulse el botón Iniciar en la barra de herramientas.

Botón Iniciar

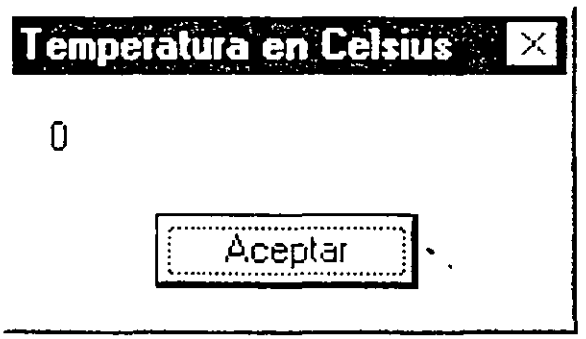
El programa se iniciará y la función InputBox le pedirá que introduzca una temperatura Fahrenheit (el formulario permanecerá invisible). Su pantalla deberá ser similar a la siguiente:



6. Escriba **32** y pulse Aceptar.

La temperatura de 32 grados Fahrenheit se convertirá en 0 grados Celsius, tal como se muesra en el siguiente cuadro de mensaje.

El programa Celsius completo se encuentra en la carpeta \Vb5Sbs\Less06



7. Pulse Aceptar. Escriba **72** en el cuadro de entrada y pulse Aceptar.
   La temperatura de 72 grados Fahrenheit se convertirá en 22 grados Celsius.
8. Pulse Aceptar y después salga del programa pulsando Cancelar en el cuadro de entrada.
   El programa finaliza y en su pantalla volverá a aparecer el entorno de programación.
9. Guarde el formulario y el proyecto en disco con el nombre **MiCelsius**.

Botón Guardar proyecto

Utilización de la palabra clave Until en los bucles Do

Los bucles Do con los que hemos trabajado utilizan la palabra clave While para ejecutar una serie de sentencias mientras que la condición de bucle sea Verdadera. Visual Basic también le permitirá utilizar la palabra clave Until en los bucles Do para ejecutar un bucle *hasta* que cierta condición sea Verdadera. La palabra clave Until puede utilizarse al principio o al final de un bucle Do para evaluar una condición al igual que ocurre con la palabra clave While. Por ejemplo, el siguiente bucle Do utiliza la palabra clave Until para repetir el bucle hasta que el usuario introduzca la palabra *Fin* en un cuadro de entrada:

```
Do
    Nombre = InputBox("Escriba su nombre o introduzca Fin
        para salir.")
    If Nombre <> "Fin" Then Print Nombre
Loop Until Nombre = "Fin"
```

Como puede comprobar, los bucles que utilicen la palabra clave Until son similares a los que utilizan la palabra clave While a excepción de que la condición suele contener el operador contrario; en este caso el operador = (igual a) frente a <> (distinto de). Si le interesa utilizar la palabra Until en sus bucles, no dude en hacerlo.

EMPLEO DE OBJETOS TEMPORIZADORES

Un objeto temporizador es como un reloj invisible dentro de un programa

Visual Basic le permitirá ejecutar un grupo de sentencias durante un *periodo específico de tiempo* mediante el uso de un objeto temporizador. Un *objeto temporizador* es un reloj invisible que le permitirá acceder al reloj del sistema desde el programa. Puede utilizarse igual que un reloj-avisador de cocina para ir descontando tiempo; también le permitirá introducir retrasos temporales en sus programas.

puede ser utilizado para repetir una acción cada vez que transcurran un número determinado de intervalos temporales.

Los objetos temporizadores tienen una precisión de milisegundos, es decir, de 1/1000 segundos. Aunque los temporizadores no están visibles en tiempo de ejecución, cada uno de ellos estará asociado a un procedimiento de suceso que se ejecutará cada vez que el intervalo fijado para el temporizador se haya cumplido. Para fijar el intervalo de un temporizador deberá utilizar la propiedad Interval y para activarlo deberá definir como True la propiedad Enable. Una vez activado el temporizador, éste permanecerá en marcha de forma constante —ejecutando el procedimiento de suceso asociado en los intervalos predefinidos— hasta que el usuario detenga el programa o se desactive el temporizador.

Desarrollo de un reloj digital utilizando un objeto temporizador

Uno de los ejemplos más frecuentes que muestra el empleo de un objeto temporizador es un reloj digital. En el siguiente ejercicio le mostraré cómo crear un sencillo reloj digital que le proporcionará la hora de segundo en segundo. En el presente ejemplo asignará a la propiedad Interval el valor 1000, haciendo que Visual Basic actualice el reloj cada 1000 milisegundos, es decir, una vez por segundo. Al ser Windows un sistema operativo multitarea puede que otros programas estén también utilizando el reloj del sistema o estén consumiendo tiempo de proceso. En estos casos, puede que Visual Basic no tenga la opción de actualizar el reloj cada segundo, pero pronto recuperará el tiempo perdido. Si desea actualizar la hora utilizando otro intervalo (por ejemplo, cada décima de segundo), bastará con ajustar el valor de la propiedad Interval.

La propiedad Interval define la velocidad del contador de un temporizador

Creación del programa Reloj digital

Control Timer

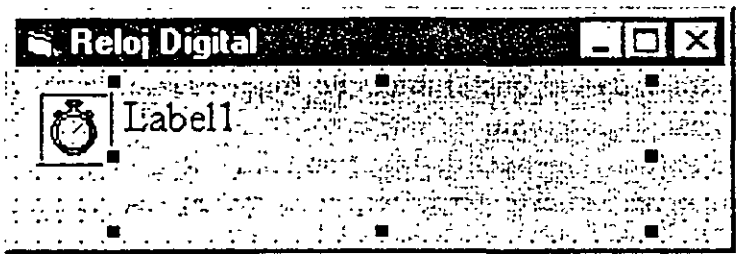
Un objeto temporizador solo puede tener el tamaño especificado por defecto

1. En el menú Archivo, seleccione la opción Nuevo proyecto y pulse Aceptar.
2. Minimice el formulario hasta dejarlo convertido en una pequeña ventana.
   No interesa que el reloj ocupe mucho espacio.
3. Pulse el control Timer (temporizador) contenido en el cuadro de herramientas.
4. Cree un pequeño objeto temporizador situado en la parte izquierda del formulario.
   Cuando haya terminado de crear el temporizador, Visual Basic hará que el temporizador adopte el tamaño estándar.

Control Label

5. Pulse el control Label del cuadro de herramientas.
6. En el centro del formulario, cree una etiqueta que ocupe la mayor parte del mismo.

Esta etiqueta servirá para mostrar la hora del reloj. Su formulario deberá presentar un aspecto similar al siguiente:



La propiedad Caption de un formulario controla el nombre que aparecerá en la barra de títulos del programa.

7. Abra la ventana Propiedades y asigne las siguientes propiedades al programa. Si desea que el programa Reloj digital muestre un título en subarra de títulos, asigne a la propiedad Caption del objeto Form1 el valor «Reloj digital».

Objeto	Propiedades	Valor
Label1	Caption	(Vacío)
	Font	Times New Roman, Negrita, 24 puntos
	Alignment	2 - Center
Timer1	Interval	1000
	Enabled	True
Form1	Caption	«Reloj digital»

Nota: *Si le interesa añadir algún dibujo como fondo de su reloj, asigne a la propiedad Picture del objeto Form1 la vía de acceso de un archivo gráfico.*

El programa Relojdigital completo se encuentra en el disco en la carpeta \Vb5Sbs\Less06.

Ahora deberá introducir el código asociado para el temporizador.

8. Pulse dos veces el objeto temporizador contenido en el formulario

   En la ventana Código aparecerá el procedimiento de suceso Timer1_Timer.

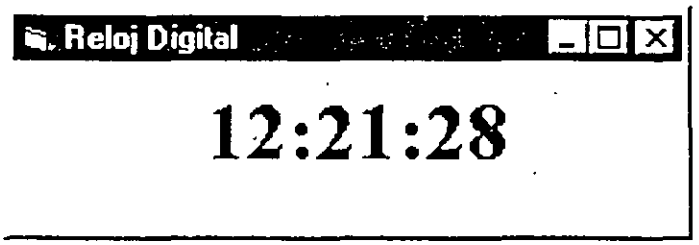
9. Escriba la siguiente sentencia:

Esta sentencia obtiene la hora del reloj del sistema y asigna este valor a la propiedad Caption del objeto Label1. En este programa sólo se necesita esta sentencia, ya que el valor de la propiedad Interval para el temporizador se asigna en la ventana Propiedades. El objeto temporizador se encargará del resto

Botón Iniciar

10. Cierre la ventana Código y pulse el botón Iniciar contenido en la barra de herramientas para poner en marcha el reloj.

    El reloj aparecerá tal como se muestra a continuación.



11. Observe el reloj durante unos instantes.

    Visual Basic actualiza la hora cada segundo.

12. Pulse el botón Cerrar de la barra de título para detener el reloj.

    Pulsar el botón Cerrar es una alternativa al uso del botón Terminar para detener el programa. Éste es el método que utilizaría un usuario para cerrar el reloj si lo estuviera ejecutando como un programa independiente.

Botón Guardar proyecto

13. Utilizando el botón Guardar proyecto almacene el formulario y el proyecto en disco con el nombre **MiRelojdigital**.

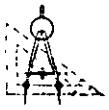
Este pequeño reloj es bastante manejable, y puede que desee compilar el programa **Mi Reloj Digital** dentro de un archivo ejecutable, de forma que pueda utilizarlo en cualquier momento. Adáptelo a sus gustos personales, si lo desea, utilizando sus propios gráficos, texto y colores.

UN PASO MÁS ALLÁ: EMPLEO DE UN OBJETO TEMPORIZADOR PARA DEFINIR UN LÍMITE TEMPORAL

Otro uso interesante de un objeto temporizador consiste en utilizarlo para esperar un periodo concreto de tiempo y, finalmente, activar o prohibir una acción. Esta operación es algo similar a utilizar el reloj del microondas. En primer lugar, deberá definir el valor de la propiedad Interval asignándole el retardo deseado. Seguidamente, pondrá en marcha el reloj asignando a la propiedad Enabled el valor True.

En el siguiente ejercicio se muestra cómo utilizar esta idea para establecer un límite temporal a la tarea de introducción de una contraseña. La contraseña para este programa es «Secreto». El programa utiliza un temporizador que lo cerrará automáticamente si el usuario no introduce una contraseña válida antes de 15 segundos (en general, una aplicación como ésta formaría parte de un programa de mayor envergadura). También podrá utilizar esta técnica de temporización para

mostrar un mensaje de bienvenida o un mensaje de copyright en la pantalla, o para repetir un suceso cada cierto intervalo de tiempo como, por ejemplo, guardar un archivo en el disco cada diez minutos.



Asignación de un tiempo límite para una contraseña

Control TextBox

Control Label

Control CommandButton

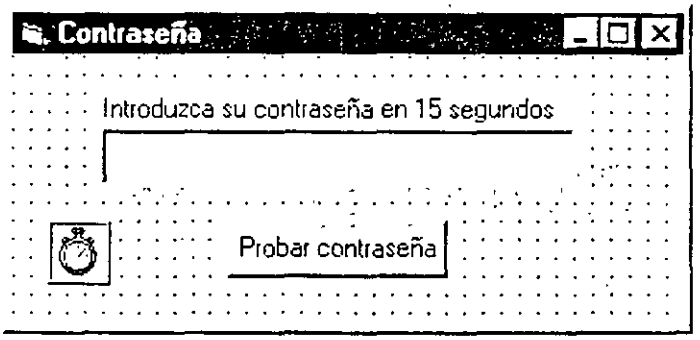
Control Timer

1. En el menú Archivo seleccione la opción Nuevo proyecto y pulse Aceptar.
2. Asigne al formulario el tamaño de una pequeña ventana rectangular más o menos la mitad de un cuadro de entrada.
3. Pulse el control TextBox del cuadro de herramientas.
4. Cree un cuadro de texto rectangular en mitad del formulario.
5. Pulse el control Label del cuadro de herramientas y después cree una etiqueta de mayor tamaño situándola por encima del cuadro de texto.
6. Pulse el control CommandButton del cuadro de herramientas y cree después un botón de orden debajo del cuadro de texto.
7. Pulse el control Timer del cuadro de herramientas.
8. Cree un objeto temporizador en la esquina inferior izquierda del formulario.
9. Asigne las siguientes propiedades al programa:

Objeto	Propiedades	Valor
Text1	Text	(Vacío)
	PasswordChar	*
Label1	Caption	«Introduzca su contraseña en 15 segundos»
Commad1	Caption	«Probar contraseña»
Timer1	Interval	15000
	Enabled	True
Form1	Caption	«Contraseña»

El valor de PasswordChar mostrará asteriscos (*) en el cuadro de texto cuando el usuario introduzca una contraseña. Al asignar el valor 15000 a la propiedad Interval del temporizador se estarán concediendo 15 segundos al usuario para introducir la contraseña y pulsar el botón Probar contraseña. Si asigna a la propiedad Enabled el valor True (valor implícito), el temporizador comenzará a funcionar cuando se inicie el programa. (Esta propiedad también puede desactivarse y activarse en un procedimiento de suceso si no le va a hacer falta utilizar el temporizador hasta más adelante en el programa).

Su formulario tendrá un aspecto similar al siguiente:



10. Realice una doble pulsación sobre el temporizador contenido en el formulario y, finalmente, introduzca las siguientes sentencias:

```
MsgBox ("Lo siento, su tiempo ha expirado.")
End
```

La primera sentencia muestra un mensaje indicando que el tiempo ha terminado, y la segunda sentencia detiene el programa. Visual Basic ejecutará este procedimiento de suceso si el intervalo del temporizador alcanza los 15 segundos y todavía no se ha introducido una contraseña válida.

11. Seleccione el objeto Command1 del cuadro de lista desplegable Objeto y escriba las siguientes sentencias en el procedimiento de suceso Command1_Click:

```
If Text1.Text = "secreto" Then
    Timer1.Enabled = False
    MsgBox ("¡Bienvenido al sistema!")
    End
Else
    MsgBox ("Lo siento, amigo, no le conozco.")
End If
```

Este código de programa comprueba si la contraseña que se ha introducido en el cuadro de texto es «secreto». Si es así, el temporizador se desactivará, aparecerá un mensaje de bienvenida y el programa finalizará. (Un programa más útil podría continuar trabajando en lugar de terminar aquí). Si la contraseña introducida no coincide, el usuario recibirá una notificación mediante un cuadro de mensaje y se le dará otra oportunidad para introducir la contraseña. El usuario tendrá que ser rápido, sólo cuenta con 15 segundos.

12. Cierre la ventana Código y pulse el botón Iniciar para ejecutar el programa.

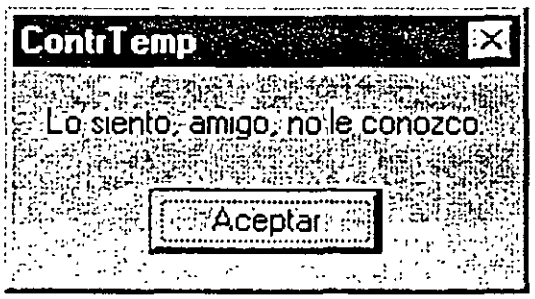
Boton Iniciar

El programa se pondrá en marcha y los 15 segundos del contador comenzarán a transcurrir.

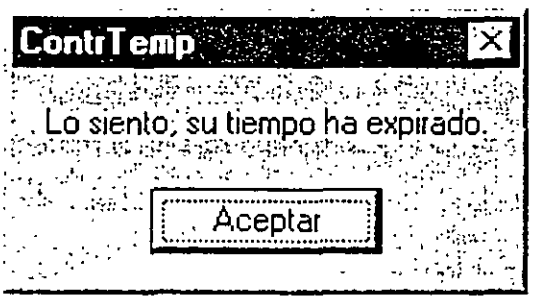
13. Escriba **abrir** en el cuadro de texto y pulse el botón Probar contraseña.

El programa ContrTemp (Contraseña Temporizada) completo se encuentra disponible en la carpeta \Vb5Sbs\Less06

En la pantalla aparecerá el siguiente cuadro de diálogo indicando que la contraseña no es válida:



14. Pulse Aceptar y espere pacientemente hasta que expire el período establecido. El programa mostrará el siguiente mensaje:



Botón Guardar proyecto

15. Pulse Aceptar para terminar el programa.
    Volverá a aparecer el entorno de programación de Visual Basic.
16. Utilizando el botón Guardar proyecto, guarde el formulario y el proyecto en disco con el nombre **MiContrTemp**.

Si desea revisar sus conocimientos de programación

➤ Consulte el ejercicio Revisión y Práctica 2 contenido en el archivo Rev2.doc que se encuentra en la carpeta \Vb5Sbs\Review de su disco fijo. Este documento contiene ejercicios paso a paso que podrá utilizar para poner a prueba sus conocimientos de programación y para revisar el contenido de los capítulos 4 a 6. La aplicación que va a desarrollar le permitirá conseguir un reloj con alarma.

Si desea continuar en el siguiente capítulo

➤ Mantenga en funcionamiento el programa Visual Basic y pase al Capítulo 7.

Si desea salir de Visual Basic por ahora

➤ En el menú Archivo, seleccione Salir.
Si en su pantalla se muestra un cuadro de diálogo Guardar, pulse Sí.

RESUMEN DEL CAPÍTULO

Para	Haga esto

Ejecutar un grupo de instrucciones de programa un número determinado de veces.	Inserte el grupo de sentencias entre las sentencias For y Next de un bucle. Por ejemplo:
	`For i = 1 to 10`
	`  MsgBox("¡Pulse Aceptar!")`
	`Next i`

Mostrar una o más líneas de salida en un formulario	Utilice el método Print. Por ejemplo:
	`For Cnt = 1 to 5`
	`  Print "El valor actual del contador es";`
	`  Cnt`
	`  Print "Cuanto más, mejor"`
	`Next Cnt`

Utilizar una secuencia específica de números con sentencias	Inserte las sentencias en un bucle For...Next y utilice las palabras clave To y Step para definir la secuencia de números. Por ejemplo:
	`For i = 2 To 8 Step 2`
	`  Print i; "...";`
	`Next i`
	`Print "¿Quién lo apreciará?"`

Salir de un bucle For..Next de forma prematura	Utilice la sentencia Exit For. Por ejemplo:
	`For i = 1 to 10`
	`  Nombre = InputBox("Nombre:")`
	`  If Nombre="Trotsky" Then Exit For`
	`  Print Nombre`
	`Next i`

Ejecutar un grupo de sentencias hasta que se cumpla una determinada condición	Inserte el grupo de sentencias entre las sentencias Do y Loop. Por ejemplo:
	`Do While Pregunta <> "Sí"`
	`  Preg = InputBox("¿Trotsky?")`
	`  If Preg <> "Sí" Then Print Preg`
	`Loop`

Para	Haga esto	Botón
Evitar un bucle Do sin fin	Asegúrese de que la condición pueda tomar el valor False	

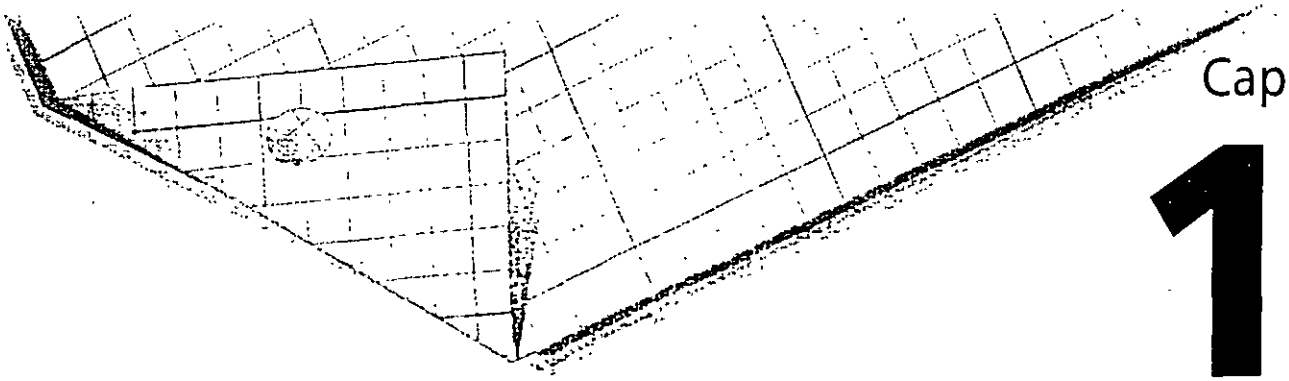
Ejecutar un bucle hasta que una determinada condición sea Verdadera	Utilice un bucle Do con la palabra clave Until. Por ejemplo:
	`Do`
	`  Entrada = InputBox("Diga 'Tío'")`
	`Loop Until Entrada = "Tío"`	

Seguir en un bucle durante un tiempo especificado en el programa	Utilice un objeto temporizador	
Introducir un nombre en la barra de títulos de una aplicación	Asigne el nombre que desee utilizar a la propiedad Caption del objeto Form1 de su aplicación.	

Sobre la ayuda en línea	En el menú Ayuda de Visual Basic, seleccione el mandato Temas de ayuda de Microsoft Visual Basic, seleccione la etiqueta Índice y a continuación
Bucles For...Next	Escriba **For, instrucción**
Bucles Do	Escriba **Do**
El método Print	Escriba **Print, método**
Objetos Temporizadores	Escriba **Timer, control**

AVANCE DEL SIGUIENTE CAPÍTULO

En el Capítulo 7, «Trabajo con formularios, impresoras y manejadores de error», aprenderá otras técnicas relacionadas con la creación de interfaces de usuario. Conocerá cómo añadir otros formularios, cómo enviar un programa como salida a una impresora, y cómo procesar acontecimientos no esperados utilizando los manejadores de suceso.



Capítulo

11

Manejo de archivos de texto y bases de datos

En este capítulo aprenderá a trabajar con información almacenada en archivos de texto y en bases de datos de su sistema. Aprenderá a abrir un archivo de texto y mostrar su contenido usando un objeto cuadro de texto. También descubrirá la forma de crear archivos de texto en el disco. Asimismo, aprenderá a abrir bases de datos existentes utilizando un objeto Data, a buscar un elemento específico y a añadir o borrar registros en/de la base de datos. Microsoft Visual Basic fue especialmente diseñado para crear interfaces personalizadas, o *front ends*, para acceder a bases de datos existentes, por lo que si lo que quiere es personalizar o trabajar con los datos creados con otras aplicaciones, como Microsoft Access, puede hacerlo ahora.

