INSTITUTO DE EDUCACION SECUNDARIA "LORENZO HERRERO"

ÍNDICE

1. Conceptos Generales 21

Conozcamos Visual Basic 21

Ediciones de Visual Basic 22

Novedades de Visual Basic 6 23

Novedades del acceso a datos 23

Novedades para Internet 25

Novedades de los controles nuevos 26

Nuevas características de los controles 27

Novedades de la creación de componentes 27

Novedades de las características del lenguaje 29

Novedades de los asistentes 30

Generar aplicaciones basadas en datos con Visual Basic 32

2. Instalación de Visual Basic 6 33

Requisitos del Sistema y del Hardware 33

Instalación de Visual Basic 34

VISUAL BASIC 6 A FONDO

© Sergio Árboles - Luis Navarro
© De la edición: INFORBOOK'S, S.L.

Se ha puesto el máximo interés en ofrecer al lector una información completa y precisa. Sin embargo, *INFORBOOK'S, S.L.* no asume ninguna responsabilidad derivada de uso, ni tampoco por cualquier violación de patentes y otros derechos de terceras partes que pudieran ocurrir estando este libro destinado a la utilización de aficionados o a la enseñanza. Las marcas o nombres mencionados lo son únicamente a título informativo, siendo propiedad de sus legales registradores.

Reservados todos los derechos.

Ninguna parte de este libro puede ser reproducida, grabada en sistema de almacenamiento o transmitida en forma alguna ni por cualquier procedimiento, ya sea electrónico, mecánico, reprográfico, magnético o cualquier otro. Asimismo queda prohibida la distribución, alquiler, traducción o exportación sin la autorización previa y por escrito de *INFORBOOK'S, S.L.*

Editado por: INFORBOOK'S, S.L.
c/ Muntaner, 181
08036 BARCELONA
93 439 15 75
email: info@inforbooks.com
web: http://www.inforbooks.com

ISBN: 84-95318-06-7
Depósito legal: B-28544-99
Diseño cubierta: Daniel Benito Cortázar
Filmación: MOLAGRAF, S.L.
Impreso por: IMGESA

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

CONSULTE NUESTRO CATÁLOGO POR INTERNET:

http://www.inforbooks.com

Podrá estar al corriente de todas las novedades.

A Carlos Agut y Javier Benayas
con afecto

Sergio

A la familia Perella Pomar
gracias por su amistad

"Un amigo es una persona con la que se puede pensar en voz alta"

Luis

INSTITUTO DE EDUCACIÓN SECUNDARIA "SOROLLA HERRERA"
ALCAZAR DE SAN JUAN (Ciudad Real)

5. La primera aplicación 87

Creación de un nuevo proyecto 87

Creación de la aplicación 88

Diseñar el Formulario 88

Establecer las propiedades del Formulario y los Controles 90

Escritura del código 93

Guardar un proyecto 95

Ejecución de la aplicación 97

Descargar un proyecto de la memoria 98

Recuperar un proyecto 98

Mejorando la aplicación 99

Finalizar la sesión con Visual Basic 100

La aplicación de la Euro-Calculadora 101

6. Controles más usuales 109

Aplicación con barras de desplazamiento y cuadros contenedores 109

Mejorando la aplicación 120

7. Matrices 125

Matrices unidimensionales 125

Matrices multidimensionales 127

Matrices dinámicas 129

Preservar el contenido de las matrices dinámicas 129

Registros 130

Acceso a los campos de un registro 131

Matriz de registros 131

Aplicación con casillas de verificación, botones de opción, cuadros contenedores y cuadros de lista 132

Orden de tabulación 146

Pequeños retoques 147

8. Imágenes e iconos en los formularios 159

El control Image 159

El control PictureBox 166

Asignar un icono a una aplicación 173

Creación de imágenes compuestas 178

Asignar un fondo al formulario 182

9. Arrays de controles 185

Diseño de un array de controles 185

10. Trabajando con menús y cuadros de diálogo . 199

Diseño de un menú 200

Creación de un Editor de textos 206

Creación del menú Archivo 209

Cuadros de diálogo 209

Creación del menú Edición 214

Creación del menú Formato 217

Creación del menú Acerca de 220

Mejorando el Editor de textos 223

Mejorando el menú Archivo 223

Mejorando el menú Formato 225

Un nuevo control para mejorar la aplicación 229

Programas instalados 45
Agregar o quitar componentes de Visual Basic 46
Cómo obtener ayuda 46

3. Fundamentos de Programación 49

Estructura de una aplicación en Visual Basic 49
Aplicaciones controladas por eventos 50
Consideraciones a tener en cuenta 51
Tipos de módulos 51
Conceptos básicos del código 52
Introducir comentarios en el código 52
Combinar instrucciones en una línea 53
Dividir una instrucción en varias líneas 53
Reglas de nomenclatura en Visual Basic 54
Declaración de variables 54
Alcance de las variables 56
Declaración de constantes 57
Declaración de tipos 58
Tipos de datos numéricos 59
Tipos de datos Byte 60
Tipos de datos String 60
Tipos de datos Boolean 60
Tipos de datos Fecha 61
Tipos de datos Object 61
Tipos de datos Variant 61
Introducción a los Procedimientos 62
Procedimientos Sub 63
Procedimientos Function 63

Procedimientos Property 64
Forzar la salida de un Procedimiento 66
Pasar argumentos a Procedimientos 66
Estructuras de control de un programa 67
Estructuras de decisión 68
Estructuras de bucle 71
Conversión de tipos de datos 75
Sistemas de numeración 76

4. Desarrollar una aplicación con Visual Basic 77

Conceptos de Visual Basic 77
Ventanas 77
Eventos 78
Mensajes 78
El entorno de desarrollo (IDE) 79
Barra de menús 80
Menús contextuales 80
Barras de herramientas 80
Cuadro de herramientas 80
Ventana Explorador de proyectos 81
Ventana Propiedades 81
Examinador de objetos 81
Diseñador de formularios 82
Ventana Editor de código 82
Ventana Posición del formulario 83
Ventanas Inmediato, Locales e Inspección 83
Opciones de entorno 83
Creación de una aplicación 84
Descripción de las propiedades, métodos y eventos 85

11. El ratón 235

El evento MouseDown 236

El evento MouseMove 238

El evento MouseUp 241

Personalizar el puntero del ratón 242

Arrastrar y colocar con el ratón 245

12. Imprimir con Visual Basic 255

El método PrintForm 256

El objeto Printer 257

Propiedades y métodos del objeto Printer 258

Propiedades 258

Métodos 260

Impresoras que hay instaladas 261

Un ejemplo del objeto Printer 263

Interceptar los errores de impresora 265

13. Programación de Objetos 267

Importancia de la POO 267

Programas y Abstracción 267

Orientación a Objetos 268

Lenguajes de Programación Orientados a Objetos (LPOO) 269

Introducción a la POO 270

¿Qué es la POO? 270

Introducción a las relaciones 272

Generalización/Especialización 273

Agregación 273

Asociación 273

Herencia 273

¿A qué clase pertenece un Objeto? 274

Ejecutar acciones sobre un Objeto 277

La propiedad por defecto 278

Crear matrices (Arrays) de objetos 279

Las colecciones de objetos 281

¿Qué es el Examinador de objetos? 284

14. Gráficos estadísticos 287

¿Cómo se utiliza el componente MSChart? 287

¿Cómo insertar títulos en el gráfico? 292

Establecer tipos de gráficos 293

15. Gestión de Bases de datos 297

El control Data 299

Gestionar la Base de datos con el control Data 304

Recordset de tipo Table 306

Recordset de tipo Dynaset 306

Recordset de tipo Shapshot 306

Gestionar una base de datos sin el control Data 310

Utilizar rejilla de datos 327

Utilizar el nuevo control ADODC 331

Utilizar el control DataGrid con ADODC 335

Utilizar los controles DataList y DataCombo 337

16. Diseñar informes 341

Utilizar Data Report 341

Cómo visualizar el informe 349

Decorar el informe 351

Informes con Crystal Reports 353

Diseñar el informe con Crystal Reports 353

Ejecutar un informe desde Visual Basic 358

Imprimir etiquetas con Crystal Reports 361

17. Utilizar el API de Windows 367

Acceder al API de Windows 368

Utilizar el Visor de texto API 368

Usar procedimientos desde Visual Basic 370

Cargar automáticamente los archivos API 373

Ver el archivo Win32API con un editor de texto 373

18. Crear ficheros de Ayuda 375

Creación del fichero .RTF 375

Compilar el fichero de ayuda 382

Insertar el fichero de ayuda en la aplicación 386

Utilizar la tecla F1 para pedir ayuda 390

19. Multiple Document Interface (MDI) 393

Crear un aplicación MDI 393

Escribir el código 395

La propiedad WindowList 399

Detectar cuándo finaliza la aplicación 400

20. Controles avanzados 403

El control ProgressBar (Barra de progreso) 403

Controles de sistema 405

Formateo de números, fechas y horas 411

El control StatusBar 415

El control SysInfo 420

El control CoolBar 426

Agregar controles a un CoolBar 427

Mostrar imágenes en un control CoolBar 428

Usar otros controles con CoolBar 430

21. Depuración de errores 431

Utilizar el depurador de programas 432

Controlar los errores 436

Trucos para detectar e inspeccionar errores 438

Errores interceptables más comunes 441

22. Cómo crear una Barra de herramientas 461

Funcionamiento de los controles ToolBar e ImageList 461

Escribir el código para cada uno de los botones 467

Agrupación de botones dentro de la barra 468

Insertar una ayuda en los botones (ToolTipText) 469

Cómo insertar separadores de botones 471

Cambiar la posición de la Barra de botones 472

Asignar un aspecto de 3D a la Barra de herramientas 473

23. Controles Multimedia 475

Crear un reproductor de CD 477
El control Animation 486
Opciones del método Play 490

24. Funciones de correo (MAPI) 493

El control MAPISession 493
El control MAPIMessage 496
Cómo acceder y enviar mensajes 497
Cómo administrar los mensajes 500
Trabajar con la Libreta de direcciones 501

25. Comunicaciones 503

Cómo se establece una conexión 504
Manejar el control MSComm 504
Propiedades de MSComm de interés 508

26. Crear controles ActiveX 511

Actualizar VBX a controles ActiveX 511
Cómo cargar los controles ActiveX 511
Clases de control ActiveX 512
Creación de un control ActiveX 513
Ejecutar el control Etiqueta en tiempo de diseño 516
Procesos de un objeto UserControl 518

Cómo dibujar el control Etiqueta 520
Guardar las propiedades del control 522
Asignar una página de propiedades al control 525
Crear un evento para nuestro control 529
Compilación de nuestro objeto 531
Utilizar nuestro control desde otras aplicaciones 532

27. Crear un componente EXE ActiveX 537

Crear el proyecto 537
Cómo mostrar el formulario de nuestra clase 540
Probar el componente desde otro proyecto 542
Ejecutar la aplicación 545

28. Crear una DLL ActiveX 547

Creación de la DLL 547
Crear propiedades y métodos 550
Programar los eventos Initialize y Terminate 553
Probar la DLL (Objeto) 554
Compilación de la DLL Objeto 559
Utilizar nuestra DLL en otras aplicaciones 560

29. Crear un documento ActiveX 561

Crear un proyecto DocuAcX 561
Ejecutar el proyecto 563
Depurar nuestro documento 564
Otro documento ActiveX para nuestro proyecto 564

Insertar un formulario a nuestro proyecto DocuAcX 567
Insertar una propiedad a nuestro objeto Document 569
Insertar un menú al proyecto DocuAcX 573

30. Asistente para distribuir Aplicaciones 577

Asistente para empaquetado y distribución 577
Kit de herramientas de instalación 578
Empaquetado de una aplicación usando el asistente 579
Archivos de distribución permitida 579
Pasos necesarios para empaquetar 580
Pasos necesarios para distribuir 580
¿Cómo generar una instalación? 581
Instalar la Aplicación 589
Cancelar la instalación de la Aplicación 591

31. Crear ficheros de recursos 593

Instalar y conocer el Editor-Compilador de recursos 593
Crear los recursos necesarios 596
Crear nuestra aplicación para utilizar los recursos 602
Utilizar el fichero de recursos sin cambiar el idioma 611

32. Asistente para aplicaciones 615

Cómo utilizar el Asistente para aplicaciones 615

33. Administrar proyectos 627

Mostrar la ventana Explorador de proyectos 628

Añadir y quitar elementos del proyecto 629
Cómo se establece el formulario principal 631
Trabajar con múltiples proyectos 633
Añadir y quitar controles a nuestro proyecto 635

34. Internet 637

Cómo establecer la propiedad AccessType 638
Llamada al método OpenURL 640
Cómo usar el método Execute con el protocolo FTP 642
Cómo usar el evento State con el método GetChunk 644
Cómo crear un sencillo navegador 645
Cómo ver el código HTML de una página 651
Propiedades, métodos y eventos de WebBrowser 655

35. El Compilador y sus opciones 663

Optimizar para código rápido 664
Optimizar para código reducido 664
Sin optimización 664
Optimizar para Pentium Pro (tm) 664
Generar información de depuración simbólica 665
Botón Optimizaciones avanzadas 665
Las constantes del compilador 667
Directivas #If...Then...Else# 668
Directiva Const 669
Las constantes de Visual Basic 669

36. Intercambio dinámico de datos 685

La función del DDE 685

Cómo crear un enlace en tiempo de diseño 686

Eventos de vinculación 698

LinkClose 699

LinkError 699

LinkNotify 700

LinkOpen 701

LinkPoke 702

LinkSend 702

LinkRequest 703

Constantes utilizadas en DDE 703

37. Una aplicación completa 705

Crear la base de datos 706

Crear las diferentes tablas 707

Crear las relaciones 715

Crear el formulario y menú principal de la aplicación 717

Crear la base para todos los formularios 721

Programar el formulario Usuario 729

El formulario Fechas 753

El formulario Citas 766

El formulario Historial 778

Modificar el formulario principal 784

Averiguar el usuario 789

Crear el listado de la tabla Historial 791

Cómo se visualiza el formulario Historial 799

Programar la función Imprimir 804

El menú Ventana 806

Realizar el EXE de la aplicación 807

Cómo hacer funcionar nuestra aplicación 809

Índice Analítico 813

1. Conceptos Generales

En este capítulo conoceremos Visual Basic, así como todas las ediciones que se han desarrollado de esta nueva versión, sin olvidar las últimas novedades incorporadas respecto a las versiones anteriores.