VISUALIZACIÓN DE ARCHIVOS DE TEXTO UTILIZANDO UN CUADRO DE TEXTO

La forma más sencilla de mostrar un archivo de texto en un programa es utilizar un objeto cuadro de texto. Podrá crear cuadros de texto con una gran variedad de tamaños y, si los contenidos del archivo no caben con holgura en el cuadro de texto, podrá añadirle barras de desplazamiento para que el usuario pueda examinar el archivo completo. Para cargar el contenido de un archivo de texto dentro de un cuadro de texto tendrá que utilizar tres instrucciones y una función. Estas palabras reservadas se describen en la siguiente tabla y su funcionamiento se comprobará en el primer ejercicio de este capítulo.

Puede emplear cuatro palabras reservadas de Visual Basic con los archivos de texto.

Palabra reservada	Descripción
Open	Abre un archivo de texto para entrada o salida.
Line Input	Lee una línea de entrada desde el archivo de texto.
EOF	Comprueba el final del archivo de texto.
Close	Cierra el archivo de texto.

Cómo abrir un archivo de texto para entrada

Los archivos de texto contienen caracteres reconocibles.

Un archivo de texto consta de una o más líneas de números, palabras o caracteres. Los archivos de texto son distintos de los archivos de documentos, que contienen códigos de formato, y de los archivos ejecutables, que contienen instrucciones para el sistema operativo. Los archivos de textos típicos de su computadora son reconocidos como «Documentos de texto» por el Explorador de Windows o tendrán las extensiones .txt, .ini, .log, .inf, .dat o .bat. Como los archivos de texto contienen únicamente caracteres reconocibles, podrá mostrarlos fácilmente utilizando los objetos cuadro de texto.

Un objeto de diálogo común muestra el cuadro de diálogo común Abrir

Podrá permitir que el usuario decida qué archivo de texto desea abrir en un programa utilizando un objeto de diálogo común para solicitar al usuario que especifique la ruta completa del archivo. Los objetos de diálogo común son compatibles con el método ShowOpen, que muestra en pantalla el cuadro de diálogo común denominado Abrir. La ruta seleccionada por el usuario desde el cuadro diálogo pasa al programa mediante la propiedad FileName y podrá utilizar este nombre para abrir el archivo. El objeto de diálogo común no abre el archivo; sólo obtiene el nombre de ruta completo.

La sentencia Open

Después de obtener del usuario el nombre de la vía de acceso al archivo podrá abrirlo en el programa utilizando la sentencia Open. La sintaxis para la sentencia Open es:

```
Open ViaDeAcceso For modo As #númeroArchivo
```

Los siguientes argumentos son importantes:

- *ViaDeAcceso* o ruta es una vía de acceso válida de Microsoft Windows.
- *Modo* es una palabra reservada que indica cómo se abrirá el archivo (usaremos los modos Input y Output en este capítulo).
- *NúmeroArchivo* es un número entero comprendido entre 1 y 255.

El número de archivo se asociará al archivo cuando éste se abra. De esta forma, tendrá que emplear este número de archivo en su programa cada vez que desee

hacer referencia al archivo abierto. No hay nada especial sobre los números de archivos; Visual Basic simplemente los usa para poder controlar a todos los archivos que tiene abiertos en un programa.

Una sentencia Open típica que utilizara un objeto de diálogo común tendría un aspecto similar a la siguiente:

```
Open CommonDialog1.Nombrearchivo For Input As #1
```

En este caso, la propiedad CommonDialog1.Nombrearchivo representa la vía de acceso, Input es el modo y 1 es el número de archivo.

Nota: Los archivos de texto que se abren empleando esta sintaxis se denominan archivos secuenciales, *porque sus contenidos se deben examinar en orden secuencial. En contraste, podrá acceder en cualquier orden a la información almacenada en una base de datos.*

El siguiente ejercicio muestra cómo podrá usar un objeto de diálogo común y la sentencia Open para abrir un archivo de texto. Este ejercicio también muestra cómo podrá emplear las palabras reservadas LineInput y EOF para mostrar el contenido de un archivo de texto en un cuadro de texto y cómo podrá usar la palabra reservada Close para cerrar un archivo.

Ejecución del programa Visor de Texto

1. Inicie Visual Basic si no está ya en marcha.
2. Pulse el botón Abrir proyecto contenido en la barra de herramientas.

Botón Abrir proyecto

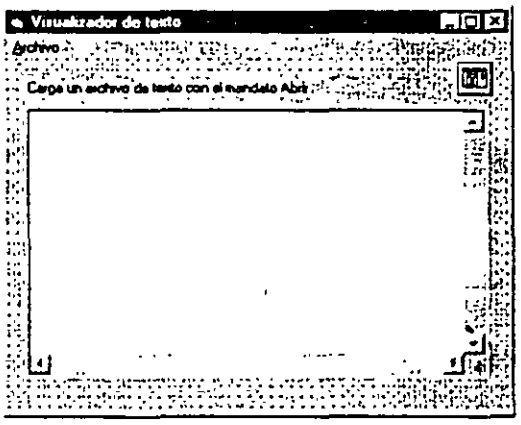
3. Seleccione la carpeta \Vb5Sbs\Less11 y pulse dos veces el nombre de archivo VisorTexto.

   El programa VisorTexto se carga en el entorno de programación.

El programa VisorTexto completo se encuentra en el disco en la carpeta \Vb5Sbs\Less11

4. Si el formulario no se muestra en pantalla, seleccione el formulario VisorTexto en la ventana Proyecto, y pulse sobre el botón Ver objeto.

   Aparecerá el formulario VisorTexto tal como se muestra en la figura:

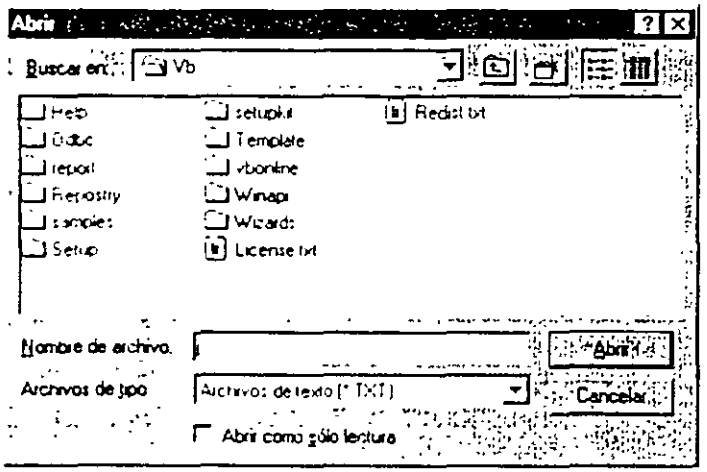


El formulario contiene un gran cuadro de texto que incluye barras de desplazamiento. También contiene un objeto de diálogo común, una etiqueta que proporciona instrucciones para manejar el programa y un menú Archivo que contiene los mandatos Abrir, Cerrar y Salir. También se han definido las propiedades mostradas en la siguiente tabla (observe especialmente las propiedades del cuadro de texto):

Objeto	Propiedad	Valor
txtArchivo	Enabled	False
	Multiline	True
	Name	txtArchivo
	ScrollBars	3-Both
	Text	(Vacio)
mnuItemCerrar	Enabled	False
	Name	mnuItemCerrar
lblArchivo	Caption	«Carga un archivo de texto con el mandato Abrir»
	Name	lblArchivo
Form1	Caption	«Visualizador de Texto»

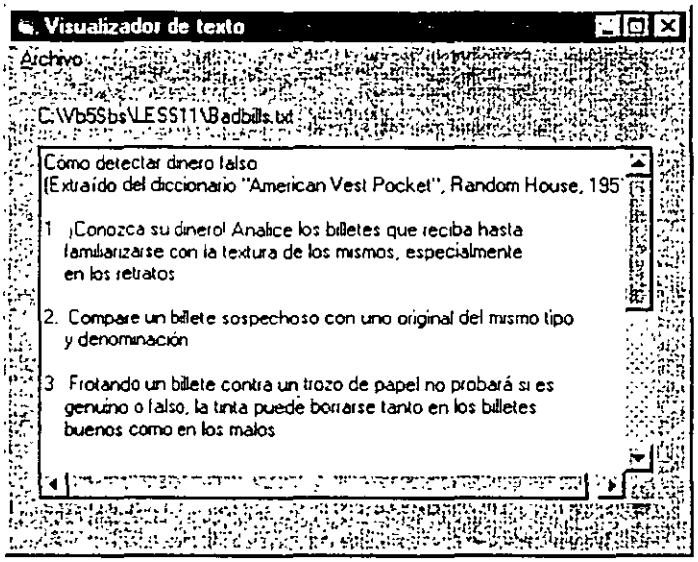

Botón Iniciar

5. Pulse el botón Iniciar contenido en la barra de herramientas.
   El programa Visualizador de Texto comenzará a ejecutarse.
6. En el menú Archivo del Visualizador de texto ejecute el mandato Abrir.
   En su pantalla aparecerá el cuadro de diálogo Abrir tal y como se muestra en la siguiente figura.



7. Seleccione la carpeta \Vb5Sbs\Less11 y realice una doble pulsación sobre el archivo Badbills.txt que se muestra en el cuadro de diálogo.

Badbills es un archivo de texto que contiene un artículo escrito en 1951 sobre dinero falsificado, y aparecerá en el cuadro de texto tal y como se muestra en la siguiente figura.



8. Utilice las barras de desplazamiento para ver el documento entero. Memorice el número 5.
9. Cuando haya acabado, ejecute el mandato Cerrar del menú Archivo para cerrar el archivo. Finalmente, ejecute el mandato Salir para abandonar el programa.

   El programa se para y el entorno de programación volverá a aparecer en pantalla. A continuación analizará en detalle dos de los procedimientos de suceso importantes del programa.

Examen del código del programa Visualizador de Texto

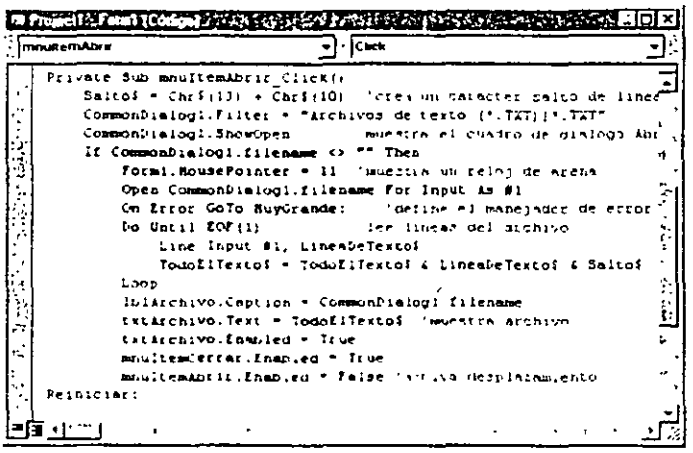
1. Abra el menú Archivo del formulario Visualizador de Texto y pulse sobre el mandato Abrir.

   En la ventana de Código aparecerá el procedimiento de suceso denominado mnuItemAbrir_Click.
2. Aumente el tamaño de la ventana Código para ver la mayor cantidad posible de código.

   Su pantalla deberá ser similar a la que se muestra en la página siguiente.

   Este procedimiento de suceso ejecuta las siguientes acciones. Las órdenes de Visual Basic empleadas se muestran entre paréntesis:

   - Solicita al usuario que introduzca un nombre de ruta completo utilizando un objeto común de diálogo.



```
mnuItemAbrir                                  Click

Private Sub mnuItemAbrir_Click()
    Salto$ = Chr$(13) + Chr$(10)   'crea un caracter salto de linea
    CommonDialog1.Filter = "Archivos de texto (*.TXT)|*.TXT"
    CommonDialog1.ShowOpen          muestra el cuadro de dialogo Abrir
    If CommonDialog1.filename <> "" Then
        Form1.MousePointer = 11   'muestra un reloj de arena
        Open CommonDialog1.filename For Input As #1
        On Error GoTo MuyGrande:     'define el manejador de error
        Do Until EOF(1)              'lee lineas del archivo
            Line Input #1, LineaDeTexto$
            TodoElTexto$ = TodoElTexto$ & LineaDeTexto$ & Salto$
        Loop
        lblArchivo.Caption = CommonDialog1.filename
        txtArchivo.Text = TodoElTexto$   'muestra archivo
        txtArchivo.Enabled = True
        mnuItemCerrar.Enabled = True
        mnuItemAbrir.Enabled = False   'activa desplazamiento
Reiniciar:
```

- Abre el archivo específico para entrada (Open...For Input).
- Copia el archivo línea a línea en una cadena denominada TodoElTextoS (Line Input).
- Copia líneas hasta que se llega al final del archivo (EOF) o hasta que no haya más espacio en la cadena de texto. La cadena TodoElTextoS tiene espacio para 64KB de caracteres.
- Muestra la cadena TodoElTextoS en el cuadro de texto y se habilitan las barras de desplazamiento.
- Maneja cualquier error que pueda suceder (On Error Go To).
- Actualiza los mandatos del menú Archivo y el puntero del ratón y cierra el archivo (Close).

3. Dedique un momento para analizar cómo funcionan las sentencias en el procedimiento de suceso cmdAbrir_Click, especialmente las palabras reservadas Open, Line Input, EOF y Close.

   Para obtener más información sobre estas sentencias, consulte la ayuda interactiva de Visual Basic. El manejador error MuyGrande: del procedimiento muestra un mensaje y aborta el proceso de carga del archivo si el tamaño de éste supera los 64KB. Este manejador de error es necesario debido a la limitación de 64KB que tiene el objeto cuadro de texto.

   Si selecciona un archivo que contenga varias páginas de texto, Visual Basic tardará algún tiempo en cargarlo. Por esta razón, se emplea la propiedad MousePointer para convertir el puntero del ratón en un reloj de arena hasta que el archivo sea mostrado en la pantalla. Siempre es buena idea dar a los usuarios una realimentación visual de lo que está sucediendo si la espera va a ser superior al segundo.

4. Abra el cuadro de lista desplegable Objeto y pulse sobre el elemento mnuItemCerrar para mostrar el procedimiento de suceso denominado mnuItemCerrar_Click.

Este procedimiento se ejecutará cuando se invoque al mandato Cerrar, e igualmente dicho procedimiento limpia el cuadro de texto, desactiva el mandato Cerrar, activa el mandato Abrir y desactiva el cuadro de texto.

5. Cuando haya terminado de analizar el código de programa del VisorTexto, cierre la ventana Código.

Ahora podrá usar este sencillo programa como base de partida para desarrollar otras utilidades más avanzadas que tengan que trabajar con archivos de texto. En el siguiente ejercicio se desarrollará un programa que le permitirá escribir su propio texto en un cuadro de texto y guardarlo en un archivo en el disco.

CÓMO CREAR UN NUEVO ARCHIVO DE TEXTO EN EL DISCO

Deberá utilizar las palabras reservadas For Output en la sentencia Open cuando quiera crear en el disco un nuevo archivo.

Para crear un nuevo archivo de texto en el disco utilizando Visual Basic, deberá volver a utilizar muchos de los objetos y palabras reservadas empleados en el ejemplo anterior. Crear nuevos archivos en el disco puede resultarle de gran utilidad si tiene pensado generar informes o documentos personalizados, almacenar importantes cálculos o valores o crear un procesador de texto o editor de texto con algún propósito especial. A continuación se muestran los pasos que tendrá que dar para desarrollar el programa:

1. Permitir la introducción de texto o cálculos matemáticos, o ambas cosas, por parte del usuario.
2. Asignar los resultados del proceso a una o más variables. Por ejemplo, podrá asignar el contenido de un cuadro de texto a una variable llamada InputForFileS.
3. Solicitar al usuario que indique el nombre completo de una vía de acceso para almacenar el archivo utilizando el cuadro de diálogo común Guardar como. En este caso utilizará el método ShowSave de un objeto de diálogo común para mostrar el cuadro de diálogo.
4. Utilizar la vía de acceso recibida en el cuadro de diálogo para abrir el archivo donde enviar la salida (Open...For Output).

La instrucción Print# envía la salida al archivo especificado

5. Utilizar la sentencia Print# para guardar uno o más valores en el archivo abierto (Print #).
6. Cerrar el archivo cuando se haya acabado con todas las operaciones (Close).

El siguiente ejemplo muestra cómo puede usar el cuadro de texto y los objetos de diálogo comunes y las instrucciones Open, Print# y Close para escribir una sencilla aplicación que le permita registrar notas. Puede utilizar esta herramienta para tomar notas en casa o en el trabajo y luego guardarlas en el disco una vez introducida la fecha del sistema.

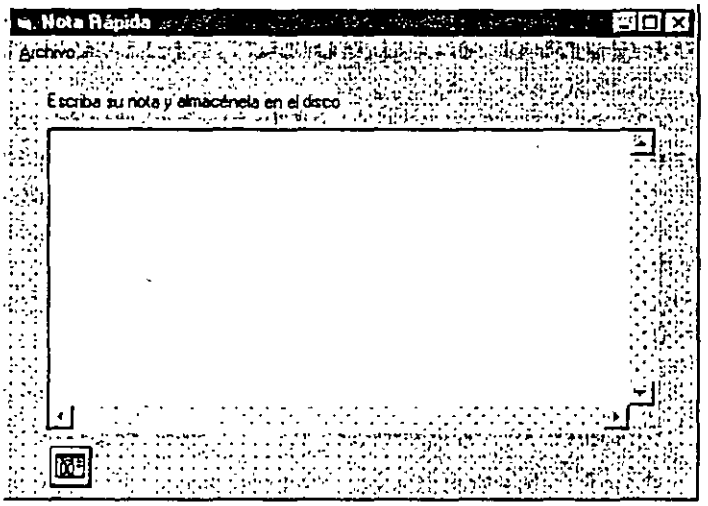
Ejecución del programa Notas rápidas (NotaR)

Botón Abrir proyecto

1. Pulse el botón Abrir proyecto de la barra de herramientas.
2. Seleccione la carpeta \Vb5Sbs\Less11 y luego realice una doble pulsación sobre el proyecto NotaR.

   El programa NotaR se cargará en el entorno de programación.
3. Si el formulario no es visible, pulse el formulario NotaR contenido en la ventana Proyecto y, finalmente, pulse sobre el botón Ver Objeto.

   El formulario NotaR aparecerá, tal como se muestra en la siguiente figura:



El formulario contiene un gran cuadro de texto que incluye barras de desplazamiento. También cuenta con un objeto de diálogo común, una etiqueta que muestra las instrucciones de manejo del programa y el menú Archivo que contiene los mandatos Guardar como, Insertar Fecha y Salir. Se han asignado los siguientes valores a las propiedades:

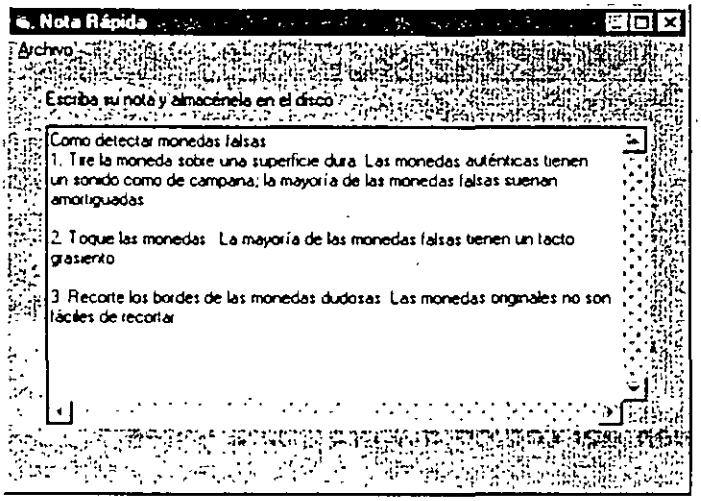
Objeto	Propiedad	Valor
txtNote	Multiline	True
	Name	txtNote
	ScrollBars	3-Both
	Text	(Vacío)
Label1	Caption	«Escriba su nota y almacénela en el disco»
Form1	Caption	«Nota rápida»

Botón Iniciar

4. Pulse el botón Iniciar de la barra de herramientas.
5. Escriba el siguiente texto, o alguno de cosecha propia, en el cuadro de texto:

```
Cómo detectar monedas falsas

1. Tire la moneda sobre una superficie dura. Las monedas
auténticas tienen un sonido como de campana; la mayoría de las
monedas falsas suenan amortiguadas.

2. Toque las monedas. La mayoría de las monedas falsas tienen
un tacto grasiento.

3. Recorte los bordes de las monedas dudosas. Las monedas
originales no son fáciles de recortar.
```

Cuando haya terminado, su pantalla será similar a la siguiente:



Truco: Para copiar un texto desde el Portapapeles de Windows hasta el cuadro de texto, pulse MAYUS+INS. *Para copiar texto desde el cuadro de texto hacia el Portapapeles de Windows, seleccione el texto utilizando el ratón y pulse simultáneamente* CTRL+C.

Pruebe ahora los mandatos contenidos en el menú Archivo.

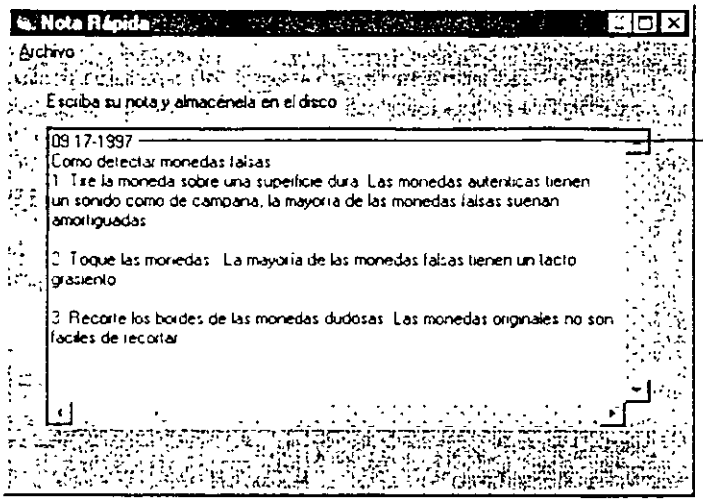
6. En el menú Archivo, ejecute el mandato Insertar Fecha.

   La fecha actual aparecerá en la primera línea del cuadro de texto como se muestra en la figura de la página siguiente.

   El mandato Insertar Fecha es un método eficaz y rápido para incluir la fecha actual en un archivo. Se trata de una función de gran utilidad para el caso de que desee crear un diario o un libro de anotaciones.

7. En el menú Archivo, ejecute el mandato Guardar como.

8. En el cuadro de diálogo Guardar como, seleccione la carpeta \Vb5Sbs\Less11 si es que todavía no se encuentra seleccionada. Introduzca el nombre Monfalsa.txt en el cuadro de texto Nombre de archivo y pulse el botón Guardar.



El texto que acaba de introducir se almacenará en el disco con el nombre Monfalsa.txt. Ahora ya podrá salir del programa.

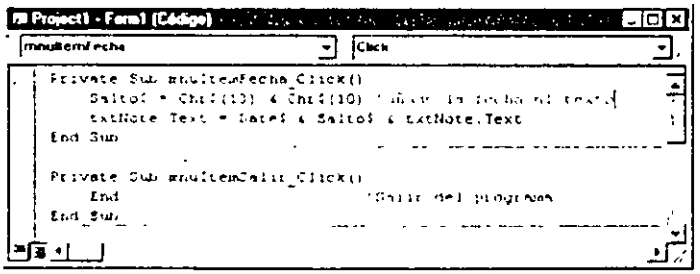
9. En el menú Archivo, ejecute el mandato Salir.

   El programa se parará y volverá a aparecer el entorno de programación. Ahora le invito a que analice los procedimientos de suceso contenidos en el programa.

Examen del código del programa NotaR

1. En el menú Archivo del formulario NotaR, pulse el mandato Insertar Fecha.

   En la ventana Código aparecerá el procedimiento de suceso denominado mnuItemFecha_Click, tal y como se muestra en la siguiente figura:



La función Date$ obtiene la fecha del sistema

Este procedimiento de suceso añade la fecha actual al cuadro de texto uniendo o *concatenando* la fecha actual (generada por la función Date$), un retorno de carro y la propiedad Text. Podrá utilizar una técnica similar para añadir la fecha actual o cualquier otra información al texto contenido en el cuadro de texto.

2. Analice durante unos instantes cómo funciona la instrucción de concatenación y, a continuación, abra el procedimiento de suceso mnuItemGuardar_Click.

   Este bloque de instrucciones utiliza un objeto de diálogo común para mostrar el cuadro de diálogo Guardar como, abre el archivo de salida con el número 1, escribe el valor de la propiedad txtNote.Text en el disco utilizando la sentencia Print# y, finalmente, cierra el archivo de texto. Analice con especial atención la instrucción

   ```
   Print #1, txtNote.Text
   ```

   que asigna el contenido completo del cuadro de texto al archivo abierto. Print# es similar al método Print, salvo en que la salida va a un archivo especificado en lugar de a la pantalla o a la impresora. Lo importante aquí es que se almacena el contenido completo del archivo en la cadena txtNote.Text.

La sentencia Print# necesita un número de archivo como primer argumento.

3. Dedique unos segundos a revisar las órdenes Open, Print# y Close y luego cierre la ventana Código.

Ya ha acabado con el programa Notas Rápidas. Ha aprendido lo suficiente en estos dos ejemplos como para avanzar un buen trecho con archivos de textos. Para obtener más información sobre cómo manejar los archivos secuenciales, consulte la ayuda interactiva de Visual Basic.

GESTIÓN DE BASES DE DATOS

Como aprendió en el Capítulo 2, una base de datos es una colección organizada de información almacenada electrónicamente en un archivo. Podrá crear potentes bases de datos usando una gran variedad de productos de bases de datos, incluyendo Microsoft Access, Microsoft FoxPro, Btrieve, Paradox y dBASE. También puede usar bases de datos cliente-servidor ODBC tal como Microsoft SQL Server.

Visual Basic es compatible con una gran variedad de formatos de bases de datos.

Si trabaja regularmente con bases de datos, especialmente con las nombradas anteriormente, debería pensar en Visual Basic como una herramienta potente para mejorar y mostrar su información. Dado que Visual Basic incluye la misma tecnología de bases de datos que Microsoft Access (un gestor de bases de datos llamado Microsoft Jet), podrá crear aplicaciones de bases de datos personalizadas sin más que utilizar una docena de líneas de código de programa.

En este apartado aprenderá a utilizar un objeto de datos de Visual Basic para gestionar la base de datos Biblio.mdb incluida con el software de Visual Basic y dentro de Access. Biblio.mdb es una base de datos desarrollada con Access que contiene un índice bibliográfico de libros que analizan el tema de la programación con bases de datos. En el siguiente ejemplo aprenderá a mostrar múltiples registros de un base de datos, a buscar una información específica, a añadir nuevos registros, a borrar registros inservibles y a realizar copias de seguridad de sus ar-

chivos de datos. Cuando haya terminado, estará en disposición de ponerse a trabajar para construir sus propias aplicaciones de bases de datos.

Cómo crear aplicaciones de bases de datos particularizadas

Las aplicaciones personalizadas de bases de datos muestran listas personalizadas de registros y campos de cualquier base de datos

Una aplicación personalizada de bases de datos es un programa que coge los campos y registros de una base de datos y los muestra en una forma determinada, significativa para un gran número de usuarios. Por ejemplo, una biblioteca pública podría crear una versión personalizada de su catálogo de libros para un grupo de científicos. Las bases de datos personalizadas habitualmente presentan una gran variedad de órdenes a sus usuarios. Estos mandatos permiten que los usuarios utilicen filtros de visualización y realicen operaciones de búsqueda, impresión, adición y borrado de registros y puedan crear copias de seguridad de su base de datos. Debido a las peculiaridades de su diseño y su consiguiente evolución, algunas bases de datos están organizadas de una manera que hace difícil usarlas en su forma original o en el entorno de base de datos que sirvió para generarla. Con Visual Basic podrá construir una aplicación de base de datos personalizada que muestre únicamente la información que el usuario desee ver y podrá incluir en su nueva aplicación exclusivamente aquellos mandatos necesarios para procesar esos datos.

Empleo de controles ligados para mostrar información contenida en la base de datos

Los controles ligados procesan la información de bases de datos de forma automática

La mayoría de los objetos que puede crear utilizando los controles del cuadro de herramientas de Visual Basic tienen la cualidad intrínseca de mostrar información contenida en bases de datos. En la terminología de bases de datos, estos objetos se denominan *controles ligados*. Un objeto está ligado a una base de datos cuando se asigna a su propiedad DataSource el nombre de una base de datos válida y su propiedad DataField almacena el nombre de una tabla existente en dicha base de datos. Una *tabla* es un grupo de campos y registros que usted o alguien ha definido cuando se creó la base de datos. Tal como comentamos en el Capítulo 2, puede enlazar sus programas Visual Basic con bases de datos utilizando un objeto de datos. Después de que la conexión haya sido establecida, podrá mostrar la información de la base de datos utilizando objetos creados con cualquiera de los controles estándar mostrados en la siguiente tabla:

Control	Descripción
	Cuadro de verificación
	Cuadro combo
	Imagen

Control	Descripción
	Etiqueta
	Cuadro de lista
	Cuadro de dibujo
	Cuadro de texto

Empleo de cuadros de texto para mostrar datos

El siguiente programa utiliza un objeto de datos y cuatro objetos cuadro de texto para mostrar cuatro campos de la base de datos Biblio.mdb. El programa nos muestra cómo puede crear una aplicación de base de datos personalizada para visualizar únicamente la información que desee. En esta aplicación, la propiedad ReadOnly del objeto de datos toma el valor True, para que la información contenida en la base de datos pueda ser visualizada pero no modificada. Si desea permitir que los usuarios puedan realizar cambios en la base de datos Biblio.mdb tendrá que asignar el valor False a la propiedad ReadOnly utilizando la ventana Propiedades.

Truco: Para conservar intacta la base de datos original Biblio.mdb utilice el Explorador de Windows para realizar una copia de seguridad de la misma antes de completar el siguiente ejercicio.

Ejecución del programa VisorDatos

Botón Abrir proyecto

1. Pulse el botón Abrir proyecto de la barra de herramientas.
2. Seleccione la carpeta \Vb5Sbs\Less11 y realice una doble pulsación sobre el programa VisorDatos.vbp.

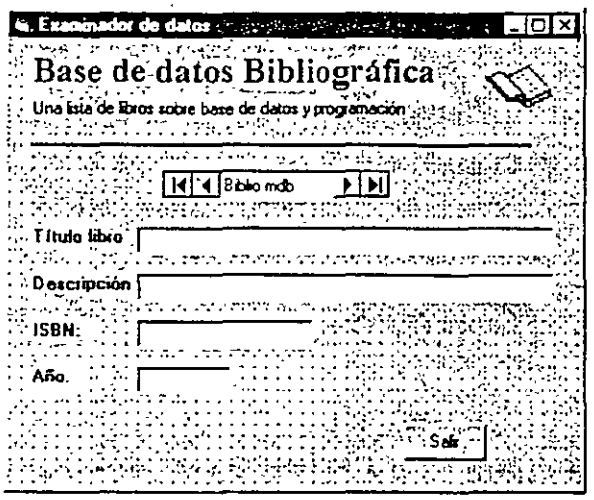
   El programa VisorDatos se cargará en el entorno de programación.

Botón Ver Objeto

3. Si el formulario no se encuentra visible, pulse sobre el formulario VisorDatos en la ventana Proyecto, y pulse el botón Ver Objeto.

   Aparecerá el formulario Examinador de datos, tal como se muestra en la figura de la página siguiente.

   Este formulario contiene información sobre el programa, un dibujo, un objeto de datos, varias etiquetas y cuadros de textos y un botón de orden. En la tabla siguiente se muestran los valores asignados a las propiedades indicadas correspondientes al objeto de datos y a los objetos cuadros de



texto, que son los elementos implicados en la transferencia de datos. Para examinar los valores asignados a las propiedades de los demás objetos utilice la ventana Propiedades.

Objeto	Propiedad	Valor
datBiblio	Caption	«Biblio.mdb»
	Connect	Access
	DatabaseName	«c:\Archivos de programa\devstudio\vb\biblio.mdb»
	Name	datBiblio
	ReadOnly	True
	RecordsetType	0-Table
	RecordSource	Titles
txtTitle	DataField	Title
	DataSource	datBiblio
	Name	txtTitle
	Text	(Vacío)
txtInfo	DataField	Description
	DataSource	datBiblio
	Name	txtInfo
	Text	(Vacío)
txtSBN	DataField	ISBN
	DataSource	DatBiblio
	Name	txtISBN
	Text	(Vacío)
txtYear	DataField	Year Published
	DataSource	datBiblio
	Name	txtYear
	Text	(Vacío)

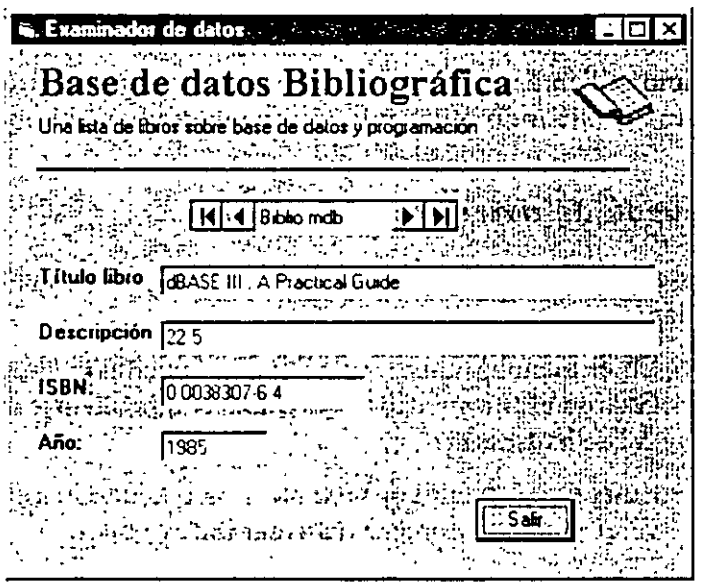
El objeto de datos tiene asignado el valor Titles en la propiedad RecordSource y «c:\Archivos de programa\devstudio\vb\biblio.mdb» en la propiedad DatabaseName; los cuatro cuadros de texto tienen idénticas propiedades DataSource (datBiblio) y valores distintos para el campo DataField. Esto establece la unión básica entre la base de datos bibliográfica almacenada en el disco, el objeto de datos del programa y los campos de texto individuales contenidos en el formulario. Las demás propiedades retocan estas asignaciones básicas.

Nota: Si la copia de Biblio.mdb no está en la carpeta c:\Archivos de programa\devstudio\vb\biblio.mdb (ubicación por defecto), actualice el valor asignado a la propiedad DatabaseName del objeto de datos para corregir la vía de acceso en su programa.

Botón Iniciar

4. Pulse el botón Iniciar de la barra de herramientas.

   El programa VisorDatos comenzará a ejecutarse y el primer registro de la tabla «Títulos» (Titles) contenida en la base de datos Biblio.mdb aparecerá en el formulario.



5. Pulse la flecha interior derecha para ver el segundo registro almacenado en la tabla Titulos.

   El registro correspondiente al libro *The dBASE Programming Language* aparecerá en el formulario. Cada vez que realice un desplazamiento de la base de datos, el contenido de los cuatro cuadros de texto se actualizará.

6. Pulse la flecha exterior derecha para ver el último registro contenido en la tabla Titulos.

Aparecerá el registro correspondiente al libro *1988 National Database and 4-5 Generation Language.*

7. Pulse el botón Salir para detener el programa.
   El programa VisorDatos finalizará.

Este programa contiene una sola línea de código de programa (la instrucción End), pero aun así, le proporcionará bastante información. Lo más interesante de todo esto es que la aplicación es capaz de mostrar únicamente la información almacenada en los campos que desee ver de la base de datos Biblio. Utilizando un objeto de datos y varios cuadros de texto ligados, podrá crear una aplicación de base de datos muy efectiva.

EMPLEO DEL OBJETO RECORDSET

Un objeto Recordset representa los datos con los que está trabajando en el programa.

En el programa VisorDatos utilizó una propiedad denominada RecordsetType para identificar la información de la base de datos como una tabla. En Visual Basic, un *Recordset* es un objeto que representa la parte de la base de datos con la que está trabajando el programa. El objeto Recordset incluye propiedades y métodos especiales con los que podrá buscar, ordenar, añadir y borrar registros. En el siguiente ejercicio utilizará un objeto Recordset para buscar registros almacenados en la base de datos Biblio.mdb utilizando el campo «título» (title).

Búsqueda de datos en Biblio.mdb

Antes de modificar el programa, guárdelo con un nuevo nombre para proteger el original.

1. En el menú Archivo, seleccione la opción Guardar VisorDatos.frm como. Guarde el formulario VisorDatos como **MiBuscarDatos.frm** en la carpeta \Vb5Sbs\Less11.
2. En el menú Archivo, seleccione la opción Guardar proyecto como. Archive el proyecto como **MiBuscarDatos.vbp**.

Control CommandButton

3. Pulse el control CommandButton y cree un objeto botón de orden en la parte inferior izquierda del formulario.
4. Asigne las siguientes propiedades para el objeto botón de orden:

Objeto	Propiedad	Valor
Command1	Caption	«Buscar»
	Name	cmdBuscar

5. Realice una doble pulsación sobre el botón de orden Buscar para abrir el procedimiento de suceso cmdBuscar_Click en la ventana Código.

6. Escriba las siguientes sentencias de programa en el procedimiento de suceso:

```
mensaje$ = "Introduzca el título completo del libro."
'obtiene la cadena que se utilizará en la búsqueda del título
SearchStr$ = InputBox(mensaje$, "Búsqueda del libro")
datBiblio.Recordset.Index = "Title"        'usa título
datBiblio.Recordset.Seek "=", SearchStr$ 'y busca
If datBiblio.Recordset.NoMatch Then        'si no encuentra
                                           'ninguno
    datBiblio.Recordset.MoveFirst          'va al primer
    registro.
End If
```

El método Seek busca un registro que coincida.

Este procedimiento de suceso muestra un cuadro de diálogo de búsqueda que le solicitará al usuario que introduzca una cadena de búsqueda (SearchStr$) y utiliza el método Seek para buscar en el campo Título (Title) de la base de datos de principio a fin hasta que encuentre uno que coincida o hasta que llegue al final de la lista. Si no se encuentra ningún registro coincidente, Visual Basic muestra un mensaje en un cuadro de aviso y el primer registro de Recordset vuelve a aparecer en la pantalla. En la siguiente tabla se muestran las propiedades y métodos de Recordset utilizados en el procedimiento de suceso.

Propiedad o método Recordeset	Descripción
Index	Propiedad utilizada para definir el campo de la base de datos que se utilizará en la búsqueda.
Seek	Método utilizado para buscar el registro. Además del operador =, se pueden utilizar los operadores relacionales > =, >, < = y < para comparar la cadena de búsqueda con el texto de la base de datos.
NoMatch	Propiedad que toma el valor True si no se encuentra el registro buscado.
MoveFirst	Método que define el primer registro contenido en Recordset como el registro activo.

Botón Guardar royecto

7. Cierre la ventana Código y pulse el botón Guardar proyecto para guardar los cambios en disco.

Ejecute ahora el programa.

Ejecución del programa Mi Busca datos

Botón Iniciar

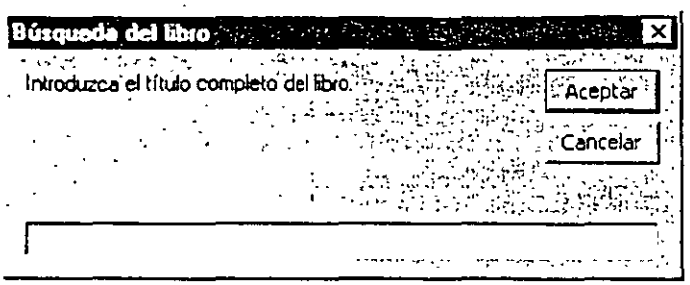
1. Pulse el botón Iniciar de la barra de herramientas.

   La información contenida en la tabla «Títulos» de la base de datos Biblio.mdb aparecerá en los cuadros de texto, igual que antes.

El programa BuscarDatos completo se encuentra en el disco en la carpeta \Vb5Sbs\Less11.

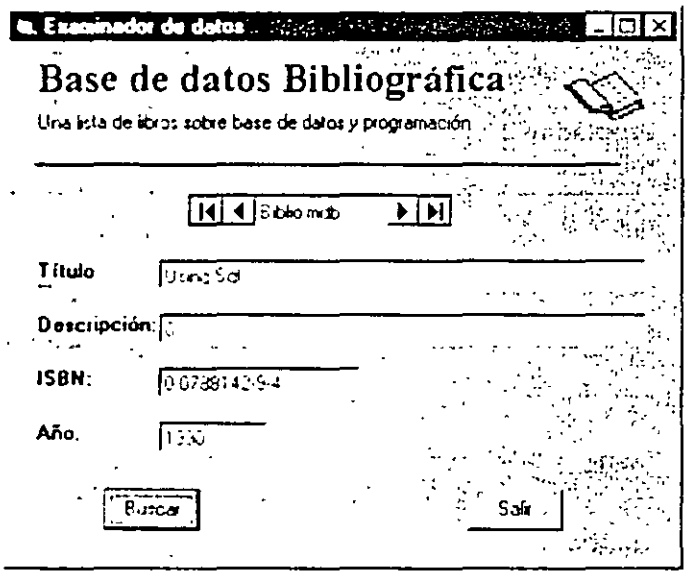
2. Pulse el botón Buscar.

   Aparecerá el cuadro de diálogo Búsqueda del libro, tal y como se muestra en la siguiente figura:



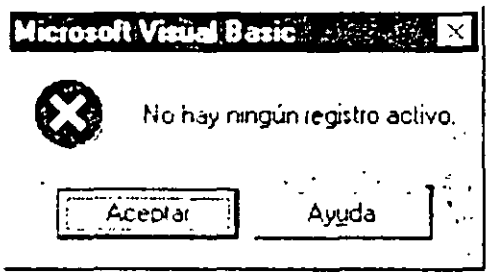
3. Escriba **Using SQL** en el cuadro de diálogo y pulse INTRO.

   El procedimiento de suceso cmdBuscar_Click busca en el campo Título (Title) de la base de datos y detectará el siguiente registro:



4. Pulse de nuevo el botón Buscar, escriba **Pez** y pulse INTRO.

   El procedimiento de suceso no encuentra un libro llamado *Pez*, así que aparecerá el siguiente cuadro de mensaje:



5. Pulse Aceptar para cerrar el cuadro mensaje.

   El registro actual pasa a ser el primer registro contenido en la base de datos (el método MoveFirst realiza bien su trabajo). Sin embargo, aunque Biblio.mdb se encuentra indexada por el campo ISBN, como el procedimiento de suceso cmdBuscar_Click redefine el índice basándose en el campo Title, el primer registro mostrado será aquel correspondiente al que tenga el primer título por orden alfabético (según el contenido del campo Title), en lugar de ser el correspondiente al campo ISBN.

6. Pulse el botón Salir para abandonar el programa.

   El programa se detendrá y aparecerá el entorno de programación.

INCLUSIÓN DE REGISTROS A LA BASE DE DATOS BIBLIO.MDB

Para añadir un nuevo registro a la base de datos tendrá que asignar el valor False a la propiedad ReadOnly en modo diseño. A continuación, tendrá que utilizar el método AddNew en un procedimiento de suceso para abrir un nuevo registro en la base de datos. Cuando aparezca el registro vacío en el formulario, el usuario deberá rellenar los campos necesarios y, una vez finalizado el proceso, deberá moverse a un registro distinto en la base de datos. La forma más sencilla para que el usuario se mueva a un registro diferente es pulsar uno de los botones del objeto de datos. Cuando el usuario se desplace a un registro diferente, el nuevo registro se insertará en la base de datos respetando el orden alfabético.

El siguiente ejercicio muestra cómo se puede utilizar la propiedad ReadOnly y el método AddNew para insertar nuevos registros en una base de datos. La función InputBox proporciona al usuario una realimentación visual durante el proceso.

Cómo permitir que los usuarios añadan registros a la base de datos

Antes de modificar el programa, almacénelo con un nuevo nombre para salvaguardar el original.