Conozcamos Visual Basic

Microsoft Visual Basic es la manera más rápida y sencilla de poder crear aplicaciones para Microsoft Windows. Da igual si usted es un profesional experimentado como un principiante que desea iniciarse en la programación en Windows.

Visual Basic le proporciona un juego completo de herramientas que le facilitarán el desarrollo de aplicaciones.

Pero sin duda, usted se estará preguntando ¿qué es Visual Basic?, pues bien, la palabra *Visual* hace referencia al método que se utiliza para crear la interfaz gráfica de usuario (GUI). En lugar de escribir numerosas líneas de código para describir la apariencia y la ubicación de los elementos que componen la interfaz gráfica, simplemente deberá agregar los objetos prefabricados que nos proporciona Visual Basic y colocarlos en la pantalla. La palabra *Basic* hace referencia al lenguaje BASIC (Beginners All-Purpose Symbolic Instruction Code), sin duda el lenguaje utilizado por más programadores hasta el momento en la historia de la programación.

Sin embargo Visual Basic ha evolucionado a partir del lenguaje BASIC original y ahora contiene centenares de instrucciones, funciones y palabras clave, muchas de las cuales están directamente relacionadas con la interfaz gráfica de Windows.

Así los programadores principiantes podrán crear aplicaciones útiles con sólo aprender unas pocas órdenes, pero, simultáneamente la eficacia del lenguaje permitirá a los profesionales desarrollar cualquier aplicación que pueda alcanzarse mediante otro lenguaje de programación bajo Windows.

También hay que tener en cuenta que este lenguaje de programación se utiliza en aplicaciones como Microsoft Excel, Microsoft Access y muchas otras aplicaciones Windows.

Las principales características de esta nueva versión de Visual Basic son:

- Las posibilidades de acceso a datos le permiten crear bases de datos, aplicaciones cliente y componentes de servidor escalables para los formatos de las bases de datos más conocidas, incluidos Microsoft SQL Server y otras bases de datos de ámbito empresarial.
- Las tecnologías ActiveX le permiten emplear la funcionalidad proporcionada por otras aplicaciones, como el procesador de textos Microsoft Word, la hoja de cálculo Microsoft Excel, la base de datos Microsoft Access y otras aplicaciones Windows.
- Sus capacidades sobre Internet facilitan el acceso a documentos y aplicaciones a través de Internet o Intranet desde su propia aplicación, o la creación de aplicaciones de servidor para Internet.
- La aplicación terminada es un auténtico archivo **.exe** que utiliza una máquina virtual de Visual Basic que puede distribuir con toda libertad sin necesidad de tener el programa de Visual Basic.

Ediciones de Visual Basic

Visual Basic 6 se encuentra disponible en tres ediciones, cada una de ellas destinada a un tipo de programación determinada:

- ***Edición de Aprendizaje.*** Permite a los programadores crear aplicaciones para Microsoft Windows y Windows NT. Incluye todos los controles intrínsecos, además de los controles de cuadrícula, de fichas y los controles enlazados de datos.
- ***Edición Profesional.*** Proporciona un completo conjunto de herramientas para desarrollar soluciones para terceros. Incluye todas las características de la *Edición de Aprendizaje*, así como controles ActiveX adicionales, el diseñador de aplicaciones para Internet Information Server y Visual Database Tools and Data.
- ***Edición Empresarial.*** Permite a los profesionales crear sólidas aplicaciones distribuidas en un entorno de equipo. Incluye todas las características de la *Edición Profesional*, además de herramientas Back Office como SQL Server, Microsoft Transaction Server, Internet Information Server, Visual SourceSafe, etc.

*Para la realización de este libro se ha utilizado **Visual Basic 6 Edición Profesional.***

Novedades de Visual Basic 6

Visual Basic 6 incorpora un gran número de novedades respecto a la versión anterior. A continuación describiremos las de la Edición Profesional.

Novedades del acceso a datos

- ***ADO.*** Esta nueva tecnología de acceso a datos proporciona un modelo de objetos más sencillo, una mejor integración con las tecnologías de Microsoft y las que no lo son, una interfaz común para acceso a datos locales y remotos, conjuntos de registros que se pueden controlar en modo remoto y en modo no conectado, una interfaz de enlace de datos de fácil acceso para el usuario y conjuntos de registros jerárquicos.

  Una de las principales ventajas es su facilidad de uso, su gran velocidad, baja sobrecarga de la memoria y reducido espacio necesario en disco.

  También incluye **RDS** (Remote Data Service) el cual permite mover datos de un servidor a una aplicación cliente o una página Web, manipular los datos en el cliente y devolver actualizaciones al servidor al mismo tiempo.
- ***Entorno de datos.*** El diseñador de entorno de datos proporciona un entorno interactivo de tiempo de diseño para crear objetos ADO.
- ***Control de datos ADO.*** Permite crear una aplicación de base de datos con un mínimo de código.
- ***Enlace a datos mejorado.*** Permite enlazar cualquier origen de datos ADO/OLE DB a cualquier receptor de datos ADO/OLE DB.
- ***Compatibilidad con OLE DB.*** OLE DB es un conjunto de interfaces COM que proporcionan a las aplicaciones un procedimiento unificado de acceso a los datos almacenados en diferentes orígenes de información, tanto relacionales como no relacionales.

INSTITUTO DE ESTUDIOS ALMERIENSES - (Badajoz)

- ***Mejoras en los datos del Asistente para instalar.*** El Asistente de empaquetado y distribución incorpora la compatibilidad con ADO, OLE DB, RDO, ODBC y DAO.

- ***Informe de datos.*** Permite emplear la técnica de arrastrar y colocar para crear rápidamente informes a partir de cualquier conjunto de registros, incluso conjuntos de registros jerárquicos.

- ***Orígenes de datos.*** Podrá crear controles personalizados y clases que son orígenes de datos a los que puede enlazar otros controles.

- ***Ventana Vista Datos.*** Utilice la ventana Vista Datos para examinar todas las bases de datos a las que se conecta y mostrar sus tablas, vistas, procedimientos almacenados, etc.

- ***Control Hierarchical FlexGrid.*** Versión actualizada del control **FlexGrid** que puede mostrar una jerarquía de objetos **Recorset** de ADO. Puede ver datos de varias tablas relacionadas con un formato claro y sangrado de forma lógica en una única cuadrícula.

- ***Control DataRepeater.*** Permite crear una vista personalizada de una base de datos similar a los formularios de Access. Los controles UserControl pueden contener controles TextBox, CheckBox u otros controles enlazados a campos de datos.

- ***Pasar objetos Recorset de ADO.*** Proporciona un modo eficiente de mover datos entre los diferentes niveles de una aplicación multinivel.

- ***Mejoras del Asistente para formularios de datos.*** Ofrece la posibilidad de generar formularios de sólo código donde los controles no están enlazados a un control de datos.