1. En el menú Archivo, seleccione el mandato Guardar MiBuscarDatos.frm como. Guarde el formulario como **MiAñadirRegistros.frm**. utilice la orden Guardar proyecto como para guardar el proyecto como **MiAñadirRegistros.vbp**.

2. Pulse el objeto datBiblio (el objeto de datos) en el formulario y luego abra la ventana de propiedades.

3. Cambie la propiedad ReadOnly del objeto datBiblio a False.

   La propiedad ReadOnly determina cómo será abierta la base de datos Biblio.mdb. Asignando a esta propiedad el valor False, permitirá que los

usuarios del programa puedan hacer cambios en la base de datos e inserta registros nuevos.

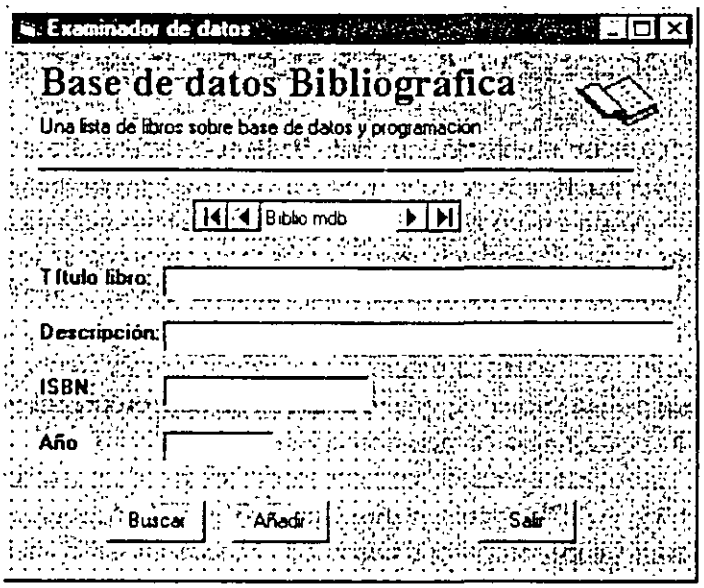
Control CommandButton

4. Pulse el control CommandButton y cree un nuevo objeto botón de orden a la derecha del botón Buscar contenido en el formulario.

5. Asigne las siguientes propiedades para el nuevo objeto botón de orden:

Objeto	Propiedad	Valor
Command1	Caption	«Añadir»
	Name	cmdAñadir

Su formulario deberá tener ahora un aspecto similar al siguiente:



6. Pulse dos veces el botón Añadir para abrir el procedimiento de suceso cmdAñadir_Click en la ventana Código.

7. Escriba las siguientes sentencias de programa en el procedimiento de suceso:

```
mensaje$ = "Introduzca un nuevo registro y pulse el botón
           flecha izquierda."
  reply = MsgBox(mensaje$, vbOKCancel, "Añadir Registro")
  If reply = vbOK Then              'si el usuario pulsa Aceptar
    txtTitle.SetFocus              'mueve el cursor al cuadro de
                                   'título
    datBiblio.Recordset.AddNew     'y obtén un nuevo registro
    'define el campo PubID como 14 (este campo es necesario
    datBiblio.Recordset.PubID = 14  'para Biblio.mdb)
End If
```

El método AddNew añade un nuevo registro a la base de datos

El procedimiento muestra, en primer lugar, un cuadro de diálogo conteniendo instrucciones para introducir datos. La función MsgBox utiliza el argumento vbOKCancel (una constante numérica definida por Visual Basic)

para mostrar un cuadro de diálogo que cuenta con botones Aceptar y Cancelar. Si el usuario pulsa Aceptar, se crea un nuevo registro utilizando el método AddNew. Si el usuario pulsa Cancelar, se ignora la operación. El procedimiento de suceso utiliza también el método SetFocus para situar el cursor en el cuadro de texto Título. El método SetFocus puede ser utilizado para activar cualquier objeto que pueda recibir el foco.

La última sentencia contenida en la estructura de decisión If utiliza código de programa para definir el campo PubID contenido en la base de datos Biblio.mdb. Cada nuevo registro que desee añadir necesitará este campo y, debido a que el visor de bases de datos que está construyendo no permite acceder a este campo, tendrá que insertar el valor para el campo PubID mediante el programa. El valor 14 significa que el libro será publicado por Microsoft Press.

Botón Guardar proyecto

8. Cierre la ventana Código y pulse el botón Guardar proyecto de la barra de herramientas para almacenar los cambios.

Utilice ahora el botón Añadir para añadir un registro a la base de datos.

Ejecución del programa MiAñadirDatos

Botón Iniciar

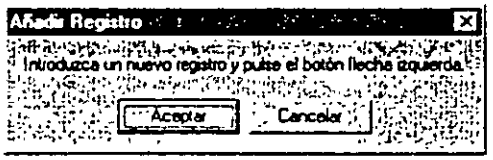
1. Pulse el botón Iniciar de la barra de herramientas.

   En los cuadros de texto contenidos en el formulario aparecerá la información almacenada en los campos correspondientes al primer registro de la tabla Title de la base de datos Biblio.mdb.

El programa AñadirDatos completo se encuentra en la carpeta \Vb5Sbs\Less11 del disco

2. Pulse el botón Añadir.

   Aparecerá el cuadro de diálogo Añadir registro, tal y como se muestra en la siguiente figura:

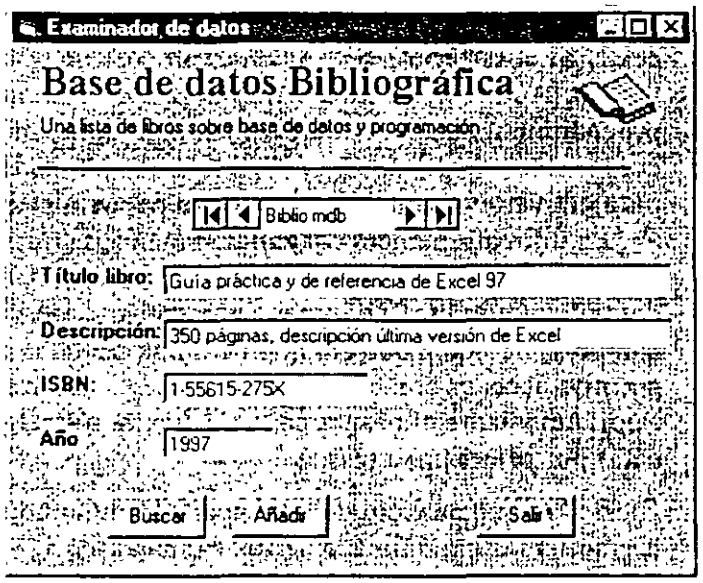


3. Pulse el botón Aceptar.

   En el formulario aparece un nuevo registro en blanco. Introduzca la información que se muestra en la figura de la página siguiente para el nuevo registro. Pulse la tecla TAB para moverse entre los distintos campos.

4. Cuando haya terminado de dar entrada al registro ficticio pulse la flecha exterior izquierda del objeto de datos.

   El registro *Guía Práctica y de referencia de Excel 97* se insertará en la base de datos como último registro. Inmediatamente después en el formulario aparecerá el primer registro de la base de datos.



5. Pulse el botón Buscar, escriba **Guía Práctica y de referencia de Excel 97** y pulse INTRO.

   El registro *Guía Práctica y de referencia de Excel 97* aparecerá en el formulario.

6. Pulse la flecha interior izquierda en el objeto de datos.

   El registro *The NewDirect Marketing* aparecerá en el formulario. Puede utilizar el botón Añadir para añadir cualquier número de registros a la base de datos Biblio.mdb.

7. Pulse el botón Salir para finalizar el programa.

   El programa se detendrá y aparecerá el entorno de programación.

ELIMINACIÓN DE REGISTROS DE LA BASE DE DATOS BIBLIO.MDB

Para borrar registros de una base de datos deberá mostrar el registro que desea borrar y utilizar, posteriormente, el método Delete con el objeto Recordset para eliminar el registro. Antes de abrir la base de datos en el programa deberá asignar el valor False a la propiedad ReadOnly del objeto de datos (ya hizo esta misma operación anteriormente con el método AddNew). Después de borrar el registro de la base de datos tendrá que mostrar un nuevo registro en la base de datos, porque el objeto de datos no realiza esta operación automáticamente. Normalmente, la mejor técnica es usar el método MoveFirst para mostrar el primer registro contenido en la base de datos.

El siguiente ejercicio muestra cómo podrá utilizar Visual Basic para borrar registros de la base de datos Biblio.mdb. Preste particular atención al uso de la función MsgBox en el programa. Debido a que el objeto de datos no proporciona

una utilidad «deshacer», es importante que el programa verifique las intenciones del usuario antes de borrar definitivamente el registro de la base de datos.

Advertencia: *El método Delete borra definitivamente un registro de la base de datos. No proporcione a los usuarios acceso a este método salvo que decida concederles el permiso de borrar registros.*

Cómo permitir que los usuarios borren registros de la base de datos

Antes de modificar el programa MiAñadirDatos deberá almacenarlo con otro nombre para proteger el original.

1. En el menú Archivo, ejecute el mandato Guardar MiAñadirDatos.frm como. Guarde el formulario MiAñadirDatos como **MiBorrarDatos.frm**. Utilice la opción Guardar proyecto como para guardar el proyecto como **MiBorrarDatos.vbp**.
2. Pulse el objeto datBiblio (el objeto de datos) en el formulario y luego abra la ventana Propiedades.
3. Verifique que el valor de la propiedad ReadOnly del objeto datBiblio tiene asignado el valor False.

Control CommandButton

4. Pulse el control CommandButton y cree un nuevo botón de orden a la derecha del botón Añadir en el formulario.
5. Asigne las siguientes propiedades para el objeto botón de orden:

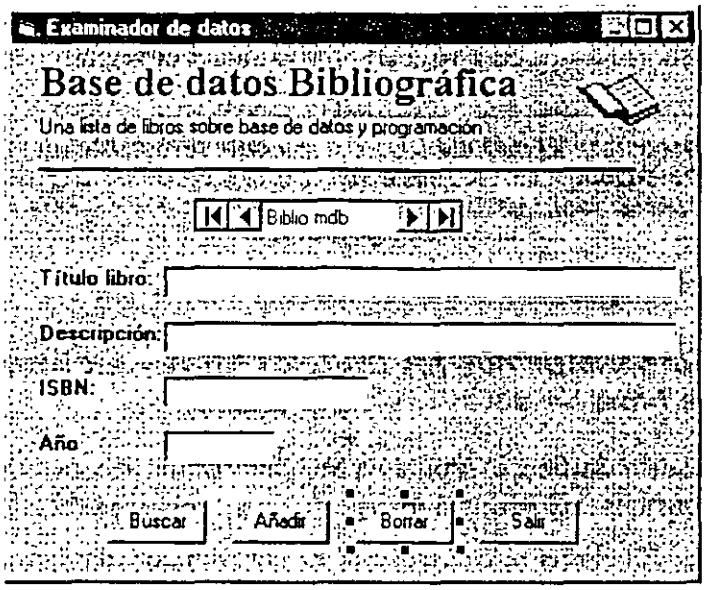
Objeto	Propiedad	Valor
Command1	Caption	«Borrar»
	Name	cmdBorrar —

Su formulario deberá parecerse al que se muestra en la figura de la página siguiente.

6. Pulse dos veces el botón Borrar para abrir el procedimiento de suceso cmdBorrar_Click en la ventana Código.
7. Incluya las siguientes sentencias de programa en el procedimiento de suceso:

El método Delete borra un registro de la base de datos

```
mensaje$ = "¿Seguro que quiere borrar este registro?"
respuesta = MsgBox(mensaje$, vbOKCancel, "Borrar registro")
If respuesta = vbOK Then          'si el usuario pulsa Aceptar
    datBiblio.Recordset.Delete    'borra el registro actual
    datBiblio.Recordset.MoveNext 'mueve al siguiente registro
End If
```



Este procedimiento muestra, en primer lugar, un cuadro de diálogo preguntando al usuario si desea borrar el registro actual. De nuevo, se ha utilizado la función MsgBox con el argumento vbOKCancel para permitir que el usuario anule su operación de borrado si decide no seguir adelante. Si el usuario pulsa Aceptar, el registro actual se borrará utilizando el método Delete y se muestra el siguiente registro utilizando el método MoveNex Si el usuario pulsa Cancelar, la operación de borrado se anula.

Botón Guardar proyecto

8. Cierre la ventana Código y pulse el botón Guardar proyecto para almacenar los cambios realizados.

Utilice ahora el botón Borrar para borrar el registro *Guía Práctica y de referencia de Excel 97* introducido recientemente en la base de datos.

Ejecución del programa MiBorrarDatos

Botón Iniciar

1. Pulse el botón Iniciar de la barra de herramientas.

   La información contenida en la tabla «Títulos» (Title) contenida en la base de datos Biblio.mdb aparece en los cuadros de texto.

El programa BorrarDatos completo se encuentra en el disco en la carpeta \Vb5Sbs\Less11

2. Utilice el botón Buscar para mostrar el registro *Guía Práctica y de referencia de Excel 97*.

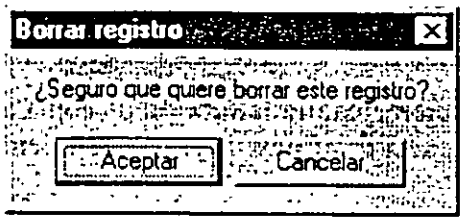
   El registro añadido en el pasado ejercicio aparece en el formulario.

Advertencia: *Los siguientes pasos borrarán este registro de forma permanente en la base de datos*

---

3. Pulse el botón Borrar contenido en el formulario.

   Aparecerá el cuadro de diálogo Borrar registro tal como se muestra en la siguiente figura:



4. Pulse el botón Aceptar para borrar el registro.

   El registro *Guía Práctica y de referencia de Excel 97* se borrará de la base de datos.

5. Pulse el botón Salir para finalizar el programa.

Acaba de terminar su trabajo con el objeto de datos en el presente capítulo. Si desea aprender nuevos temas sobre el desarrollo de aplicaciones para bases de datos, introduzca la palabra Recorsets en la ayuda en línea de VisualBasic y revise las aplicaciones ejemplo incluidas en el programa de Visual Basic. La base de datos Biblio.mdb es utilizada por dichas aplicaciones ejemplo, por lo que reconocerá sin dificultad muchos de los campos y registros que hemos mencionado aquí, en los anteriores ejercicios propuestos.

UN PASO MÁS ALLÁ: CÓMO HACER UNA COPIA DE SEGURIDAD DE UN ARCHIVO

Si su caso es similar al de la mayoría de los usuarios de bases de datos, la información que guarda en ellas tendrá para usted una gran importancia y se verá en una situación comprometida si algo le ocurriera. Por esta razón, siempre es buena idea realizar una copia de seguridad de cada una de sus bases de datos antes de realizar modificaciones sobre ellas. Si ocurriera algún problema mientras trabaja con la copia sólo tendría que sustituir la copia con la base de datos original. Su rutina de copia de seguridad podría incluir un programa de copia de seguridad, el Explorador de Windows o una función especial de copia en su programa de base de datos. Como medida de seguridad adicional puede crear una copia de seguridad de uno o más archivos utilizando un programa desarrollado en Visual Basic sin más que emplear la sentencia FileCopy. FileCopy realiza una copia del archivo (al igual que sucede con la opción Copiar del menú Edición del Explorador de Windows) cuando utilice esta instrucción con la siguiente sintaxis:

La sentencia FileCopy realiza una copia de seguridad de un archivo

```
FileCopy archivoorigen, directoriodestino
```

donde *directorioorigen* es la vía de acceso del archivo que quiere copiar y *directoriodestino* es la vía de acceso del archivo copia de seguridad.

Nota: FileCopy no funciona si el archivo especificado por directorioorigen *está abierto en ese momento.*

En el siguiente ejercicio añadirá la función de copia al programa MiBorrarDatos añadiendo una sentencia FileCopy en el procedimiento de suceso Form_Load.

Empleo de FileCopy para realizar una copia de seguridad de Biblio.mdb

En primer lugar deberá guardar el programa MiBorrarDatos bajo un nuevo nombre para proteger el original. Si el programa MiBorrarDatos no estuviese abierto, cargue el proyecto BorrarDatos desde el disco y abra su formulario.

1. En el menú Archivo, seleccione la opción Guardar MiBorrarDatos.frm como. Almacene el formulario MiBorrarDatos como **MiCopiaSeguridad.frm**. Utilice la opción Guardar proyecto como para guardar el proyecto como **MiCopiaSeguridad.vbp**.

2. Si el formulario no se encuentra visible, seleccione el formulario en la ventana Proyecto, pulse el botón Ver Objeto y realice una doble pulsación sobre el formulario (no sobre uno de sus objetos) para abrir el procedimient de suceso Form_Load en la ventana Código.

   Al colocar la sentencia FileCopy en el procedimiento de arranque obligará al usuario a realizar una copia de seguridad de su base de datos antes de que pueda realizar ninguna otra operación sobre ella.

3. Introduzca las siguientes sentencias de programa en el procedimiento de suceso Form_Load:

```
mensaje$ = _
      "Quiere crear una copia de seguridad de la base
      de datos?"
respuesta = MsgBox(mensaje$, vbOKCancel,
            datBiblio.DatabaseName)
If respuesta = vbOK Then    'copiar la base de datos si el
                            usuario pulsa Aceptar
    FileNm$ = InputBox$ _
              ("Introduzca la ruta para la copia de
              seguridad.")
    If FileNm$ <> "" Then _
        FileCopy datBiblio.DatabaseName, FileNm$
End If
```

La sentencia FileCopy realiza una copia de la base de datos

Esta rutina muestra un cuadro de mensaje cuando se ejecute el programa preguntando al usuario si desea hacer una copia de seguridad de la base de datos. La función MsgBox se usa conjuntamente con el argumnto vbOK-

Cancel para dar al usuario la oportunidad de cancelar el proceso. En esta ocasión, la propiedad DatabaseName se usa también en la función MsgBox para mostrar el nombre de la base de datos en la barra de títulos del cuadro de diálogo. Si el usuario pulsa Aceptar, otro cuadro de mensajes permitirá introducir la vía de acceso del archivo de copia y la orden FileCopy copia el archivo.

Botón Guardar proyecto

4. Cierre la ventana Código y pulse el botón Guardar proyecto para guardar los cambios realizados.

Ejecute ahora el programa para ver cómo funciona la utilidad de copia.

Ejecución del programa MiCopiaSeguridad

Botón Iniciar

1. Pulse el botón Iniciar de la barra de herramientas.

   Un cuadro de diálogo aparece en la pantalla preguntándole si desea, o no, hacer una copia de seguridad de la base de datos:

El programa CopiaSeguridad completo se encuentra en el disco en la carpeta \Vb5Sbs\Less11.

2. Pulse Aceptar para realizar una copia de seguridad.

   Aparecerá un cuadro de diálogo solicitando que introduzca la vía completa del archivo de destino.

3. Escriba **c:\Vb5Sbs\Less11\mibiblio.mdb** y pulse Aceptar.

   Visual Basic copia la base de datos Biblio.mdb al directorio de prácticas \Vb5Sbs\Less11 y le da el nombre de MiBiblio.mdb. Si a partir de ahora cometiera cualquier error con el archivo Biblio.mdb, podría utilizar la copia de seguridad para subsanarlo

4. Pulse el botón Salir para finalizar el programa.

Si desea continuar con el siguiente capítulo

➤ No salga de Visual Basic y pase al Capítulo 12.

Si desea salir de Visual Basic por ahora

➤ En el menú Archivo, seleccione Salir.

Si en su pantalla aparece un cuadro de diálogo que le permite almacenar los cambios, seleccione Sí.

RESUMEN DEL CAPÍTULO

Para	Haga esto
Abrir un archivo de texto	Utilice la sentencia Open...For Input. Por ejemplo: `Open CmnDialog1.FileName For Input As #1.`
Obtener una línea de entrada desde un archivo de texto	Utilice la sentencia Line Input. Por ejemplo: `Line Input #1, LíneadeTextot$`

Detectar el fin de un archivo	Utilice la función EOF. Por ejemplo:
	`Do Until EOF(1)`
	`  Line Input#1, LíneaDeTexto$.`
	`  Texto$ = Texto$ & LíneaDeTexto$ &`
	`  Salto$`
	`Loop`

Cerrar un archivo abierto	Utilice la sentencia Close. Por ejemplo: `Close #1`

Mostrar un archivo de texto	Utilice la sentencia Line Input para copiar el texto desde un archivo abierto a una variable de cadena y asigne después la variable de cadena a un objeto cuadro de texto. Por ejemplo:
	`Do Until EOF (1)`
	`  Line Input #1, LíneaDeTexto$`
	`  Texto$ = Texto$ & LíneaDeTexto$`
	`  & Salto$`
	`Loop`
	`txtDisplay.Text = Texto$`

Mostrar un cuadro de diálogo Abrir	Utilice el método ShowOpen del objeto de diálogo común. Por ejemplo: `CmnDialog1.ShowOpen`
Crear un nuevo archivo de texto	Utilice la sentencia Open...For Output. Por ejemplo: `Open CmnDialog1.FileName For Output As #1`
Mostrar un cuadro de diálogo Guardar como	Utilice el método ShowSave del objeto de diálogo común. Por ejemplo: `CmnDialog1.ShowSave`
Almacenar texto en un archivo	Utilice la sentencia Print #. Por ejemplo: `Print #1, txtNote.Text`
Abrir una base de datos	Utilice el control Data para crear un objeto de datos en un formulario y asigne a su propiedad DatabaseName el nombre de la base de datos. Especifique el tipo de base de datos utilizando la propiedad Connect y especifique el tipo de registro usando la propiedad RecordsetType.
Abrir una base de datos en modo solo lectura	Asigne el valor True a la propiedad ReadOnly del objeto de datos.

Para	Haga esto
Mostrar un campo de datos en un cuadro de texto	Defina las propiedades DataField y DataSource del cuadro de texto.

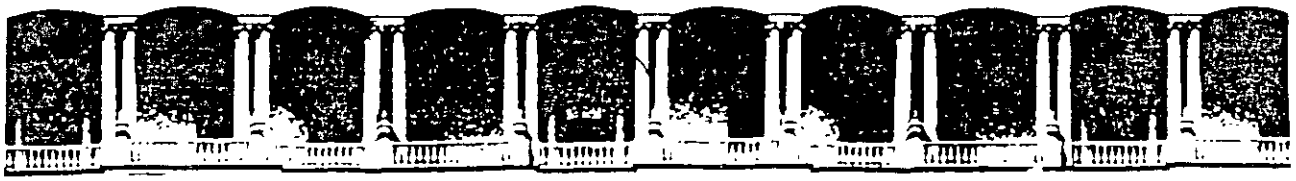
Buscar datos en una base de datos	Solicite al usuario una cadena de búsqueda y utilice las propiedades Index, Seek, NoMatch y MoveFirst del objeto Recordset en un procedimiento de suceso. Por ejemplo:
	`Indicador$ = "Introduzca el título completo del libro"`
	`CadBúsqueda$ = InputBox(Indicador$, "Búscar libro")`
	`datBiblio.Recordset.Index = "Título".`
	`datBiblio.Recordset.Seek "=",CadBúsqueda$`
	`If datBiblio.Recordset.NoMatch Then`
	`  MsgBox ("Libro no encontrado.")`
	`End If`

Añadir un registro a una base de datos	Utilice el método AddNew del objeto Recordset. Por ejemplo: `datBiblio.Recordset.AddNew`
Añadir un campo a un registro de una base de datos utilizando código de programa	Especifique el nombre del campo como una propiedad del objeto RecordSet. Asegúrese de especificar el formato de datos adecuado. Por ejemplo: `datBiblio.Recordset.PubID = 14`
Borrar un registro de una base de datos	Utilice el método Delete del objeto Recordset. Por ejemplo: `datBiblio.Recordset.Delete`
Mostrar el primer registro de una base de datos	Utilice el método MoveFirst del objeto Recordset. Por ejemplo: `datBiblio.Recordset.MoveFirst`
Copiar un archivo	Utilice la sentencia FileCopy. Por ejemplo: `FileCopy datBiblio.DatabaseName, FileNm$`
Proporcionar el foco a un objeto	Utilice el método SetFocus del objeto. Por ejemplo: `txtTitle.SetFocus`

Sobre la ayuda en línea	En el menú Ayuda de Visual Basic, seleccione el mandato Temas de ayuda de Microsoft Visual Basic, seleccione la etiqueta Índice y a continuación
Trabajar con archivos de texto	Escriba **archivos, entrada y salida**
Trabajar con bases de datos	Escriba **control Data**
Propiedades y métodos RecordSet	Escriba **Recordset, propiedad**

AVANCE DEL SIGUIENTE CAPÍTULO

En el siguiente capítulo, «Conexión con Microsoft Office», aprenderá a ejecutar aplicaciones basadas en Windows desde sus programas de Visual Basic. Además, analizaremos la forma de programar objetos de aplicación utilizando Automation y cómo utilizar el Examinador de Objetos para incorporar en sus programas objetos desde aplicaciones tales como Microsoft Word y Microsoft Excel.



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

CURSOS INSTITUCIONALES

"INTRODUCCION A VISUAL BASIC"

Del 16 al 27 de febrero de 1999

Complemento

Ing. Víctor Caamaño Rosas
Palacio de Minería

1999.

```
Datasource: Web SQL
Username: sa
Template: findme.htx
SQLStatment
+SELECT
+from pubs.dbo.authors
+where au_fname like '%fname%%%'and
+au_lname like '%lname%%%'
```

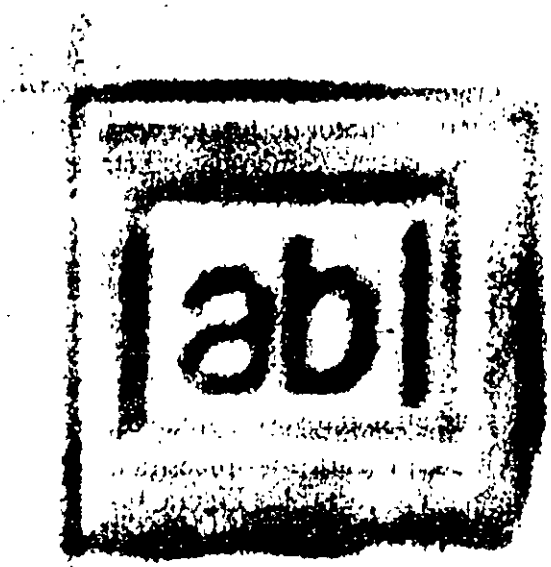
4. El resultado de la consulta SQL se combina en el archivo .IDL con el archivo .HTX de abajo. En la carpeta C:\InetPub\guiones, cree una carpeta llamada findme.htx que contenga los siguientes datos.

```
<HTML>
<HEAD><TITLE>Resultados de Encuéntreme</TITLE></HEAD>
<BODY>
Resultados de la consulta:<P>
<%begindetail%>
<%au_fname%>, <%aun_lname%>, <%phone%>, <%address%>, <%city%>,
  <%state%>, <%zip%><BR>
<%enddetail%>
</BODY>
</HTML>
```

5. Utilizando un explorador Web, abra el documento findme.htm utilizando el protocolo HTTP, como en *http://miservidor/findme.htm*. Introduzca el nombre de la persona que desea encontrar y después pulse el botón Enviar. Como la consulta SQL del archivo .IDL utiliza el carácter comodín de SQL (%) introduciendo sólo la inicial del nombre o del apellido, debe encontrar todas las personas que tengan esa inicial en la base de datos. Dejando ambos campos en blanco y pulsando el botón Enviar, devolverá todos los registros de la tabla de la base de datos.

Con optimismo, este ejercicio le habrá dado la esencia de IDC. Para más información sobre IDC, consulte la documentación de IIS. Todavía se admite IDC, pero se está suplantando por las bastante más potentes y faciles de utilizar Páginas de servidor activo discutidas en el Capítulo 14.

14 Componentes de datos ActiveX



Aunque este libro es sobre programación ActiveX en Microsoft Visual Basic—especialmente en lo relativo a Internet—no podemos eludir la discusión de aspectos de bases de datos, ya que existen muy pocos sistemas en Internet que no requierean acceso a bases de datos. En este capítulo, exploraremos los aspectos de acceso a bases de datos de Visual Basic, desde sus humildes principios hasta la plétora de opciones de hoy en día. De forma específica, examinaremos las características de acceso a datos de los siguientes componentes ActiveX y de su utilización desde Visual Basic: Objetos de acceso de datos, Conectividad abierta de bases de datos (ODBC), Objetos de datos remoto, OLE DB y Objetos de datos ActiveX.

El motor de bases de datos Jet

Visual Basic 3.0 llegó al mercado en 1993 con un control de datos incorporado (el cual llamaremos control de datos Jet, para diferenciarlo del control de datos Remote). Visual Basic 3.0 también proporcionaba otros controles relativos a datos que el desarrollador podía enlazar a un control de datos Jet y a una interfaz de programación orientada a objetos del motor de la base de datos de Jet 1.1, llamado Objetos de acceso de datos. Utilizando el motor Jet, las aplicaciones Visual Basic pueden acceder a tres categorías de orígenes de datos: archivos de Microsoft Access .MDB; archivos creados en programas de escritorio de bases de datos de terceros, incluyendo dBASE, Btrieve, Paradox y Microsoft FoxPro; y orígenes de datos ODBC.

El formato nativo de archivos de bases de datos de Jet, es el formato de archivo de Micosoft Access .MDB, aunque esto no debe sorprendernos, ya que Access utiliza el motor de Jet. Jet también admite una variedad de archivos de bases de datos externas de escritorio. Jet puede importar los datos externos o puede trabajar con los datos en el formato externo. Jet también admite orígenes de datos ODBC, que normalmente consisten en el respaldo de un servidor SQL (como Microsoft SQL Server, Sybase System 11 u Oracle 7 Server) o cualquier otro almacenamiento de datos que se encuentre ubicado en cualquiera de estos orígenes de datos, también puede conectarse a los tres tipos de forma concurrente. Utilizando el motor de Jet, los desarrolladores bastante experimentados escriben programas que van a buscar y unen datos heterogéneos de uno o más orígenes de datos ODBC y archivos de bases de datos de escritorio.

Las aplicaciones de bases de datos en Visual Basic suelen tener tres partes, tal y como muestra la Figura 14-1. El componente de interfaz de usuario consiste en los verdaderos formularios, controles y código asociado a la aplicación. Esto se lleva a cabo interactuando con el motor de la base de datos Jet, la cual turna la realización de las operaciones solicitadas en el almacén de datos y devuelve el resultado a la aplicación. El almacén de datos consiste en uno o varios archivos, que contienen los datos reales.

La Figura 14-1 muestra la arquitectura de una aplicación personal de base de datos de único usuario. La Figura 14-2 muestra la configuración de un sistema de base de datos remota multiusuario. Aquí, los clientes ejecutan la interfaz (la aplicación) que llama el motor de Jet.

Es un sistema multiusuario, pero todavía dista mucho de la verdadera configuración cliente/servidor. Todo cliente tiene su propia copia del motor de Jet y el servidor no se utiliza para nada más que para compartir espacio en disco.

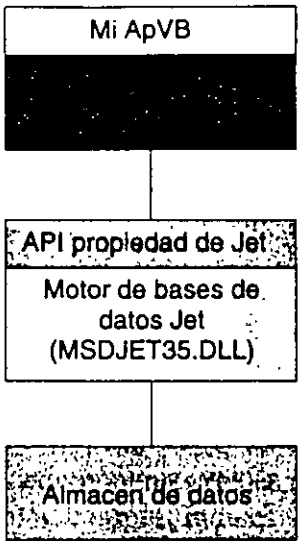


Figura 14-1. *Aplicación de base de datos de usuario único.*

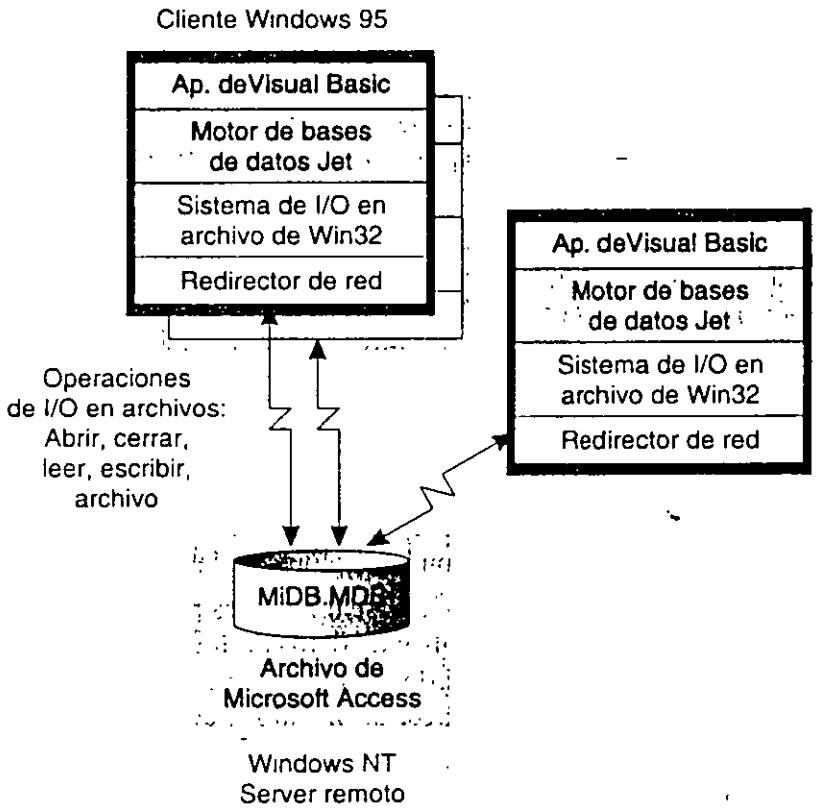


Figura 14-2. *Aplicación de base de datos multiusuario.*

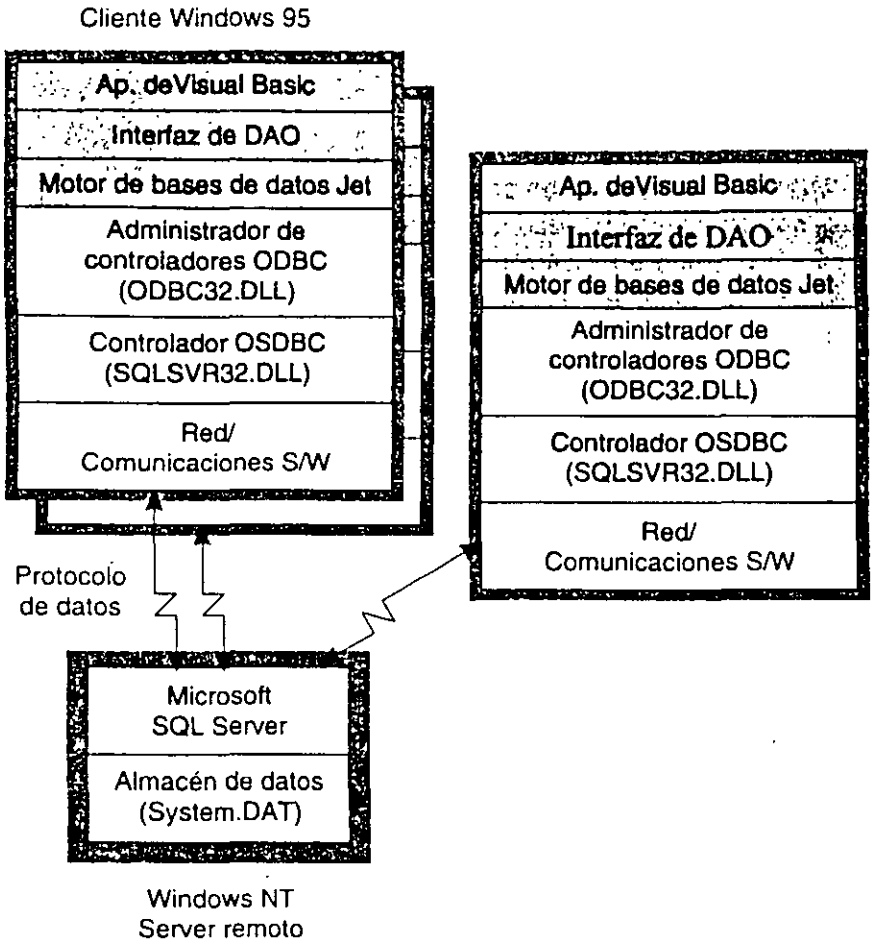


Figura 14-3. *Acceso a SQL Server por medio del motor de la base de datos Jet.*

¿Qué pasa si se desea un verdadero sistema cliente/servidor? Bueno, ya casi estamos —exceptuando un pequeño problema—: Jet no es un verdadero motor de bases de datos cliente/servidor. Afortunadamente, Jet puede proporcionar acceso a un servidor de bases de datos con características plenas, como Microsoft SQL Server por medio de ODBC, tal y como muestra la Figura 14-3. Aquí, el motor de Jet no actualiza directamente el almacén físico de datos. Esto conecta con el origen de datos ODBC, optimiza las consultas y actúa como conducto general de datos entre la aplicación y el verdadero motor de la base de datos (como Microsoft SQL Server). Esto permite a un motor de base de datos, controlar el acceso a los datos, en oposición a que cada motor de Jet del cliente tenga que intentar cooperar con los motores de Jet de otros clientes.

La forma más simple de acceder a los datos desde Visual Basic es por medio del control de datos de Jet. Las aplicaciones que utilizan el control de datos de Jet pueden permitir que el usuario vea y actualice los registros de datos desde cualquier origen de datos admitido, sin escribir ni una línea de código. Esto se puede hacer utilizando el control de datos de Jet en combinación con los controles de datos enlazados. Se puede determinar si el control es aplicable a los datos (si se puede enlazar a un control de datos de Jet) verificando si tiene las propiedades *DataSource* y *DataField*. Mediante la asignación de la propiedad *DataSource* (a la propiedad *Name* del control de datos) y la propiedad *DataField* (el campo de la pro-

piedad *RecordSource* del control de datos) del control aplicable a datos, se puede enlazar ese control a un determinado campo de la tabla. El control (por ejemplo, un cuadro de texto) mostrará los datos del campo especificado del registro actual cuando el usuario recorra la tabla. También actualizará el registro para reflejar cualquier cambio en los datos. (Los controles de datos utilizan el concepto de registro actual —en cualquier momento dado, los controles enlazados están mostrando los datos que referencia el registro actual del control de datos de Jet.) Aunque hay aspectos potentes, carecen de cierta funcionalidad, como la de métodos para borrar registros, procesar transacciones o crear nuevas tablas.

Objetos de acceso a datos

Para suplir las necesidades de una aplicación de base de datos algo más sofisticada, el motor de Jet expone una interfaz de programación orientada a objetos llamada Objetos de acceso a datos (DAO). DAO es una interfaz de programación para el motor de Jet que por tanto proporciona acceso pleno a las características del motor de la base de datos Jet, al contrario que el control de datos de Jet.

Como se puede observar en la Figura 14-4, DAO encierra la estructura y jerarquía de los componentes reales de la base de datos —todo, desde la misma base de datos hasta las tablas, campos, índices y relaciones, contenidos en ella—. La belleza de DAO es que se trabaja con objetos que hacen referencia a los componentes de una base de datos virtual. Las acciones sobre esos objetos ocasionan que el motor de Jet manipule el almacén físico de datos. La base de datos física se puede almacenar en un archivo .MDB en la máquina local o en un servidor de Oracle que ejecute VAX; el código escrito es idéntico.

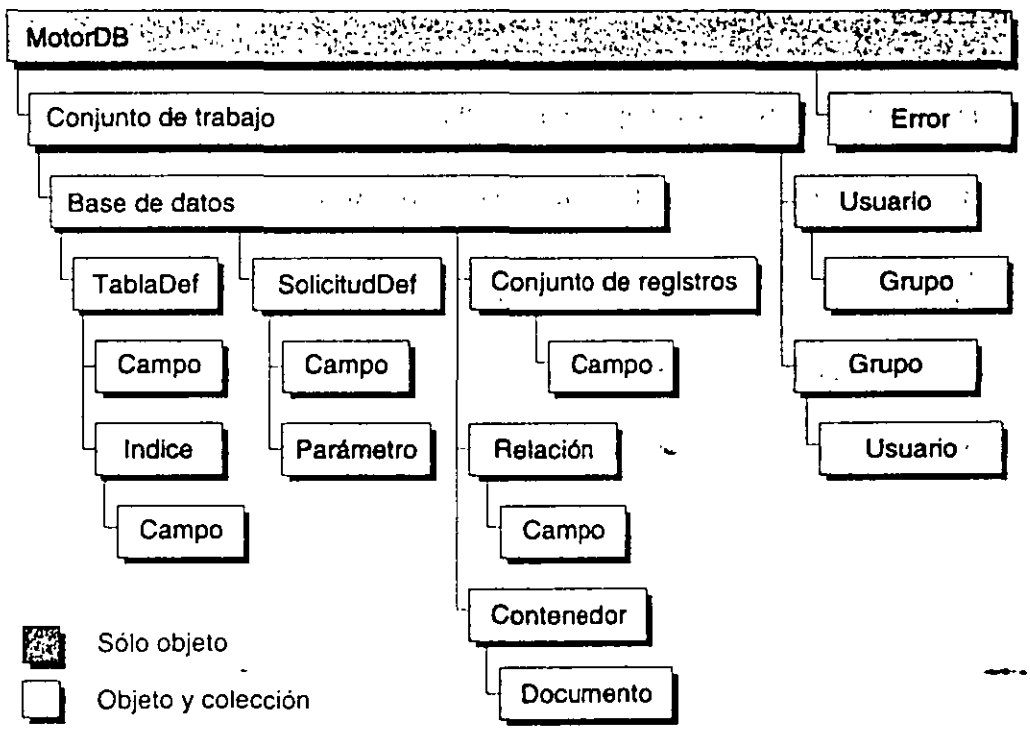


Figura 14-4. *Jerarquía de DAO.*

Nota: *Para utilizar DAO desde Visual Basic, es necesario asignar una referencia al componente DAO ActiveX. Elija la orden Proyecto, Referencias y verifique Microsoft DAO 3.5 Objetc Library.*

Las características de DAO se pueden dividir en dos categorías principales: aquellas diseñadas para definir el esquema de la base de datos y aquellas diseñadas para trabajar con datos de una base de datos existente. Al primero a veces se le denomina Lenguaje de definición de datos (DDL) y al último se le denomina Lenguaje de manipulación de datos (DML). No hay lenguajes diferentes, sino que DAO admite operaciones diferentes.

En Visual Basic, el Objeto *DBEngine* representa el mismo motor de la base de datos Jet. *DBEngine* contiene dentro de él una colección de áreas de trabajo —la primera de las cuales, *Workspaces(0)*, se crea automáticamente—. Un objeto *Workspace* es una sesión en las que se puede crear o abrir un objeto de la base de datos. El siguiente ejemplo utiliza las características DDL de DAO para crear un archivo de base de datos .MDB:

```
Dim MyDB As Database, MyWs As Workspace, MyTable As TableDef
Dim MyField As Field
Set MyWs = DBEngine.Workspaces(0)
Set MyDB = MyWs.CreateDatabase("MYDBNAME.MDB",¬
  dbLangGeneral)
SetMyTable = MyDB.CreateTableDef("MyTablaName")
SetMyField = MyTable.CreateField("MyFieldName",dbText)
MyField.Size = 100
MyTable.Fields.Append MyField
MyDB.TableDefs.Append MyTable
MyDb.Close
```

Encontrará el programa ejemplo Createdb.vbp en el disco que acompaña a este libro, dentro de la carpeta \BK-SAMP\Chap14\Createdb.