- ***Asistente para objetos de datos.*** Automatiza la creación de objetos de la capa media enlazados al entorno de datos o a controles de usuario.

- ***Objetos del sistema de archivos.*** Ofrece un conjunto especializado de subrutinas que permiten desplazarse por el sistema de archivos y crear archivos de texto y directorios.

- ***Objetos Format.*** Proporciona la conversión de datos en los dos sentidos entre una base de datos y un control enlazado.

- ***Control DataGrid.*** Permite generar rápidamente una aplicación para mostrar y modificar conjuntos de registros. También admite el nuevo control de datos ADO. Muestra y modifica recordsets mediante una interfaz de tipo hoja de cálculo.

- ***Control DataList, controles DataCombo.*** Estos controles son versiones para OLE DB. También admiten el nuevo control de datos ADO. Proporcionan un cuadro combinado y un cuadro de lista que puede rellenar desde un campo de base de datos del control Data al que están enlazados. Permiten cambiar dinámicamente los orígenes de datos.

Novedades para Internet

- ***Aplicaciones para IIS.*** Son aplicaciones de Visual Basic que utiliza HTML y código compilado de Visual Basic para crear una aplicación basada en explorador. Estas aplicaciones pueden ejecutarse en cualquier explorador y en una intranet o Internet.

  Las aplicaciones IIS residen físicamente en un servidor Web, donde reciben peticiones del explorador, ejecutan el código asociado a las consultas y devuelven las respuestas al explorador. También puede generar aplicaciones IIS que realicen funciones avanzadas como:

  - Consultar bases de datos, escribir información en registros e información de registros.
  - Recuperar páginas HTML y reemplazar partes con contenido dinámico antes de enviarlos al explorador.
  - Crear elementos HTML de forma dinámica y generar eventos para esos elementos en tiempo de ejecución.

- ***Aplicaciones para DHTML.*** Son aplicaciones de Visual Basic que consiste en una combinación de HTML dinámico y código compilado de Visual Basic. Estos elementos juntos crean una aplicación interactiva basada en explorador. Permiten exponer propiedades, métodos y eventos para cada uno de los elementos de una página Web. Puede generar aplicaciones DHTML más complicadas que realizan las siguientes funciones:

  - Obtener datos del usuario y utilizarlos a continuación para consultar una base de datos.
  - Actualizar la apariencia y el comportamiento de una página.
  - Crear elementos HTML e insertarlos en una página en respuesta a peticiones de usuarios.

- ***Asistente para publicación en Web.*** Utilice el Asistente de empaquetado y distribución para publicar fácilmente el paquete de su aplicación en un sitio Web.

  La interfaz de usuario de su aplicación de Visual Basic para Web puede contener:

  - Una serie de páginas HTML.
  - Una mezcla de páginas HTML y formularios de Visual Basic.
  - Un formulario de Visual Basic que utiliza controles.

- ***Mejoras para AsyncRead.*** Las operaciones asíncronas se han mejorado para proporcionar más información acerca del progreso y el estado de la operación, como el número de bytes leídos hasta el momento y el número de bytes que quedan por leer.

- ***Compatibilidad para descarga de documentos ActiveX en Internet Explorer 4.x.*** La descarga de documentos ActiveX en Internet Explorer 4.x funciona ahora del mismo modo que la descarga de controles ActiveX.

  Visual Basic ofrece soporte para ayudarle a empaquetar y descargar sus controles ActiveX, componentes de código y documentos ActiveX. Este soporte incluye:

  - *Asistente de empaquetado y distribución*, que puede ayudarle a prepara su software para la descarga de componentes de Internet.
  - La herramienta de *Creación de paquetes de licencia*, que le permite crear archivos de licencia para páginas HTML.

Novedades de los controles nuevos

- ***Control de datos ADO.*** Un nuevo control de origen de datos compatible con OLE DB le permite crear una aplicación de bases de datos con un mínimo de código.

- ***Control CoolBar.*** Un control contenedor que puede usarse para crear barras de herramientas configurables similares a las de Microsoft Internet Explorer.

- ***Control DataGrid.*** Permite generar rápidamente una aplicación para mostrar y modificar conjuntos de registros. También admite el nuevo control de datos ADO.

- ***Control DataList, controles DataCombo.*** Estos controles son versiones para OLE DB. También admiten el nuevo control de datos ADO.

- ***Control DataRepeater.*** Permite crear una vista personalizada de una base de datos similar a los formularios de Access.

- ***Control DateTimePicker.*** Coloque este control en un formulario si necesita proporcionar un calendario desplegable para escribir rápidamente las fechas y las horas.

- ***Control Flat Scrollbar.*** Permite una apariencia elegante de los controles de barras de desplazamiento planas en las aplicaciones.

- ***Control Hierarchical FlexGrid.*** Versión actualizada del control **FlexGrid** que puede mostrar una jerarquía de objetos **Recorset** de ADO.

- ***Control ImageCombo.*** Permite agregar imágenes a una lista de elementos.

- ***Control MonthView.*** Permite al usuario final elegir una fecha o un intervalo mediante una representación gráfica del calendario.

Nuevas características de los controles

- ***Mejoras de validación de datos.*** Permite usar el nuevo evento Validate combinado con la propiedad *CausesValidation* para validar el contenido del control antes de que el control pierda el enfoque.

- ***Adición dinámica de controles.*** Permite agregar o quitar controles de un formulario por programa.

- ***Mejoras de LoadPicture.*** Permite especificar el tamaño y la intensidad del color.

- ***Método ValidateControls.*** Valida el contenido del último control en el formulario antes de salir del formulario.

Novedades de la creación de componentes

- ***Crear sus propios receptores de datos.*** Puede crear receptores de datos basados en un control de usuario o en una clase que admiten la capacidad de enlace complejo.

- ***Crear sus propios orígenes de datos.*** Puede crear sus propios orígenes de datos basados en un control de usuario o en una clase.
- ***Opciones de inicio del proyecto.*** Las nuevas opciones de la tecla F5 permiten elegir dónde ejecutará un componente para depurarlo, puede ejecutar un control en un explorador en vez de ejecutarlo en un formulario de Visual Basic.
- ***Controles de usuario compactos.*** Puede crear controles de usuario sin ventana y transparentes, que consumen menos cantidad de recursos.
- ***Mejoras en las matrices.*** Ahora los procedimientos permiten devolver matrices, lo que significa que las clases pueden tener matrices como propiedades. Asimismo, es posible asignar el contenido completo de una matriz a otra matriz, siempre y cuando la matriz del lado izquierdo de la asignación sea una matriz dinámica.
- ***Persistencia de clases.*** Las propiedades de clases pueden hacerse persistentes mediante un objeto *PropertyBag* para su almacenamiento y recuperación.
- ***Soporte técnico para administrar ClassID y compatibilidad.*** Ahora, la compatibilidad de proyectos permite conservar los Id. de clase entre las versiones de componentes ActiveX.
- ***Eventos de validación para los controles de usuario.*** El evento Validate permite validar los datos de un control antes de que ocurra el evento LostFocus.
- ***Subprocesos independientes.*** Puede crear componentes ActiveX que utilizan subprocesos independientes para ejecutar procedimientos asíncronamente en subprocesos separados.
- ***Mejoras del programa Generador de clases.*** Admite las valores *ParamArray, Optional, ByVal* y *Default* en la lista de argumentos y admite también enumeraciones.
- ***Mejoras de la función CreateObject.*** Un argumento nuevo permite crear objetos en equipos remotos.
- ***Asistente de objetos de datos.*** Automatiza la creación de objetos de la capa media enlazados al entorno de datos o a controles de usuario.
- ***Evento FontChanged.*** Transmite una notificación a su control ActiveX cuando un origen externo cambia las características de la fuente.