Escribir código que utilice Objetos de acceso a datos, no es la única forma de crear una base de datos .MDB. Se puede, por ejemplo, utilizar Microsoft Access o el Administrador visual de datos (una herramienta de administración y definición de bases de datos que viene con Visual Basic) para crear una base de datos de forma interactiva. Una vez definida, la base de datos se puede abrir y manipular, utilizando las características DML de DAO. El siguiente ejemplo utiliza el método *OpenDatabase;* nótese que los objetos definidos dentro de la base de datos (tablas, campos y demás) están disponibles de inmediato, una vez que se abra la base de datos:

```
Dim MyDB As Database, MyWs As Workspace, MyTabla As TableDef
Dim MyField As Field
Set MyWs = DBEngine.Workspaces(0)
Set MyDB = MyWs.OpenDatabase_
 ("C:\Archivos de programa\DevStudio\VB\BIBLIO.MDB)
SetMyTable = MyDB.TableDefs("Auhors")
SetMyField = MyTable.Fields("Author")
```

Cuando se trabaja con «objetos» de bases de datos (*Database, Workspace, Recordset* y demás), en realidad se trabaja con variables que hacen referencia a los objetos subyacentes (es decir, referencias). Éste es el porqué de la necesidad de la palabra *Set;* es decir, «se asig-

na esta variable para referenciar un objeto». Una vez abierta la tabla de la base de datos, se puede abrir un *RecordSet* y mostrar los registros:

```
DimMyRs As Recordset
    Dim Count As Integer

    Set MyRs = MyTable.OpenRecordset

    For Count = 1 To 10
       Print MyRs.Fields(0), MyRs,Fields(1),_
        MyRes.Fields(2)
       MyRs.MoveNext
    Next Count
    MyDB.Close
```

Encontrará el programa ejemplo Opendb.vbp en el disco que acompaña a este libro, dentro de la carpeta \BK-SAMP\Chap14\Opendb.

Se puede añadir o eliminar un registro como sigue:

```
MyRs.AddNew
MyRs![Author] = "¡Datos en el registro!"
MyRs.Update
```

También se puede utilizar DAO para consultar orígenes de datos ODBC, tal y como se muestra aquí:

```
Set MyDB = MyWs.OpenDatabase("", dbDriverNoPrompt,_
  False, "ODBC;DSN=MIDSN;UID=MILDdeUsuario;PWD=MiContraseña;")
Set MyRs = MyDB.OpenRecordset("Select* from authors")

Do Until MyRs.EOF
    Print MyRs.Fields(0), MyRs.Fields(2), MyRs.Fields(2)
    MyRs.MoveNext
Loop
```

DAO y el control de datos de Jet no son interfaces mutuamente excluyentes. De hecho, a menudo se utilizan juntas. Aunque DAO es una interfaz de programación potente, no proporciona características para el enlace de controles con los campos de la tabla de una base de datos. En contraste, el control de datos de Jet, combina unas escasas características de programación, con un modelo de control de datos enlazados potente. DAO y el control de datos de Jet se complementan muy bien. La propiedad *Database* del control de datos de Jet expone la utilización subyacente del objeto *Database*. Con esta y otras propiedades del control de datos de Jet, se puede manipular la base de datos utilizando DAO. La utilización del control de datos de Jet junto con DAO puede ser bastante práctica, particularmente cuando se accede a datos de un archivo .MDB o a otro tipo de archivo admitido de base de datos de escritorio. Se puede obtener una programación de DAO con características de alto nivel del control de datos de Jet.

Sin embargo, DAO y el motor de Jet tienen sus limitaciones. Hay que pagar un precio por la interfaz de alto nivel que marca la diferencia entre los archivos de bases de datos loca-

les y las conexiones a servidores SQL remotos. En este caso, la desventaja es la disminución de rendimiento cuando se conecta al origen de datos ODBC, debida a la capa extra del motor de Jet. Como la mayoría de las aplicaciones cliente/servidor escritas en Visual Basic, terminan utilizando una conexión ODBC a un servidor remoto de bases de datos, esta disminución del rendimiento en los orígenes de datos ODBC por medio del motor de Jet, es un gran problema. Muchos desarrolladores han recurrido a aprender cómo utilizar directamente la API de ODBC desde Visual Basic, para obtener un aumento del rendimiento. Utilizando esta solución, los desarrolladores eluden el motor de Jet y controlan los datos por completo.

El modo *ODBCDirect*

Para resolver algunos aspectos de rendimiento, Microsoft ha añadido recientemente a DAO el soporte para el modo *ODBCDirect*. *ODBCDirect* es un modo alternativo de DAO que permite acceder directamente a orígenes de datos ODBC, eludiendo el motor de Jet y aprovechando las ventajas de la capacidad de procesamiento de orígenes remotos de datos. *ODBCDirect* es el objetivo de los desarrolladores que utilizan DAO junto a un origen de datos ODBC y que pretenden mejorar el rendimiento de sus aplicaciones. No tiene la intención de reemplazar componentes de desarrollo de empresa como los Objetos de datos Remote, discutidos más tarde en este capítulo. *ODBDirect* en realidad se implementa utilizando la API de ODBC.

Para especificar *ODBCDirect*, asigne *dbUseODBC* a la propiedad *DBEngine.DefaultType*. Las solicitudes sucesivas ejecutadas dentro del área de trabajo, no caerán en la sobrecarga del motor de Jet:

```
Dim MyDB As Database, MyWs As Workspace, MyRs As Recordset
DBEngine.DefaultType.dbUseODBC
Set MyWs = DBEngine.Workspaces(0)

Set MyDB = MyWs.OpenDatabase("", dbDriverNoPrompt, ¬
  False, "ODBC;DSN=MiDSN;UID=MiIDdeUsuario; ¬
  PWD=MiContraseña;")
 SetMyRs = MyDB.OpenRecordset("Selección * por autores")

Do Until MyRs.EOF
    Print MyRs.Fields(0), MyRs.Fields(2), MyRs.Fields(2)
    MyRs.MoveNext
 Loop
```

Cómo entender ODBC

La API de ODBC es un conjunto de funciones que definen una interfaz para la consulta de servidores de bases de datos utilizando SQL y se diseñó para utilizarse con C y C++. La mayoría de los servidores SQL proporcionan una API con la que se puede conectar al servidor y pasar las sentencias SQL para su procesamiento, pero este proceso restringe la utilización del servidor de bases de datos del vendedor. (Por cierto, los vendedores no parece que lo piensen.) Lo bonito de ODBC es que más que tener que aprender una API propiedad del vendedor de la base de datos, se puede utilizar la API de ODBC para acceder a cualquier origen de datos para

la que esté disponible un controlador ODBC. Uno de los principales objetivos de diseño de ODBC es proporcionar un rendimiento equivalente al del la API nativa de DBMS. Esto significa que si el lunes por la mañana el jefe decide cambiar por completo el almacenamiento los datos del departamento, de Oracle a SQL Server, las aplicaciones que utilicen la API de ODBC, virtualmente no necesitaran modificaciones. No obstante, para utilizar ODBC, se tiene que conocer SQL y la API de ODBC. La arquitectura ODBC se puede dividir en cuatro componentes: aplicaciones, administrador de controladores, controladores y orígenes de datos.

En este modelo, la aplicación es un programa Visual Basic. Basada en las acciones del usuario, realiza solicitudes al administrador de controladores, por medio de la API de ODBC. El administrador de controladores (ODBC32.DLL) es un intermediario entre la aplicación y el verdadero controlador ODBC. El propósito del administrador de controladores es triple: carga y descarga los controladores ODBC, refleja las llamadas de la API de ODBC en los controladores ODBC apropiados y permite que el usuario final cree un Nombre de origen de datos (DSN).

Veamos primero el DSN como el vehículo principal de conexión en ODBC. Un DSN en realidad es un alias de un origen de datos ODBC. La idea es que los usuarios no tienen que conocer el nombre y la ubicación de la base de datos, para conectarse a un origen de datos ODBC. En cambio, toda esta información se reunirá y almacenará en un único lugar (el registro de Microsoft Windows) y después se hará referencia a posteriori por medio de un nombre dado de usuario —el DSN—. El usuario o el administrador puede crear DSN por medio del icono ODBC del Panel de control de Windows.

Tipos de DSN

Existen tres tipos de DSN: de usuario, de sistema y de archivo. Los DSN de usuario, el tipo estándar, sólo están disponibles para los usuarios que los crean y sólo en esa máquina. Los DSN de sistema están disponibles para cualquiera que utilice esa máquina, incluyendo los servicios de Microsoft Windows NT. Debido a que Internet Information Server (IIS) está implementado como un conjunto de servicios de Windows NT, los orígenes de datos a utilizar con IIS, es necesario que sean de este tipo —de sistema—. Normalmente, la información de un DSN se almacena en el registro de Windows. El DSN de archivo almacena esta información en un archivo en el disco. Esto significa que los usuarios de las máquinas pueden compartir estos orígenes de datos a través de la red, suponiendo que tengan instalados los controladores ODBC necesarios.

Sin el administrador de controladores, las aplicaciones tendrían que cargar y descargar los controladores apropiados, recolectar la información de la conexión y demás. Además, también se puede utilizar el administrador de controladores para cargar varios controladores ODBC de una vez. Por ejemplo, varias aplicaciones pueden acceder de forma simultánea a orígenes de datos que requieren controladores diferentes o se pueden consultar datos de dos servidores que requieren controladores diferentes. Ésta es una de las principales razones para la utilización del administrador de controladores: el administrador de controladores puede administrar conexiones múltiples concurrentes a múltiples controladores ODBC. Un controlador ODBC es el responsable de:

- Implementar las funciones de la API de ODBC.
- Establecer una conexión y remitir las solicitudes al origen de datos.
- Devolver los resultados a la aplicación.
- Traducir los errores a códigos de error estándar.

- Declarar los cursores de manipulación.
- Administrar las transacciones.

El objetivo es aislar las aplicaciones de la API propiedad de los vendedores de bases de datos. Como mencionamos anteriormente, las aplicaciones se comunican a través de la API de ODBC, el administrador de controladores interpreta estas llamadas y carga el controlador ODBC necesario y el controlador interactúa con el origen de datos. La aplicación puede acceder a cualquier origen de datos por medio de una única API.

Aunque ODBC se diseñó para ajustar al entorno cliente/servidor, es lo bastante flexible como para tratar con DBMS no cliente/servidor. Por ejemplo, los controladores ODBC están disponibles para la interacción con archivos .MDB, otros archivos de bases de datos de escritorio, archivos de texto e incluso con archivos de Microsoft Excel (.XLS). Para tratar con orígenes de datos cliente/servidor y no cliente/servidor, ODBC define dos tipos diferentes de controladores: de un piso y de dos pisos.

Un controlador ODBC de un piso accede a archivos de una base de datos de escritorio como .MDB o a un sencillo archivo de texto. La configuración habitual cubre tanto el sistema que tiene la base de datos alojada en la misma máquina —o piso— como en el controlador. Los controladores de un piso suelen tener una gran cantidad de huellas en la memoria, ya que son un controlador y un DMS fundido en uno. Por ejemplo, el controlador ODBC de .MDB contiene una versión especial del motor de Jet. Como el motor de Jet ya proporciona acceso a la mayoría de las bases de datos de escritorio de un PC estándar, los controladores de un piso no son del todo útiles desde Visual Basic. Es más común un controlador de dos pisos, el cual se ajusta al molde clásico de cliente/servidor. El controlador (cliente) envía las solicitudes SQL y recibe los resultados del DBMS (servidor), normalmente a través de la red.

La ODBC estándar define tres niveles de conformidad del controlador: núcleo, nivel 1 y nivel 2. Las funciones del núcleo corresponden con las funciones de X/Open y de la especificación Grupo de acceso SQL (SAG), en las que se basa OBDC. Los niveles 1 y 2 son dos conjuntos de funcionalidad extendida definidos por Microsoft. Éstos proporcionan aspectos como los cursores desplazables y el procesamiento asíncromo. Cada función de la API de ODBC está encuadrada en uno de estos tres niveles. La siguiente tabla muestra una lista de las funciones del núcleo de ODBC y su utilización.

Funciones del núcleo de ODBC

Tarea	Nombre de la función	Propósito
Conexión a un origen de datos	SQLAllocEnv	Obtiene un descriptor de entorno . Los descriptores de entorno se utilizan en una o más conexiones.
	SQLAllocConnect	Obtiene un descriptor de conexión.
	SQLConnect	Conecta con un determinado controlador mediante su nombre de origen de datos, identificador de usuario y contraseña.
Preparación de solicitudes SQL	SQLAllocStmt	Asigna un descriptor de sentencia.

(continuación)

Tarea	Nombre de la función	Propósito
	SQLPrepare	Prepara una sentencia SQL para su posterior ejecución.
	SQLGetCursorName	Devuelve el nombre asociado a un descriptor de sentencia.
	SQLSetCursorName	Especifica un nombre de cursos.
Envío de solicitudes	SQLExecute	Ejecuta una sentencia preparada.
	SQLExecDirect	Ejecuta una sentencia.
Obtención de resultados e información sobre los resultados	SQLRowCount	Devuelve el número de filas afectadas por una solicitud de inserción, actualización o eliminación.
	SQLNumResultCols	Devuelve el número de columnas del conjunto de resultados.
	SQLDescribeCols	Describe una columna del conjunto de resultados.
	SQLColAttributes	Describe los atributos de una columna del conjunto de resultados.
	SQLBindCol	Asigna el almacén para una columna de resultados y especifica el tipo de datos.
	SQLFetch	Devuelve múltiples filas de resultados.
Finalización de una sentencia	SQLFreeStmt	Finaliza el procesamiento de una sentencia.
	SQLCancel	Cancela una sentencia SQL.
	SQLTransact	Termina o repliega una transacción.
Finalización de una conexión	SQLDiconnect	Cierra la transacción.
	SQLFreeConnect	Inicia el descriptor de transacción.
	SQLFreeEnv	Inicia el descriptor del entorno.

Objetos de datos Remote

Al evolucionar Visual Basic, Microsoft se dio cuenta de que el motor de Jet estaba demasiado limitado para la mayoría de los sistemas cliente/servidor. Aunque el motor de Jet es una tecnología potente y de características ricas, intenta abstraer la programación de bases de datos, hasta el punto en que enmascara las diferencias entre los archivos de bases de datos de escritorio y los servidores SQL remotos y estos impone una penalización de rendimiento demasiado grande. Microsoft también se dio cuenta de que la API de ODBC simplemente es demasiado difícil y voluminosa para la mayoría de los desarrolladores de Visual Basic. Por lo que Microsoft desarrolló los Objetos de datos Remote (RDO). Al igual que los Objetos de acceso a datos del motor de Jet, RDO es una interfaz de programación orientada a objetos-de bases de datos. A diferencia de Jet, RDO no es un motor de base de datos, pero si una fina capa de código implementada por encima de ODBC. En otras palabras, RDO es la interfaz de Visual Basic para la API de ODBC y como la API de ODBC, utiliza sentencias SQL para acceder y consultar los orígenes de datos ODBC. Aunque RDO puede acceder a cualquier origen de datos ODBC (archivos de bases de datos de escritorio,

ervidores finales SQL y demás), se ha optimizado para provechar los sofisticados motores de consulta en productos como Microsoft SQL Server y Oracle. Hay cinco ventajas principales de utilizar RDO, cuando se compara con la API de ODBC.

- No es necesario aprender cómo utilizar la API de ODBC desde Visual Basic.
- Rendimientos aproximados a los de la API de ODBC.
- RDO es una interfaz de programación nativa de Visual Basic.
- Los modelos del control de datos remotos y del control de datos enlazados, están disponibles.
- La API de ODBC se puede utilizar junto con RDO.

La Figura 14-5 muestra el modelo de objeto RDO y revela las relaciones entre RDO y la API de ODBC. Los tres tipos de descriptores de la API de ODBC (descriptores de entorno, de conexión y de sentencia) tienen sus correspondientes tipos de objeto en RDO *(rdoEnvironment, rdoConnection y rdoResultset).* Aunque es una simplificación excesiva, se puede decir que RDO es la API de ODBC, lo que la Biblioteca Microsoft de creación de clases (MFC) es a la API de Windows. Nótese que la terminología de bases de datos de RDO y ODBC es algo diferente a la de Jet; en RDO, a los campos se les llama columnas y a los registros se les llama filas.

Nota: *Para utilizar RDO desde Visual Basic, es necesario asignar una referencia al componente RDO ActiveX. Elija la orden Proyecto, Referencias y verifique Microsoft Remote Data Object 2.0.*

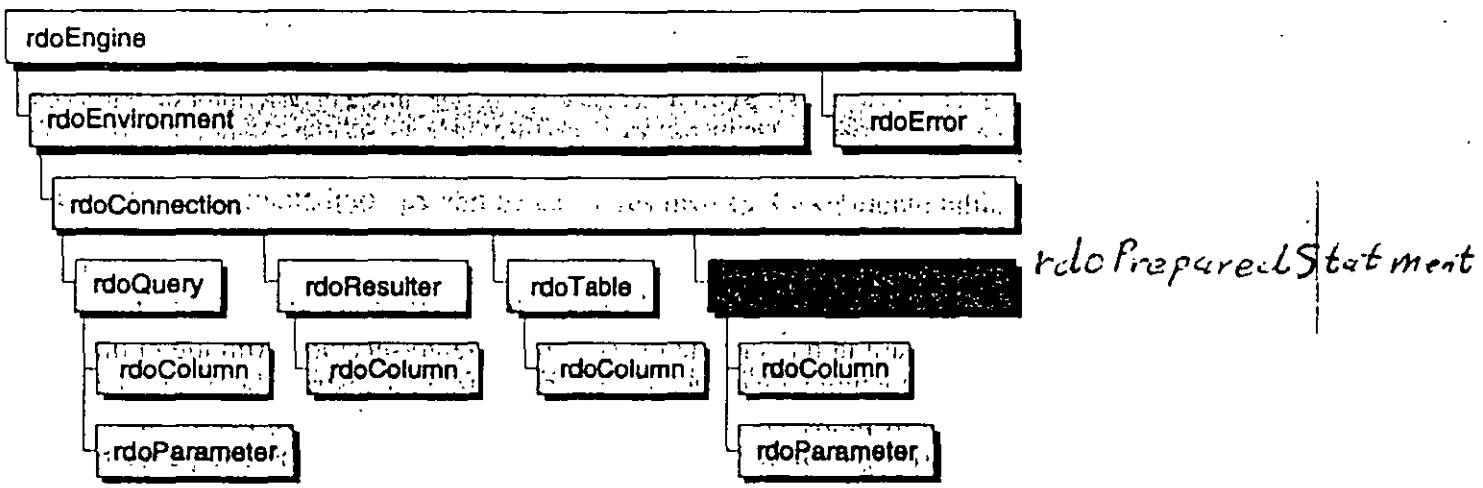


Figura 14-5. *Modelo de objeto RDO 2.0.*

Nota: *El objeto* rdoPreparedStatement *está obsoleto y sólo se admite con propósitos de compatibilidad. Se debe cambiar a la utilización del nuevo objeto* rdoQuery, *el cual admite un superconjunto de la funcionalidad del objeto* rdoPreparedStatement.

A lo mejor deberíamos distraernos por unos momentos y discutir los conceptos de contenedor y colección. El objeto *rdoEngine* se sitúa en la parte más alta del diagrama de jerarquía de objetos de la Figura 14-5. Es un objeto contenedor y contiene los objetos *rdoEnvironment y rdoError.* A los objetos del contenedor se hace referencia con la notación habitual de puntos:

```
rdoEngie.rdoEnvironment
```

El objeto *rdoEngine* contiene en realidad una colección de objetos *rdoEnvironment,* llamada *rdoEnvironments.* Una colección es un grupo denominador de componentes afines —básicamente, un array de objetos del mismo tipo—. A los miembros de una colección se hace referencia mediante la notación habitual de array.

```
rodEngine.rdoEnvironments(n)
```

No es necesario crear el objeto *rdoEngine* —está predefinido y sólo puede haber uno por aplicación—. Se puede utilizar el objeto *rdoEngine* para crear objetos de entorno (con el método *rdoCreateEnvironment*) o para registrar un origen de datos ODBC (utilizando *rdoRegisterDataSource*). El objeto *rdoEngine* crea automáticamente un entorno inicial; se puede hacer referencia al objeto contenido en *rdoEngine,* como *rdoEnvironments(0):*

```
Dim myEnv As rdoEnvironment
Set myEnv = rdoEngine.rdoEnvironments(0)
```

Se pueden crear objetos *rdoEnvironment* adicionales con el método *rdoCreateEnvironment* del objeto *rdoEngine.* Estos nuevos objetos *rdoEnvironment* después forman parte de la colección *rdoEnvironments.*

```
Dim myEnv As rdoEnvironment
Set myEnv = rdoEngine.rdoCreateEnvironment_
 ("NombredeMiEnv", "MiIDdeUsuario","MiContraseña")
```

Los entornos también pueden determinar el alcance de las transacciones. Se utilizan los métodos *BeginsTrans, CommitTrans y Rollbacktrans* del objeto *rdoEnvironment,* para controlar las transacciones dentro de ese entorno. (Se ven afectadas todas las bases de datos *rdoConnection* abiertas del entorno.) Cuando se tiene un objeto *rdoEnvironment* válido, el siguiente paso es abrir una conexión con un origen de datos ODBC, utilizando el método *OpenConnection;* éste establece un enlace físico con el origen de datos. El método *OpenConnection* acepta los parámetros mostrados en la siguiente tabla.

Parámetros del método OpenConnection

Parámetro	Significado
DsName	Nombre del origen de datos ODBC registrado.
Prompt	Este indicador especifica si el administrador de controladores ODBC debe mostrar un cuadro de diálogo indicando al usuario información sobre la conexión.
Readonly	Este indicador booleano, si se le asigna False, especifica que se desea actualizar datos a través de esta conexión.
Connect	Cadena que proporciona los argumentos de la conexión al administrador de controladores ODBC. (Los distintos orígenes de datos ODBC, requieren diferentes parámetros.)