- ***Compatibilidad con Microsoft Transaction Server.*** La nueva propiedad *MTSTTransactionMode* permite especificar el comportamiento transaccional de una clase empleada en combinación con Microsoft Transaction Server.
- ***Propiedad HasDC.*** Determina si un control personalizado debe asignar un hDC fijo durante todo su ciclo de vida en vez de usar un hDC compartido cuando sea necesario.

Novedades de las características del lenguaje

- ***Tipos definidos por el usuario en métodos públicos.*** Los tipos definidos por el usuario pueden ser argumentos o tipos devueltos de propiedades y métodos públicos.
- ***Las funciones pueden devolver matrices.*** Las funciones y los procedimientos de propiedades son capaces de devolver matrices.
- ***Las matrices de tamaño variable se pueden asignar.*** Las matrices de tamaño variable pueden ahora aparecer en el lado izquierdo de una asignación.
- ***Objetos del sistema de archivos.*** Ofrece un conjunto especializado de rutinas que permiten desplazarse por el sistema de archivos de texto y directorios.
- ***CallByName.*** Permite tener acceso a una propiedad o un método que especifica una cadena que contiene el nombre de la propiedad o del método, en vez de tener que definir su nombre explícitamente en el código.
- ***Mejoras de la función CreateObject.*** La función *CreateObject* permite ahora especificar un nombre de equipo como un argumento opcional, lo que permite crear objetos en equipos remotos.
- ***Mejoras de la función StrConv.*** Los nuevos argumentos LCID permiten especificar una variable *LocaleID* para una cadena de una configuración regional distinta de la variable *LocaleID* del sistema.
- ***Objeto Dictionary.*** El objeto *Dictionary* resuelve muchas de las limitaciones propias del objeto *Collection* y es más sencillo de usar al generar un número menor de errores en tiempo de ejecución en operaciones comunes.
- ***Funciones de filtro (Filter).*** Devuelven una matriz basada en ceros que contiene un subconjunto de una matriz de cadenas basada en unos criterios de filtro especificados.

- ***Función FormatCurrency.*** Devuelve una expresión formateada como un valor de moneda utilizando el símbolo de moneda definido en el panel de control del sistema.
- ***Función FormatDateTime.*** Devuelve una expresión formateada como una fecha o una hora.
- ***Función FormatNumber.*** Devuelve una expresión formateada como un número.
- ***Función FormatPercent.*** Devuelve una expresión formateada como un porcentaje (multiplicado por 100) con un carácter % final.
- ***Función InstrRev.*** Devuelve un tipo *Variant(Long)* que especifica la posición de la primera aparición de una cadena en otra.
- ***Función Join.*** Devuelve una cadena que se creó uniendo subcadenas contenidas en una matriz.
- ***Función MonthName.*** Devuelve una cadena que indica el mes especificado.
- ***Función Replace.*** Devuelve una cadena en la que se reemplazó una subcadena especificada con otra subcadena un número determinado de veces.
- ***Función Round.*** Redondea un número con el número de lugares decimales especificados.
- ***Función Split.*** Devuelve una matriz de una dimensión, basada en ceros, que contiene un número determinado de subcadenas.
- ***Función StrReverse.*** Devuelve una cadena en la que se invierte el orden de los caracteres de una cadena especificada.
- ***Función WeekdayName.*** Devuelve una cadena que indica el día de la semana especificado.

Novedades de los asistentes

Visual Basic incorpora muchas características para ayudarle a escribir código de una forma más rápida. Estas características incluyen:

Asistentes. Ayudan a simplificar muchas tareas que antes eran complejas.

- ***Asistente para interfaz de control ActiveX.*** Ayuda a crear la interfaz pública de un control ActiveX generado por Visual Basic una vez que se haya creado la interfaz de usuario.

- ***Asistente para migración de documentos ActiveX.*** Ayuda a convertir los formularios existentes en documentos ActiveX. Éstos pueden colocarse y activarse en contenedores de documentos ActiveX como por ejemplo Microsoft Internet Explorer.
- ***Asistente para páginas de propiedades.*** Ayuda a crear páginas de propiedades para controles de usuario.
- ***Mejoras del Asistente para instalar.*** El Asistente de empaquetado y distribución permite distribuir su archivo .cab en un servidor Web, en un espacio compartido de red o en otra carpeta. El nuevo asistente incorpora soporte técnico de acceso a datos para ADO, OLE DB, RDO, ODBC y DAO, así como compatibilidad con las nuevas aplicaciones para IIS y DHTML.
- ***Asistente de objetos de datos.*** Automatiza la creación de objetos de la capa media enlazados al entorno de datos o a controles de usuario.
- ***Mejoras del Asistente para formulario de datos.*** Ofrece la posibilidad de generar formularios de sólo código donde los controles no están enlazados a un control de datos. Permite usar código ADO.
- ***Mejoras del Asistente para aplicaciones.*** Ahora permite guardar sus configuraciones como perfiles para sus uso posterior de forma que pueda crear múltiples aplicaciones con el mismo formato.
- ***Diseñador de complementos.*** Permite iniciar el desarrollo fácilmente al especificar el comportamiento predeterminado de la carga de complementos, el nombre, destino versión de la aplicación, así como otras propiedades si lo desea.
- ***Mejoras del programa Generador de clases.*** Admite los valores *ParamArray, Optional, ByVal* y *Default* en la lista de argumentos y también admite enumeraciones.
- ***Asistente para barra de herramientas.*** Cuando tiene cargado el Asistente para aplicaciones, se abre automáticamente el Asistente para barra de herramientas al agregar una barra de herramientas a un formulario de forma que pueda crear barras de herramientas personalizadas.

Controles Active X. Visual Basic 6 contiene más de 60 controles ActiveX que puede usar para crear sus aplicaciones.

Capacidad de completar automáticamente el código. Visual Basic hace que sea mucho más fácil escribir código con características que permiten completar automáticamente instrucciones, propiedades y argumentos. Al

escribir el código, el editor de código que incorpora Visual Basic muestra una lista en la cual se puede seleccionar la orden adecuada.

Generar aplicaciones basadas en datos con Visual Basic

Esta nueva versión proporciona nuevas mejoras que facilitan el acceso a datos con Visual Basic. Éstas son:

- ***ADO 2.0.*** Esta nueva versión tiene un modelo de objetos más sencillo, mejor integración con las tecnologías de Microsoft y con otras tecnologías, una interfaz común para acceso a datos locales y remotos, recordsets remotos y desconectados, una interfaz de enlace de datos accesible para el usuario y recordsets jerárquicos.
- ***Diseñador de Entorno de datos.*** Este diseñador ayuda a crear y manipular visualmente conexiones y comandos ADO en tiempo de diseño.
- ***Ventana Vista Datos.*** Esta ventana contiene una lista de todas las conexiones de bases de datos disponibles actualmente para su proyecto. Puede ver la estructura completa de la base de datos mediante esta ventana.
- ***Integración de Visual Database Tools.*** Proporciona una interfaz visual para ayudarle a crear y modificar estructuras de bases de datos y consultas.

Si quiere obtener una explicación más detallada de las novedades y nuevas funciones de Visual Basic 6 consulte la librería de ayuda que incorpora el propio Visual Basic 6.

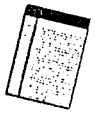
2. Instalación de Visual Basic 6

En este capítulo explicaremos los requisitos necesarios en el equipo donde se realiza la instalación, los pasos a seguir para instalar Visual Basic 6, cómo agregar o quitar componentes instalados y cómo obtener ayuda.

Requisitos del Sistema y del Hardware

Si quiere utilizar el programa Visual Basic 6.0 en su ordenador, este deberá cumplir los siguientes requisitos:

- Pc con un procesador 486 DX/66 MHz o superior, se recomienda procesador Pentium o superior.
- Microsoft Windows 95 o posterior, o Microsoft Windows NT versión 4.0 con Service Pack 3 o posterior.
- 16 MB de RAM para Windows 95 o posterior (se recomienda 32 MB); 24 MB para Windows NT 4.0 (se recomienda 32 MB).
- Microsoft Internet Explorer 4.01 Service Pack 1.



No es necesario disponer de Microsoft Internet Explorer 4.01 Service Pack 1, si no desea utilizar las ayudas y complementos disponibles en Internet.

- Espacio necesario en disco:

Edición Estándar:	- Instalación típica 48 MB.
	- Instalación completa 80 MB.
Edición Profesional:	- Instalación típica 48 MB.
	- Instalación completa 80 MB.
Edición Empresarial:	- Instalación típica 128 MB.
	- Instalación completa 147 MB.

- Se requiere espacio en disco adicional para la instalación de los siguientes productos:

- Internet Explorer:	43 MB típica.
	59 MB máxima.
- MSDN:	57 MB típica.
	493 MB máxima.
- Windows NT 4.0 Option Pack:	20 MB Windows 95 o posterior.
	200 MB Windows NT 4.0.

- Unidad de CD-ROM.
- Monitor VGA o de resolución superior; se recomienda Super VGA.
- Microsoft Mouse o compatible.

Instalación de Visual Basic

Visual Basic se instala en su equipo mediante un programa de instalación. Este programa instala Visual Basic y otros componentes del producto desde el CD-ROM al disco duro de su ordenador. También instala los archivos necesarios para mostrar la documentación del CD de MSDN. Si así lo decide, puede instalar tan sólo la documentación y los ejemplos de Visual Basic en su ordenador.

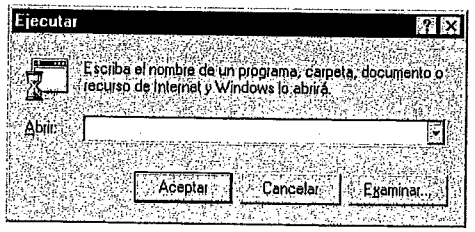
No puede copiar los archivos del CD-ROM en su disco duro y ejecutar Visual Basic. Deberá utilizar el programa de instalación para descomprimir e instalar los archivos en los directorios apropiados.

Para efectuar la instalación realice los siguientes pasos:

1. *Inserte el disco en la unidad de CD-ROM.*

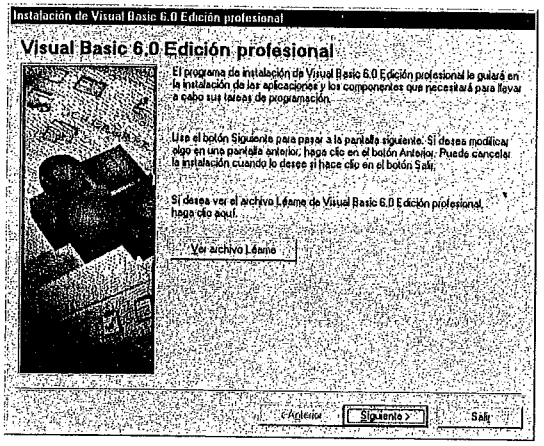
Si tiene activada la opción AutoPlay del CD-ROM, el programa de instalación se ejecutará automáticamente. De no ser así siga los pasos que a continuación se detallan:

1. *Haga clic sobre* [Inicio] *de la barra de tareas y seleccione la opción* [Ejecutar].

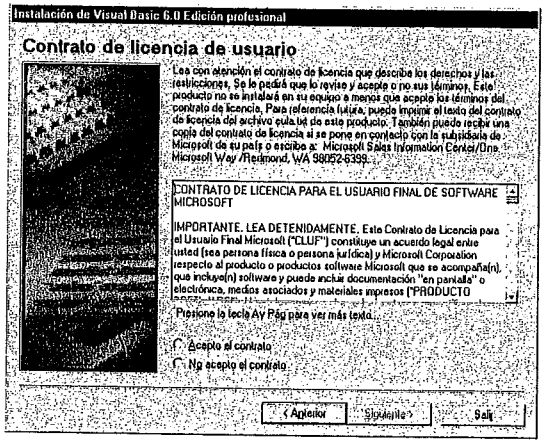


2. *Teclee **D:\Setup.exe** (si su unidad de CD-ROM no es la letra **D**, reemplácela por la correspondiente).*
3. *Efectúe un clic sobre el botón* [Aceptar].

Al cabo de unos instantes aparecerá la pantalla de instalación indicándonos la versión de Visual Basic que va a instalar, así como una explicación de cómo funciona.



2. *Haga clic sobre el botón* [Siguiente >] *para pasar a la siguiente pantalla.*



La siguiente pantalla muestra el contrato de licencia de usuario así como dos opciones a elegir:

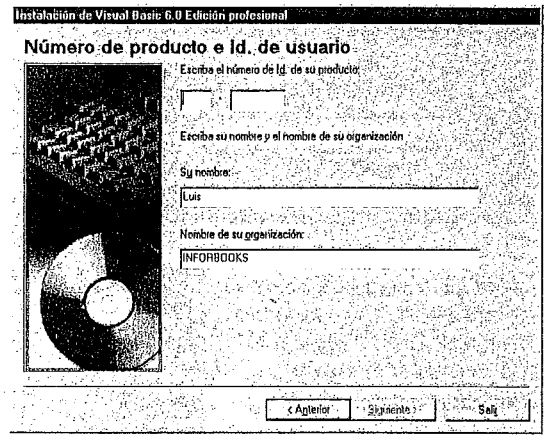
- **Acepto el contrato.**
- **No acepto el contrato.**

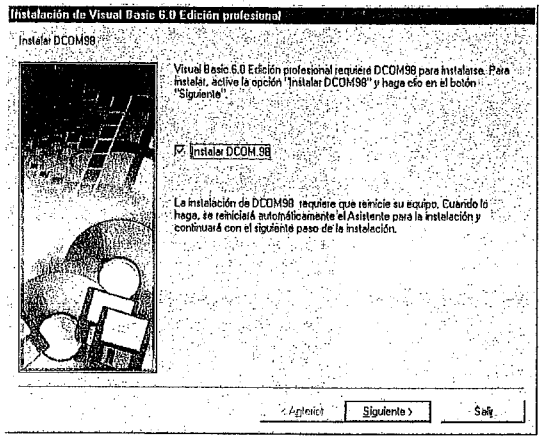
3. *Seleccione **Acepto el contrato** y pulse sobre el botón* [Siguiente >].