El siguiente código crea un objeto *rdoConnection* y llama a su método *EstablishConnection* para conectar con el servidor. Después se ejecuta una consulta SQL, utilizando el método *OpenResultset* y se muestran los resultados devueltos:

```
Dim myCont As rdoConnection

Dim myRes As rdoResultset

myCon.Connect = "DSN=MiDSN;UID=MiIDdeUsuario;PWD=MiContraseña;"
myCon.EstablishConnection rdoDriverNoPrompt, False
Set myRes = myCon.OpenResultset("Select au_lname ¬
  from authors")

Do Until myRes.EOF
    Print myRes.rdoColumns(0)
    myRes.MoveNext
Loop
```

Para solicitar y obtener filas de un origen de datos, es necesario crear un objeto *rdoResultset* llamando al método *OpenResultset*. Los dos objetos RDO que admiten este método son *rdoConnection y rdoQuery*. La utilización del objeto *rdoConnection* es una buena idea si se pretende una consulta ad hoc de una vez que no se desea ejecutar de neuvo (por ejemplo, una consulta introducida por el usuario). La utilización del objeto *rdoQuery* se recomienda cuando se está creando una consulta que se pretende ejecutar múltiples veces. El objeto *rdoQuery* permite incluso crear consultas con parámetros que se sustituyen en la consulta antes de que se ejecute, como muestra el siguiente código (en breve, más sobre esto):

```
Dim myCont As rdoConnection
Dim myQry As rdoQuery
Dim myRes As rdoResultset

myCon.Connect = "DSN=MiDSN;UID=MiIDdeUsuario;PWD;MiContraseña;"
myCon.EstablishConnection rdoDrivernoPrompt, False

Set myQry = myCon.CreateQuery("Mi primera consulta", ¬
  Seleccione aulname, au_fname de los autores")
Set myQry = myQry.OpenResultset

Do Until myRes.EOF
    Print myRes.rdoColumns(0), myRes.rdoColumns(1)
    myRes.MoveNext
 Loop
```

Cómo definir cursores RDO

Un cursor es un conjunto lógico de registros administrado por un origen de datos o por un administrador de controladores ODBC, en una máquina cliente. El tipo de cursor *rdoResultset* creado en el código anterior es el tipo predeterminado, sólo adelante. Los otros tipos

disponibles (dependiendo de la capacidad del controlador ODBC) son estáticos , conjunto de claves y dinámico. El conjunto de resultados sólo adelante, obtiene rápidamente los datos y con sobrecarga mínima. Este tipo de conjunto de resultados expone un único registro cada vez y sólo se puede avanzar en el conjunto de resultados. Otro problema es que las filas del conjunto de resultados no se pueden actualizar si se cambian en el origen de datos. No obstante, puede que sea más eficiente crear de nuevo un conjunto de resultados sólo adelante, que utilizar otro de los tipos. La principal diferencia entre un conjunto de resultados sólo adelante y un cursor estático (parecido al objeto *Recordset* del tipo Snapshot de Jet) es que es posible moverse hacia delante y hacia atrás en el conjunto de resultados. Un *rdoResultset* del tipo conjunto de claves (parecido al objeto *Recordset* del tipo Dinaset de Jet) es desplazable y las filas reflejan los cambios realizados tras crear el conjunto de resultados. Se establece un conjunto de claves cuando el cursor obtiene claves únicas para los registros del origen de datos, en lugar de para los propios datos. A estos identificadores de registro se les denomina claves y a la tabla de claves se la denomina conjunto de claves. Cuando se solicita un determinado registro, el cursor utiliza la clave del registro para obtener el contenido completo del registro.

Un *rdoResultset* de tipo dinámico es idéntico al del tipo conjunto de claves, con la excepción de que los miembros no son estáticos; esto significa que RDO comprueba constantemente si registros nuevos o actualizados del origen de datos, cumplen las restricciones de la consulta. Debido a esta comprobación, este tipo de cursor conlleva la mayor sobrecarga. La siguiente tabla resume los cuatro tipos de cursor *rdoResultset*.

Atributo del cursor *rdoResultset*

Atributo	Sólo adelante	Estático	Conjunto de claves	Dinámico
Actualizable	No*	Sí	Sí	Sí
Número de miembros	Fijo	Fijo	Fijo	Dinámico
Visibilidad	Una fila	Cursor	Cursor	Cursor
Movimiento de la fila actual	Adelante	Cualquiera	Cualquiera	Cualquiera
Resultado de una unión	Sí	Sí	Sí	Sí

* Sí, cuando se utilizan cursores del lado del servidor en Microsoft SQL Server 6.0 o posterior.

Algunos orígenes de datos admiten cursores del lado del servidor (por ejemplo, Microfoft SQL Server). Cuando se da este caso, el motor de la base de datos remota crea conjuntos de claves del cursor en el servidor. Los cursores en el lado del servidor reducen la cantidad de memoria y el espacio en disco necesarios en la máquina cliente, pero los trasladan al servidor. Se puede configurar la propiedad *CursorDrive* del objeto *rdoEnvironment* para que controle si el cursor se crea y mantiene en la máquina cliente o servidor, utilizando las opciones que están en la tabla de la página siguiente.

Control de la creación de cursores con la propiedad *CursorDriver*

Opción	Descripción
rdUseClientBatch	RDO utiliza la biblioteca de cursores del lote del cliente.
rdUseIfNeeded	(Predeterminada.) El controlador ODBC elige automáticamente el estilo apropiado. Se utilizan los cursores del lado del servidor, si están disponibles.
rdUseNone	RDO crea un conjunto de resultados sin cursores.
rdUseODBC	RDO utiliza la biblioteca de cursores de ODBC, la cual crea los cursores en la computadora cliente. Esta opción da mejores rendimientos en conjuntos pequeños de resultados, pero el rendimiento se degrada rápidamente al aumentar el conjunto de resultados.
rdUseServer	Siempre se utilizan los cursores del lado del cliente, si están disponibles. Esta opción suele dar mejores rendimientos, pero puede ocasionar un mayor tráfico en la red.

Cómo recorrer y editar filas con RDO

En cualquier momento, sólo se puede exponer un cursor para obtener o modificar datos —la fila a la que hace referencia el actual puntero de fila—. Se puede cambiar la fila actual utilizando métodos como *Move, MoveNext, MovePrevious, MoveFirst* o *MoveLast* o asignando las propiedades *AbsolutePosition* o *PercentPosition* del objeto *rdoResultset*. También se pueden modificar los datos de la fila actual utilizando el método *Edit*, suponiendo que se ha asignado *True* a la propiedad *Updatable* de *rdoResultset*. Asigne los valores de la columna como desee y utilice el método *Update* para guardar de nuevo la columna en el origen de datos. Si se cambia de idea, se puede utilizar *CancelUpdate* en lugar del método *Update* para descartar los cambios. También se pueden añadir o eliminar filas utilizando los métodos *AddNew* y *Delete*.

```
Do Until myRes.EOF
    If myRes.Columns("au_lname").Value =  "White" Then
        myRes.Edit
        myRes.rdoColumns("au_lname").Value = "Eddon"
        myRes.Update
    End If
    myRes.MoveNext
 Loop
```

En lugar de utilizar los métodos *Edit, AddNew, Delete* y *Update*, se puede utilizar el método *Execute* para realizar una solicitud SQL que contenga una o más sentencias *UPDATE, INSERT o DELETE*. Por ejemplo, el código anterior se podría escribir de forma más eficiente como solicitud de actualización SQL.

```
myCon.Execute"Update Autores Set au_lname = "Eddon"Where ↵
 au_lname = 'White'"
```

El método *Execute* está diseñado para ejecutar peticiones de acción que devuelvan filas. También se puede controlar cómo un objeto *rdoResultet* creado mediante el método *Open-*

Resultset, administra la concurrencia, asignándole al argumento *LockType* alguno de los indicadores mostrados en la siguiente tabla:

Indicador de argumentos de *LockType*

Indicadores de *LockType*	Descripción
rdoConcurBatch	La biblioteca de cursores del lote del cliente se está utilizando y se desea aplazar todas las actualizaciones hasta que se utilice el método *BatchUpdate*.
rdoConcurLock	Bloqueo pesimista. La fila o filas que se están actualizando se bloquean tan pronto como se ejecutan los métodos *Edit* o *AddNew* y se mantiene el bloqueo hasta que se llama al método *Update* y se haya actualizado el origen de datos.
rdoConcurReadOnly	El cursor es de sólo lectura. Éste es el valor por defecto.
rdoConcurRowver	Bloqueo optimista. Sólo se bloquean las filas que de verdad se van a actualizar, basándose en las versiones de la fila.
rdoConcurValues	Bloqueo optimista. Sólo se bloquean las filas que de verdad se van a actualizar, basándose en los valores de la fila.

Cómo procesar conjuntos múltiples de resultados

Utilizando RDO, se pueden crear consultas que cuando se ejecuten, devolverán conjuntos múltiples de resultados. Cuando se están desarrollando sistemas cliente/servidor que se van a ejecutar a través de una red muy amplia o de Internet, el envío de una solicitud es mucho más costoso que la llamada a un servidor para la próxima creación. En tales casos, el envío de múltiples consultas SQL a la vez, puede ser una forma mucho más eficiente de interactuar con el servidor, especialmente cuando el coste de las peticiones de red es alto. Empaquetar las peticiones dentro de una única sentencia o procedimiento almacenado y permitir que el servidor envíe resultados múltiples de una vez, también elimina la necesidad de que el cliente trate múltiples sesiones con el servidor. El siguiente ejemplo se divide en tres partes: dos sentencias *SELECT* que devuelven columnas de las tablas y una sentencia *UPDATE*, que no devuelve ninguna columna.

```
Dim myCont As rdoConnection
Dim myQry As rdoQuery
Dim myRes As rdoResultset

myCon.Connect="DSN=MiDSN;UID=MiIDdeUsuario;PWD;MiContraseña;"
myCon.EstablishConnection rdoDriverNoPrompt, False

Set myQry = myCon.CreateQuery("MiSegundaConsulta", ↵
  "Select au_lname from authors;" &_
  "Seleccione hire_date de los empleados;" &_
  "Update authors Set state='NJ' Where state='CA'")

myQry.RowsetSize = 1
Set myRes = myQry.OpenResultset
```

El código anterior crea la consulta y después la ejecuta llamando al método *OpenResultset*. Esto abre el primer conjunto de resultados correspondiente a la primera sentencia *Select*, el cual devuelve la columna *au_lname* de todas las filas de la tabla de autores. el siguente código ejecuta un bucle que muestra en un cuadro de diálogo la primera columna (columna 0) del conjunto de resultados.

```
Do Until myRes.EOF
    Print myRes.rdoColumns(0)
    myRes.MoveNext
Loop
```

Cuando se termina de trabajar con los datos del primer conjunto de resultados, se puede activar el siguiente, mediante la ejecución del método *MoreResults* (después del cual, el primer conjunto de resultados no volverá a estar disponible). El siguiente código selecciona el próximo conjunto de resultados y muestra todas las filas.

```
If myRes.MoreResults Then
    Do Until myRes.EOF
        Print myRes.rdoColumns(0)
        myRes.MoveNext
    Loop
End If
```

Ahora ya se está preparado para procesar el último conjunto de resultados de la consulta, el cual se generó a partir de la sentencia *Update* que no devolvió filas. No obstante, se tiene que procesar el conjunto de resultados. La única información devuelta potencialmente útil de este conjunto de resultados, se encuentra disponible por medio de la propiedad *RowsAffected* del objeto *rdoResultset*, la cual se asigna al número de filas que se vieron afectadas por la solicitud.

```
If myRes.MoreResults Then
    MsgBox myQuery.RowsAffected & "filas actualizadas."
End If
```

Cómo crear consultas con parámetros

Puede que algunas veces se desee ejecutar la misma consulta en distintas ocasiones, pero con parámetros diferentes cada vez. Por ejemplo, se puede hacer una consulta a una aplicación, que devuelve los apellidos de las personas que ganan 30.000$ al año o más («Select Apellidos from Empleados Where Salario>=30000»). Sin embargo, dependiendo de la entrada del usuario, puede que se desee cambiar el criterio monetario y haciéndolo por turnos, se determina el número de miembros de las filas del conjunto de resultados. Una posibilidad es crear y ejecutar una nueva consulta con diferentes cantidades cada vez. Esto puede ser lento e ineficiente ya que el servidor SQL, necesitará recompilar la consulta cada vez que la ejecute. Es mucho más eficiente crear una consulta que especifique como valor desconocido, el criterio del salario: «Select Apellidos from Empleados Where Salario>=?». (La interrogación, por supuesto, es el símbolo de lo desconocido.) Para toda ? de la cadena SQL, se crea automáticamente un objeto *rdoParameter* y se le añade a la colección *rdoParameters*. Esto permite que se asignen valores a los parámetros mediante referencias al objeto de la colección utilizando la notación *rdoParameters(n)*. Por ejemplo, el primer pará-

metro de una cadena SQL es *rdoParameters(0)*, el segundo parámetro es *rdoParameters(1)* y así sucesivamente. Los valores desaparecidos se tienen que asignar antes de la ejecución de la consulta con el método *OpenResultset*, tal y como se muestra abajo en negrita.

```
Din myCont As rdoConnection
Din myQry As rdoQuery
Din myRes As rdoResultset

myCon.Connect="DSN=MiDSN;UID=MiIDdeUsuario;PWD=MiContraseña,"
myCon.EstablishConnection rdoDriverNoPrompt, False
Set myQry = myCon.CreateQuery("MiTerceraConsulta",-
  "Select au_lname, au_fname, state from autores" &_
  "Where state=?")
myQry.rdoParameters(0).Value = "CA"
Set my Res = myQry.OpenResultset

Do Until myRes.EOF
    Print myRes.rdoColumns(0), Print myRes.rdoColumns(1),
    myRes.MoveNext
Loop
```

Para ejecutar de nuevo la consulta con un valor diferente del parámetro, sólo habría que cambiar el valor del objeto *rdoParameter* y utilizar el método *Requery*.

```
myQry.rdoParameters(0).Value = "UT"
myRes.Requery

Do Until myRes.EOF
    Print myRes.rdoColumns(0), Print myRes.rdoColumns(1),
    myRes.MoveNext
Loop
```

Aunque la utilización del objeto *rdoParameter* es relativamente inmediata, RDO admite un truco que facilita aún más la utilización de parámetros. Una vez definida la consulta utilizando el método *CreateQuery*, el nombre de esa consulta (en este caso, *MiTerceraConsulta*) aparece automáticamente como un nuevo método del objeto *rdoConnection* en el cual se creó. Se puede utilizar un sencillo paso de argumentos al método, para asignar los valores de parámetro, como aquí se muestra.

```
myCon.MiTerceraConsulta "MI"
Set myRes = myCon.LastQueryResults

Do Until myRes.EOF
    Print myRes.rdoColumns(0), Print myRes.rdoColumns(1),
    myRes.MoveNext
Loop
```

Como no se llama al método *OpenResultset* cuando se utiliza el truco, la propiedad *rdoConnection.LastQueryResultset* se puede utilizar para obtener una referencia al *rdoResultset* devuelto por la consulta. Encontrará el programa ejemplo Rdoparam.vbp en el disco que acompaña a este libro, dentro de la carpeta \BK-SAMP\Chap14\Rdoparam.

Cómo ejecutar procedimientos almacenados

Un procedimento almacenado es un conjunto de sentencias SQL Transact que se almacenan en una base de datos de Microsoft SQL Server. Se puede pensar en los procedimientos almacenados como una parte de código que se almacena, mantiene y ejecuta bajo los auspicios de SQL Server. La principal razón para utilizar procedimientos almacenados es trasladar el código de la aplicación, desde la estación de trabajo del cliente hasta el servidor de bases de datos. En un entorno típico cliente/servidor, cada petición SQL se tiene que enviar a través de la red desde el cliente hasta el servidor, para que se ejecute. El résultado de cada sentencia se tiene que enviar desde el servidor hasta el cliente. Con los procedimientos almacenados, se minimiza la sobrecarga por que el cliente puede invocar un único procedimiento almacenado, que llama a múltiples sentencias SQL. En particular, esto minimiza el tráfico de la red y aumenta el rendimiento total de la aplicación.

Además de las mejoras de rendimiento logradas mediante la reducción del tráfico de la red, los procedimientos almacenados pueden aumentar el rendimiento debido a que se almacenan en un formato analizado. Esto elimina la sobrecarga analizada que requiere cada sentencia SQL antes de ejecutarse. Además, una vez compilados y optimizados, los procedimientos almacenados se mantienen en la caché del servidor.

Para llamar a un procedimiento almacenado desde Visual Basic utilizando RDO, cree una consulta con el formato especial *{(call nombredeprocedimiento(param1, param2,...)}*. Cuando se ejecuta una consulta que llama a un procedimiento almacenado en el servidor, puede que se desee recibir un valor devuelto por el procedimiento. Para hacerlo, asigne la propiedad *Direction* del objeto *rdoParameter*. En el siguiente código, se llama a un procedimiento almacenado, *sp_add*, el cual toma dos parámetros y devuelve su suma.

```
Dim myCont As rdoConnection
Dim myQry As rdoQuery
Dim myRes As rdoResultset
myCon.Connect = "DSN=MiDSN;UID=MiIDdeUsuario;PWD=MiContraseña:"
myCon.EstablishConnection rdoDriverNoPrompt, False

Set myQry = myCon.CreateQuery("MiCuartaConsulta",-
  "(?=call sp_add (?, ?))")

myQry.rdoParameters(0).Direction = rdoParamReturnValue
myQry.rdoParameters(1).Value="5"
myQry.rdoParameters(2).Value="3"

myQry.Execute       'Ejecuta la consulta del procedimiento almacenado

MsgBox "5 + 3 = "& myQry.rdoParameters(0).Value
```

En este código, se puede ver que hemos creado una consulta con tres parámetros. Asignamos el valor *rdParamReturnValue* a la propiedad *Direction* del primer parámetro; esto indica que devuelve un valor. La propiedad *Direction* del segundo y tercer parámetro no se asigna, ya que *rdParamInput* es el valor por defecto. En realidad, tampoco es necesario asignar el valor *rdParamReturnValue;* RDO se suele imaginar lo que se está haciendo. Después, utilizaremos el método *Execute* para ejecutar realmente esta consulta en lugar de utilizar el método *OpenResultset* como en los ejemplos anteriores, por que esta consulta de

acción no devuelve un conjunto de filas (conjunto de resultados). Después de la ejecución, la propiedad *Value* del primer parámetro contiene el valor que devuelve el procedimiento almacenado *sp_add* —en este caso, 8—. Merece la pena fijarse en que Microsoft SQL Server crea procedimientos temporalmente almacenados para las sentencias *rdoQuery*; estos procedimientos temporalmente almacenados se eliminan automáticamente si la conexión con el servidor se rompe de forma anormal.

Una implementación del procedimiento almacenado *sp_add* en el lenguaje SQL Transact que utiliza Microsoft SQL Server, sería como la siguiente:

```
create proc sp_add @x int = 0, @y int = o as ↵
return @x + @y
```

El aspecto más impresionante de Visual Basic 5.0 Edición empresarial, es la capacidad para depurar procedimientos almacenados de SQL Transact. Esto se lleva a cabo mediante un complemento de Visual Basic denominado el Depurador T-SQL de VB. Después de instalar los componentes de depuración del lado del cliente, de acuerdo con la documentación de Visual Basic, elija la orden Complementos, Administrador de complementos y verifique Depurador T-SQL de VB. Después, en el menú Complementos, elija Depurador T-SQL de VB. Aparecerá un cuadro de diálogo como el de la Figura 14-6.

En el cuadro de diálogo, introduzca el DSN del servidor con el que desea conectar con propósitos de depuración. Encontrará opciones adicionales de depuración, eligiendo Opciones del depurador T-SQL en el menú Herramientas. Ahora utilizando el depurador de Visual Basic, ejecute el código sentencia a sentencia. Cuando llegue a la sentencia que llama al método *rdoQuery.Execute*, emerge el Depurador T-SQL, conecta con el servidor, carga el procedimiento almacenado y le permite que lo depure. Utilizando el complemento, es posible ejecutar paso a paso el código que se ejecuta en Microsoft SQL Server, inspeccionar variables y fijar puntos de interrupción. Utilizando la pestaña Procedimiento almacenado del cuadro de diálogo del complemento, también es posible depurar de forma interactiva cualquier procedimiento almacenado disponible en la selección, tal y como muestra la Figura 14-7 de la página siguiente.

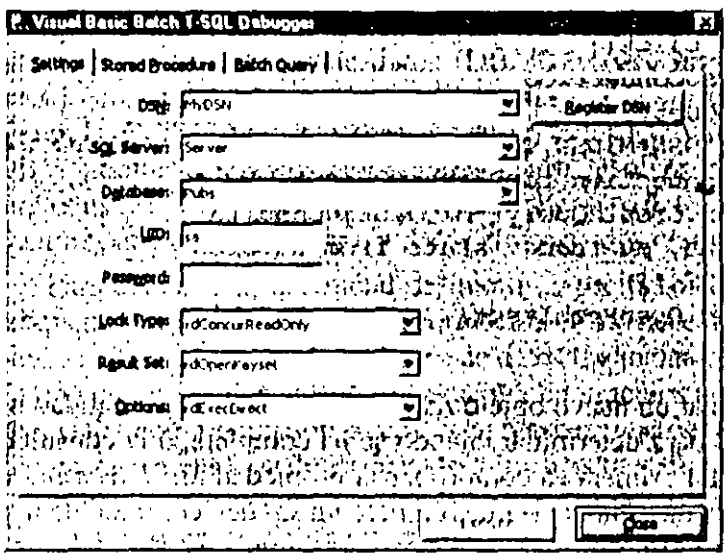


Figura 14-6. *Cuadro de diálogo Depurador T-SQL de VB.*

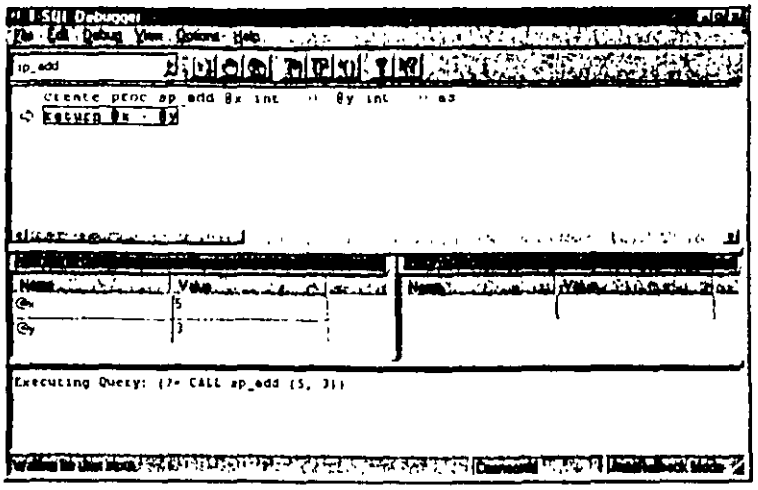


Figura 14-7. *Depurador interactivo—Depurador T-SQL.*

Cómo manejar consultas asíncronas

Normalmente, cuando se llama al método *OpenResultset* del objeto *rdoQuery*, se suspende la ejecución de la aplicación hasta que se completa la consulta. Esto es un derroche, ya que la mayoría del procesamiento tiene lugar en el servidor. RDO hace posible la ejecución de consultas asíncronas —significando que el foco de la ejecución se devolverá a la aplicación de inmediato, incluso aunque el servidor todavía no haya terminado la consulta—. Para hacer asíncrona una operación, pase el indicador *rdAsyncEnable* al método *OpenResultset*, como se muesta en el siguiente código en negrita:

```
Dim WithEvents myCon As rdoConnection
Din myRes As rdoResultset

Private sun Commandl_click()
    Set myCon = NewrdoConnection
    Din myQry As rdoQuery
    myCon.Connect = "DSN=MiDSN;UID=MilDdeUsuario;PWD =MiContraseña;"
    myCon.EstablishConnection rdoDriverNOPront, False
    Set myQry = myCon.CreateQuery("MiQuintaConsulta", &_
      "Select au_lname, au_fname, state from autores" &_
      "Where state='DS'")
    Set myRes = myQry.OpenResultset(,,rdAsyncEneable)
.End Sub
```

Todavía no se puede utilizar un nuevo objeto *rdoResultset* por que en realidad no se ha creado. Existen dos opciones para determinar cuándo se ha completado la consulta y *rdoResultset* ya está preparado. La primera es comprobar la propiedad *StillExecuting* de *rdoResultset*. Este método tiene la desventaja de que requiere un sondeo continuo de la propiedad *StillExecuting*. Una forma más sofisticada de determinar cuándo está preparado *rdoResultset*, es reaccionar al evento *rdoConnection_QueryComplete*, activado cuando se

termina la ejecución de la consulta. Utilizando este método, la aplicación puede remitir la consulta, marcharse, hacer algo más y después se le notifica de forma asíncrona por medio de evento *QueryComplete*, que se ha completado la consulta.

```
Private Sub myCon_QueryComplete(ByVal Query As
  RDO.rdoQuery, ByVal ErrorOcurred As Boolean)
      Do Until myRes.EOF
          Print myRes.rdoColumns(0), Print myRes.rdoColumns(1),
          myRes.MoveNext
      Loop
End Sub
```

Recuerde que la palabra clave *WithEvents* utilizada en el código anterior, es necesaria para interceptar los eventos activados por controles ActiveX como RDO. Después de remitir la consulta, la operación se puede abortar utilizando el método *Cancel*.