Si no desea aceptar las condiciones del contrato haga clic sobre el botón [Salir].

La tercera pantalla solicitará el número de identificación. Este número lo encontrará en el exterior de la caja del CD-ROM.

4. *Teclee el código que figure en la caja del CD-ROM y haga clic sobre* [Siguiente >].





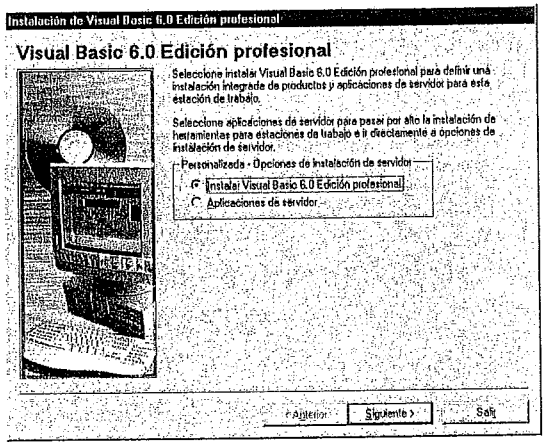
Si es la primera vez que instala Visual Basic le aparecerá la pantalla de la izquierda.

5. *Deje la opción que hay activada y pulse el botón* [Siguiente >].

Una vez copiados los archivos DCOM el ordenador debe reiniciarse para poder continuar con la instalación.

*El archivo **DCOM** es un archivo ejecutable de extracción automática que instala los componentes **DCOM** actualizados para Windows 95 o Windows 98.*

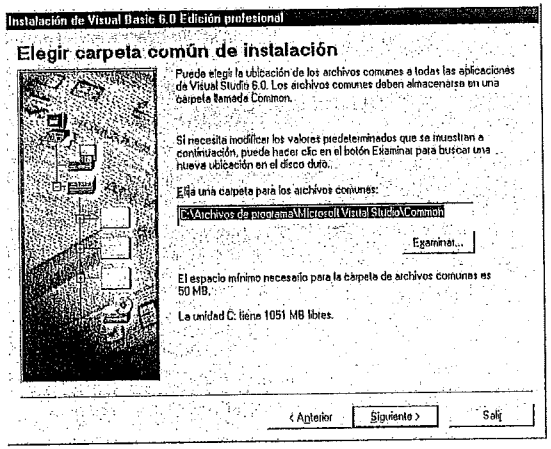
6. *Pulse el botón* [Aceptar].



Una vez se ha vuelto a reiniciar el ordenador aparecerá la pantalla de la izquierda donde podrá seleccionar el tipo de instalación que desea tener en su equipo.

7. *Deje activada la opción que aparece por defecto y pulse el botón* [Siguiente >].

La siguiente pantalla le permite especificar la carpeta donde desea que se instale Visual Basic.



8. *Haga un clic en* [Siguiente >] *para dejar la carpeta que aparece seleccionada por defecto.*

O bien:

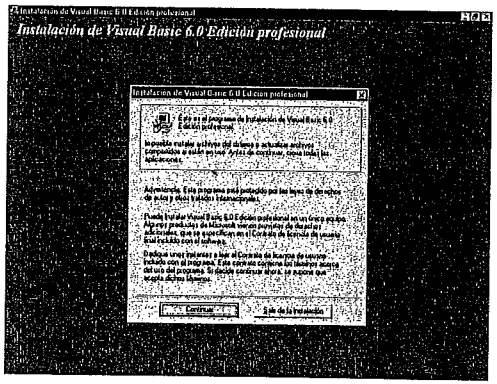
8. *Si prefiere especificar otra carpeta puede teclear directamente el nombre de la carpeta en el cuadro de texto que aparece resaltado o hacer clic sobre el botón* [Examinar...] *y buscar dicha carpeta.*

Una vez seleccionada la nueva carpeta deberá pulsar el botón [Siguiente >] *para continuar con la instalación de Visual Basic.*

A continuación comenzará la instalación de Visual Basic.

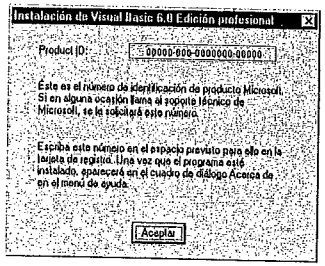
INSTITUTO DE EDUCACIÓN SECUNDARIA "ARROYO HARINA" ALMENDRALEJO - (Badajoz)

Esta pantalla se mostrará durante unos segundos mientras se inicia la instalación de Visual Basic.



Esta pantalla nos advierte que si tenemos alguna otra aplicación activa no se podrá instalar Visual Basic. Si tiene alguna otra aplicación deberá cerrarla antes de continuar con la instalación.

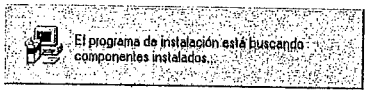
9. *Pulse en el botón* [Continuar] *para pasar a la siguiente pantalla.*



Esta pantalla muestra el número de identificación del producto que deberemos anotar por si deseamos realizar con posterioridad alguna consulta al soporte técnico de Microsoft.

10. *Haga clic en el botón* [Aceptar].

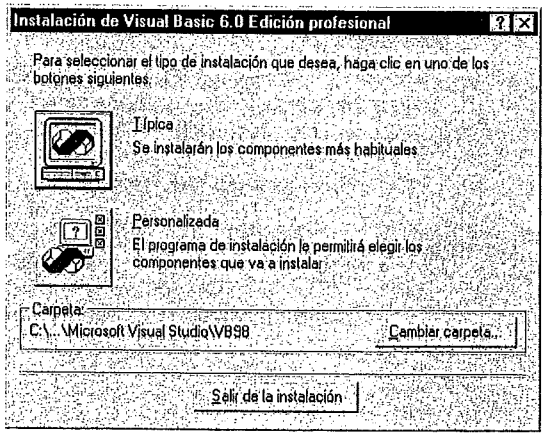
Durante unos segundos el programa de instalación examinará los componentes que hay instalados en su equipo.



En la siguiente pantalla podrá elegir el tipo de instalación que desea realizar en su equipo:

- **Típica:** Instala los componentes más habituales.
- **Personalizada:** Permite elegir los componentes que se van a instalar.