El control de datos Remote

Aunque RDO es muy útil y eficiente, no puede utilizar controles de datos enlazados. Para admitir el modelo de control de datos enlazados de Visual Basic, es necesario el control de datos Remote. Los controles de datos enlazados no pueden distinguir entre el control de datos de Jet y los controles de datos Remote, los cuales se parecen y comportan como el control de datos de Jet. Pero existe una gran diferencia entre ellos: el control de datos de Jet obviamente está implementado utilizando el motor de la base de datos Jet, mientras que el control de datos Remote se implementa mediante llamadas a la API de ODBC, dando un potente modelo de control de datos enlazados sin el motor de Jet. Además, al igual que se puede combinar el control de datos Jet con DAO, el control de datos Remote se puede combinar con código RDO.

OLE DB

Aunque ODBC proporciona acceso a datos almacenados en una variedad de sistemas de administración de bases de datos (DBMS), posee un par de limitaciones. Primera, está basado en una interfaz a nivel de llamadas (CLI) diseñada para programas escritos en C. Segunda, ODBC no proporciona acceso a todos los tipos de datos. La especificación OLE DB es un nuevo estándar con la intención de resolver las limitaciones de ODBC. OLE DB está constituido por un conjunto de interfaces basadas en COM que proporcionan a las aplicaciones un acceso uniforme a datos almacenados en diversos orígenes de información.

Para entender la necesidad de nuevas especificaciones de acceso a datos, es necesario saber que una vasta cantidad de información crítica necesaria para mantener día a día los negocios, se encuentra fuera de las bases de datos tradicionales de la producción de la corporación. En cambio, esta información se encuentra en sencillos archivos, dentro de las bases de datos personales como Microsoft Access y en herramientas de productividad como hojas de cálculo, documentos de procesadores de texto, planificadores de administración de proyectos y correos electrónicos. Para beneficiarse de la tecnología de las bases de datos —como las consultas SQL, procesamiento de transacciones y seguridad— las empresas tienen que mover primero los datos desde su ubicación original hasta un DBMS. Este proceso es caro y redundante.

Con OLE DB, la base de datos tal y como se conoce hoy en día, se convierte en un componente llamador y proveedor de datos OLE DB. Sin embargo, además de las bases de datos, otros orígenes de datos como las hojas de cálculo, documentos y correo electrónico, pueden exponer las interfaces necesarias para que se les clasifique como proveedores de datos OLE DB, se les denomina consumidores de datos OLE DB. Estos programas pueden acceder a los datos de cualquier proveedor OLE DB.

Por ejemplo, consideremos un representante de ventas que quiere encontrar todos los mensajes recibidos en la última semana de los consumidores de una determinada región geográfica, incluyendo sus direcciones, a las cuales todavía no ha respondido nadie. Esta consulta implica la búsqueda en los buzones que contengan el correo electrónico del representante de ventas, así como la búsqueda en una base de datos de consumidores, dentro de la base de datos de la empresa. OLE DB hace posible formular una consulta SQL que obtendría este tipo de información.

Utilizando OLE DB, los DBMS se convierten en un conglomerado de componentes cooperantes que consumen y producen datos a través de un conjunto uniforme de interfaces. Definiendo un conjunto uniforme de interfaces de acceso a datos, los componentes OLE DB no sólo contribuyen al acceso uniforme entre aplicaciones de datos de diversos orígenes de información, sino que también ayudan a reducir las huellas de las aplicaciones, permitiéndolas utilizar sólo la funcionalidad de DBMS que necesiten. Las áreas funcionales de OLE DB incluyen los accesos y actualizaciones de datos (conjuntos de filas), procesamiento de consultas, catálogos de información, notificaciones, transacciones, seguridad y acceso a datos remotos.

La tecnología ODBC ha madurado hasta el punto en que ODBC es una tecnología ideal para el acceso a bases de datos SQL. Como resultado, una parte integral de OLE DB es un nuevo administrador de controladores OLE DB, que permite a los consumidores OLE DB establecer coloquios con los proveedores OLE DB. Esto permite que los productos y herramientas de los consumidores de datos OLE DB, tengan pleno acceso a todos los controladores ODBC/y a los datos basados en ODBC. Microsoft lo está consiguiendo al suplantar el Administrador de controladores ODBC por un proveedor de datos OLE DB de datos ODBC. De esta forma, no se añaden nuevas capas. La única diferencia material es que los consumidores OLE DB se comunican con las interfaces de OLE DB, más que con las API de ODBC. La siguiente tabla compara aspectos técnicos de ODBC y OLE DB.

Atributo	ODBC	OLE DB
Interfaz de programación	API a nivel C	API basada en COM
Formato de datos	Datos basados en SQL	Todos los datos tabulados
Método de consulta	Estándar basado en SQL	Estándar basado en COM
Proveedores	Proveedores nativos	Arquitectura de componentes

Objetos de datos ActiveX

Las interfaces OLE DB se diseñaron principalmente para que las utilizaran ~~los~~ programadores de C++. Para aprovecharse de las ventajas disponibles de OLE DB en entornos de desarrollo de alto nivel. Microsoft ha implementado un modelo de objetos basado en Automatización, llamado Objetos de datos ActiveX (ADO). Al igual que los Objetos de datos Remote se implementan utilizando la API de ODBC.

Los Objetos de datos ActiveX se implementan utilizando las interfaces de OLE DB. Al principio, puede que cree confusión el tener tres modelos de objetos separados para acceder a datos desde Visual Basic: DAO, RDO y ADO. Pero ADO es una evolución de los anteriores modelos de objetos. Con el paso del tiempo, las cosas probablemente se estandaricen a ADO y OLE DB.

Nota: *El componente ADO actualmente está disponible en el SDK de OLE DB y en Internet Information Server 3.0. En el futuro, puede que se adjunte a Visual Basic.*

Los Objetos de datos ActiveX permiten escribir aplicaciones cliente para acceder y manipular datos dentro de una base de datos, a través de un proveedor. Las principales ventajas de ADO son facilidad de uso, alta velocidad, baja sobrecarga de memoria y pequeñas huellas en disco. La siguiente tabla compara el tamaño de los diferentes componentes de acceso a datos discutidos en este capítulo.

Componente de acceso a datos	DLL principal	Tamaño
Objetos de acceso a datos (DAO)	DA0350.DLL	568 KB
Objetos de datos Remote (RDO)	MSRDO20.DLL	403 KB
Objetos de datos ActiveX (ADO)	MSADO10.DLL	202 KB
Conjunto de registros de datosActiveX (ADR)	MSADRS10.DLL	28 KB

El modelo ADO se muestra en la Figura 14-8.

En ADO, los objetos *Connection*, *Command* y *Recordset*, son las principales interfaces de datos. Estos objetos también tienen una colección *Properties*, como se muestra en la Figura 14-9, de la página siguiente.

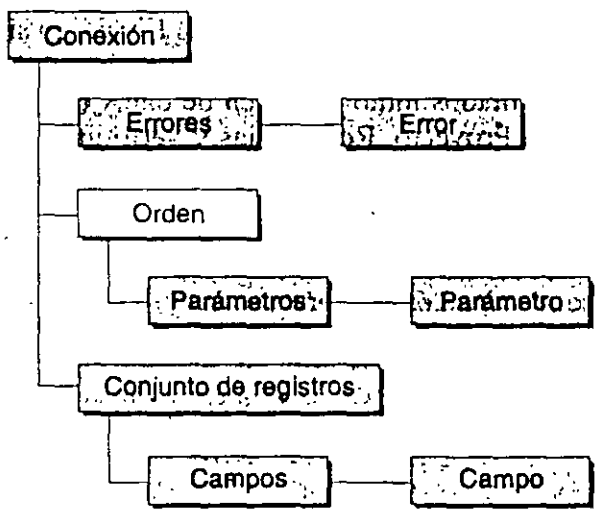


Figura 14-8. *Modelo ADO.*

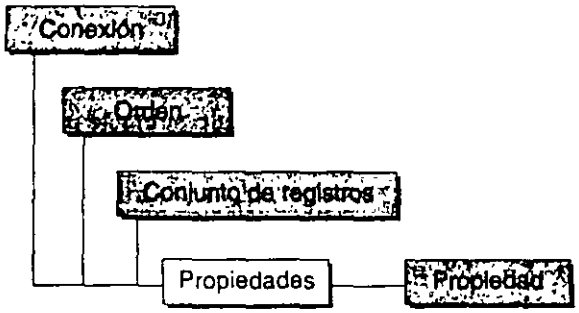


Figura 14-9. *Colección* Properties *de los objetos* Connection, Command *y* Recordset.

En ADO, la jerarquía de objetos no está resaltada. A diferencia de DAO o RDO, ya no se tiene que navegar a través de una jerarquía para crear objetos ya que la mayoría de los objetos de ADO se pueden crear de forma independiente. Esto permite que sólo se creen y vigilen los objetos que se necesitan. Este modelo también da como resultado menos objetos ADO y por lo tanto, un conjunto de áreas de trabajo menor. ADO admite aspectos para la creación de aplicaciones cliente/servidor y basadas en Web, incluyendo las siguientes:

- Independencia de los objetos creados.
- Actualización por lotes.
- Procedimientos almacenados con parámetros de entrada/salida y valores devueltos.
- Diferentes tipos de cursores, incluyendo el potencial para admitir los cursores de finalidad específica.
- Administración avanzada de la caché del conjunto de registros.
- Límite en el número de filas a devolver y otros objetivos de consulta.
- Múltiples conjuntos de registros devueltos por procedimientos almacenados o sentencias por lotes.
- Objetos de hebras libres para aplicaciones eficientes de servidor Web.

Cuando se utiliza ADO desde Visual Basic, se puede usar enlace a posteriori (*CreateObject*) o enlace a priori, por medio del cuadro de diálogo Referencias. El código siguiente, el cual utiliza la última solución por que obtiene mayor rendimiento, utiliza ADO para ejecutar una consulta SQL y mostrar los resultados junto al origen de datos ODBC. Para asignar una referencia a ADO, elija la orden Proyecto, Referencias y verifique Microsoft OLE DB ActiveX Data Objects 1.0 Library.

```
Dim adoRs As New ADODB.Recordset
adoRS.ActiveConnection =
  "DSN=MiDSN;UID=MiIDdeUsuario;PWD=MiContraseña;"
adoRs.Open "select au_lname, au_f_name, state from authors"

Do Until adoRs.EOF
    Print adoRs.Fields (0), Print adoRs.Fields (1),
    adoRs.MoveNext
Loop

adoRs.Close
```

Para poder actualizar registros utilizando ADO, se tiene que asignar *adLockOptimistic* o *adLockBatchOptimistic* a la propiedad *LockType* del conjunto de registros, tal y como aquí se muestra:

```
Dim adoRs As New ADODB.Recordset
Dim NewLastName As String
adoRS.ActiveConnection =
  "DSN=MiDSN;UID=MiIDdeUsuario;PWD=MiContraseña;"
adoRs.LoxkType = adLockOptimistic
AdoRs.Open"select au_lname, au_fname, state from authors"

Do Until adoRs.EOF
    NewLastName = InputBox("El apellido es" &_
      AdoRs.Fields(0)&_
      ".¿A cuál desea cambiarlo"?)
    IfNewLastName = ""Then Exit Do
    AdoRs.Fields(0) = NewLastName
    adoRs.MoveNext
Loop

adoRs.Close
```

En el fragmento anterior de código, la conexión se realiza automáticamente utilizando el DSN de ODBC, pasado al método *Open* del objeto *Recordset*. Aunque es aceptable para un código de ejemplo, las aplicaciones que pretendan tener un mayor control sobre el proceso de conexión, pueden crear de forma explícita con un objeto Connection, como se muestra abajo:

```
Dim adoCon As New ADODB.Connection
Dim adoRs As New ADODB.Recordset

adoCon.Open "MiDSN", "MiIDdeUsuario", "MiContraseña"
adoRs.ActiveConnection = adoCon
adoRs.Open "select au_lname, au_fname, state from authors"

Do Until adoRs. EOF
    Print adoRs.Fileds!au_lname, adoRs.Fields!au_fname
    adoRs.MoveNext
Loop

adoRs.Close
```

En este ejemplo, se llama al método *Open* del objeto *Connection*. Después se asigna el objeto *Connection* a la propiedad *ActiveConnection* del objeto *Recordset*. Se puede utilizar un objeto *Command* para ejercer mayor control sobre las órdenes de la base de datos.

```
Dim adoCmd As New ADODB.Command
Dim adoRs As New ADODB.Recordset

    adoCmd.ActiveConnection=_
      "DSN=MiDSN;UID=MiIDdeUsuario;PWD=MiContraseña;"
    adoCmd.CommandText = "Select au_lname from authors"
    Set adoRs=adoCmd.Execute
```

(continúa)

(continuación)

```
Do Until adoRs.EOF
    Print adoRs.Fields!au_fname
    adoRs.MoveNext
Loop
AdoRs.Close
```

En el fragmento anterior de código, se llama al método *Execute* del objeto *Command* para ejecutar una consulta SQL. Nótese que el método *Execute* devuelve un objeto *Recordset* que se almacena en una variable de objeto.

Conjunto de registros de datos ActiveX

En adición y por separado de los Objetos de datos ActiveX, también existe el componente Conjunto de registros de datos ActiveX (ADR). Es un componente de acceso sólo lectura a bases de datos que se integra al componente completamente funcional ADO. El objetivo de ADR es el de ser una implementación óptima y mínima del conjunto de registros —de forma específica, facilitar la implementación en Internet—. Es necesario que sea pequeño para acomodar las características de carga y es necesario que sea un conjunto mínimo, para satisfacer las necesidades básicas del consumidor en los conjuntos de registros, sin restablecer la implementación completa de ADO. El modelo de objeto ADR se muestra en la Figura 14-10.

Para utilizar el componente Conjunto de registros de datos ActiveX desde Visual Basic, se debe verificar el elemento Microsoft ActiveData Objects Recordset 1.0, en el cuadro de diálogo Referencias. El siguiente ejemplo muestra cómo se utiliza ADR en una página Web:

```
<HTML>
<HEAD>
<TITLE>Un ejemplo de página Web con acceso ADR</TITLE>
</HEAD>
<BODY>

<!--Creación de ADR.Recordset-->
</OBJECT
ID="rs"
CLASSID="CLSID:0CB666E5-D532-11CF-B606-00A0C9138C1D">'ADR
</OBJECT>

<SCRIPT LANGUAGE="VBScript">
rs.ActiveConnection = "data source=MiDSN;"&_
  user id=sa;password=;
rs.Source = "Select au_lname, au_fname from authors"
```

(continúa)

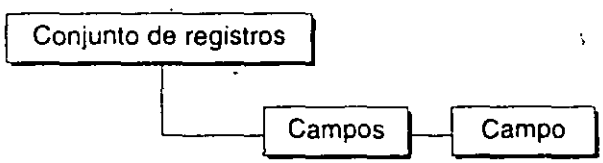


Figura 14-10. *Modelo de objeto ADR.*

(continuación)

```
    rs.Open

    Do Until rs EOF
       document.writeln rs.fields(0) + ", " + rs.fields(1)_
       + "<BR>"
       rs.MoveNext
    Loop
    rs.close
    -->
    </SCRIPT>
    </BODY>
    </HTML>
```

Nota: *Antes de ejecutar este ejemplo, asegúrese de que el nivel de seguridad de Microsoft Internet Explorer no está asignado a Alto. En Microsoft Internet Explorer, elija la orden Ver, Opciones y seleccione la pestaña Seguridad. Pulse Nivel de seguridad y después seleccione Media o Ninguna.*

En el código HTML anterior, nótese que el número CLSID utilizado es el del componente ADR. Éste será el número correcto en toda máquina en la que esté instalado ADR. Se puede convertir este ejemplo para que utilice ADO, cambiando simplemente el número CLSID al del componente ADO, tal y como sigue:

```
classid="clsid:00000281-0000-0010-8000-00AA006D2EA4">      'ADO
```

La aplicación de cronometraje VBDB

La aplicación de cronometraje VBDB compara cuatro de los cinco métodos principales de acceso a datos en Visual Basic discutidos en este capítulo: DAO, ODBC, RDO, OLE DB y ADO. Los cuatro métodos se utilizan de forma secuencial para consultar un origen de datos ODBC que se especifica haciendo referencia a su DSN. La aplicación también pide el ID de usuario y la contraseña necesaria para conectar. (Los puede dejar en blanco si su origen de datos no los requiere.) La Figura 14-11 muestra la aplicación de cronometraje VBDB.

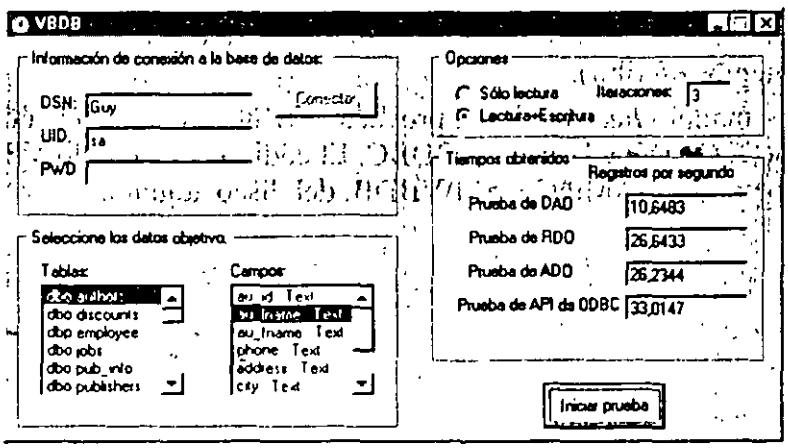


Figura 14-11. *Aplicación de cronometraje VBDB.*

VBDB se conecta y consulta la base de datos por medio de DAO para determinar qué tablas y campos hay disponibles en ese origen de datos. Elija un campo único de la tabla sobre el que se ejecutarán las pruebas. Puede decidir si desea probar leyendo solamente los registros o probar leyendo y escribiendo registros en el servidor. Si se dice a VBDB que pruebe leyendo y escribiendo registros, primero lee un registro y después invierte los datos para escribirlos. Si en este momento se muestran datos en el servidor, encontrará que se han invertido. Si después ejecuta el programa de nuevo, éste invertirá los datos de nuevo a su forma original. No le recomendamos que ejecute esta prueba con datos de interés, debería crear una base de datos ejemplo para esta prueba.

La prueba comienza utilizando DAO, continúa con RDO y ADO y después finaliza con la API de ODBC. Una vez terminada la prueba, los resultados le permitirán comparar las velocidades de acceso de cada método. La taba siguiente muestra los resultados del cronometraje de prueba obtenidos, cuando se ejecuta esta aplicación bajo Microsoft Windows 95 en un Pentium Pro a 200 MHz, conectado a través de la red a Microsoft SQL Server 6.5 ejecutándose bajo Microsoft Windows NT 4.0 en un Pentium a 90 MHz. Las pruebas se ejecutaron sobre el campo au_lname de la tabla de autores de la base de datos Pubs, que viene con Microsoft SQL Server.

Método de acceso a datos	Número de registros leídos y escritos por segundo
DAO	10
RDO	26
ADO	26
API de ODBC	33

Se puede ver que RDO y la API de ODBC dejan a DAO en el polvo. El resultado no esperado es que RDO a veces sobrepasa a la API de ODBC por un pequeño margen, cuando sólo se leen registros. La explicación de esta discrepancia radica en cómo lee los registros VBDB utilizando la función de ODBC *SQLExtendedFecth*. VBDB llama a *SQLExtendedFetch* para leer un registro cada vez, imitando a como se leen utilizando DAO Y RDO. Sería más eficiente leer un lote de registros en cada llamada a *SQLExtendedFetch*. RDO hace este tipo de comparación inteligente de forma automática y de este modo se mejora la velocidad de RDO. Esta lógica es congruente con el hecho de que la API de ODBC es el método de acceso más rápido cuando se leen y escriben registros. Este tipo de lectura frontal compacta no se puede hacer cuando se escriben datos de regreso en el servidor (ya que RDO no sabe lo que intentamos escribir).

Puede que desee ver el módulo *ReadWrite* del proyecto VBDB. Contiene el código de cronometraje para DAO, RDO, ADO y la API de ODBC. El código del ejemplo VBDB se puede encontrar en la carpeta \BK-SAMP\Chap14\VBDB, del disco adjunto.