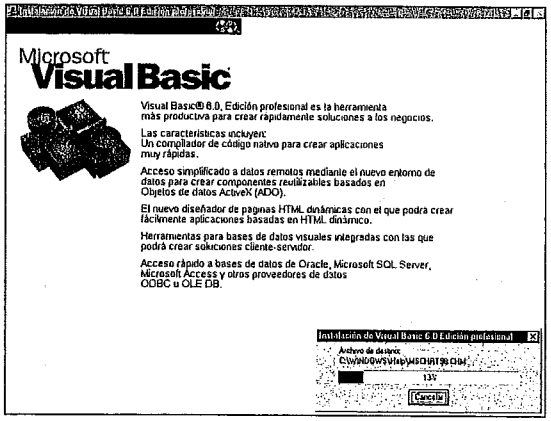
Si desea que Visual Basic se instale en otra carpeta que no sea la predeterminada por el programa haga clic en [Cambiar carpeta...].

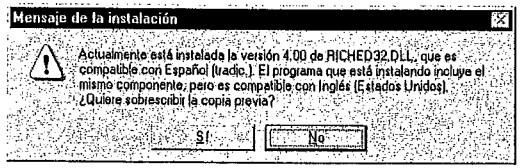


11. *Seleccione la instalación* ***Típica*** *pulsando sobre el botón que aparece a la izquierda.*

Una vez seleccionado el tipo de instalación, aparece en la parte inferior derecha una nueva ventana indicando el tanto por ciento de aplicación que se lleva instalado.

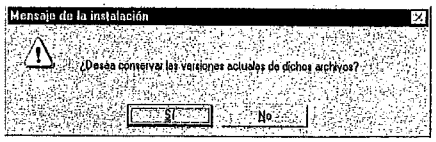
Pudiera ser que si con anterioridad tenía instalada alguna aplicación que utiliza parte de los componentes de Visual Basic, apareciese una nueva ventana indicándonos que la aplicación que estamos instalando también lo incluye pero en versión inglesa. Entonces el programa de instalación preguntará si desea reemplazar el componente que ya tiene instalado por esta nueva versión.





12. *Haga clic en el botón* [No].

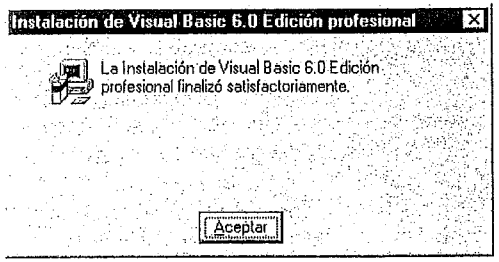
A continuación aparecerá una nueva ventana preguntando si desea conservar las versiones actuales de los archivos que en la ventana anterior preguntaba si quería reemplazar.



13. *Haga clic sobre el botón* [Sí].

Una vez finalice la instalación de Visual Basic y dependiendo de si tenía o no instalada con anterioridad otra versión de Visual Basic, aparecerán dos tipos de ventanas indicando que la instalación ha finalizado.

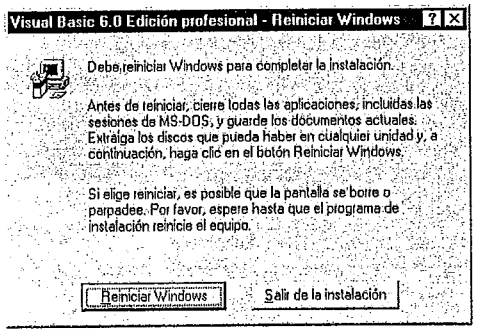
A continuación mostramos las dos ventanas que pueden aparecer:



La figura de la izquierda muestra la ventana que aparece si con anterioridad ya disponía en su equipo de alguna otra versión de Visual Basic.

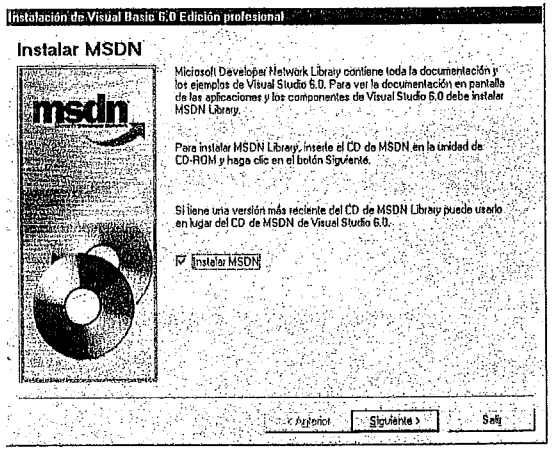
14. *Haga clic en el botón* [Aceptar].

En cambio si es la primera vez que instala Visual Basic en su equipo, aparecerá una ventana como la figura de la derecha.



14. *Haga clic sobre el botón* [Reiniciar Windows].

La pantalla que aparece a continuación permite la instalación de **MSDN Library** que contiene toda la documentación y los ejemplos de Visual Studio 6.0.

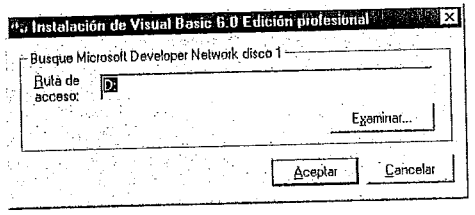


Esta nueva versión de **MSDN Library** para Visual Studio 6.0 es la primera versión que utiliza la tecnología del visor de Ayuda HTML.

15. *Deje activada la opción* ***Instalar MSDN*** *y haga clic en* [Siguiente >].

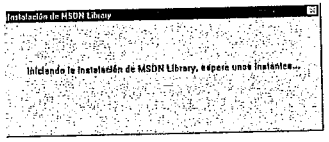
A continuación deberá extraer de la unidad de CD-ROM el CD de Visual Basic e introducir el CD etiquetado como número 1 de MSDN Library.

La ventana que nos solicita el cambio de CD se muestra en la siguiente figura:



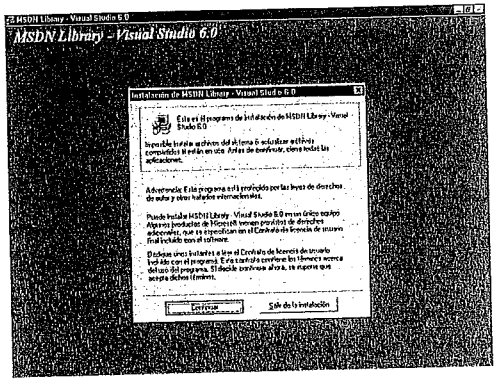
16. *Haga clic sobre el botón* [Aceptar].

En unos instantes se iniciará la instalación de la aplicación MSDN Library.

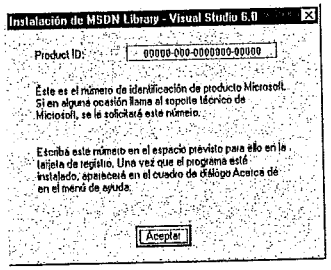


De nuevo aparecerá una ventana donde nos advierte que si tenemos alguna otra aplicación en uso no se podrá instalar MSDN Library. Si tiene alguna otra aplicación deberá cerrarla antes de continuar con la instalación.

17. *Pulse en el botón* [Continuar] *para pasar a la siguiente pantalla.*

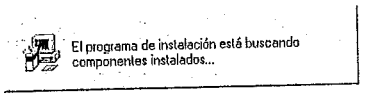


Esta pantalla muestra el número de identificación del producto que deberemos anotar por si deseamos realizar con posterioridad alguna consulta al soporte técnico de Microsoft.

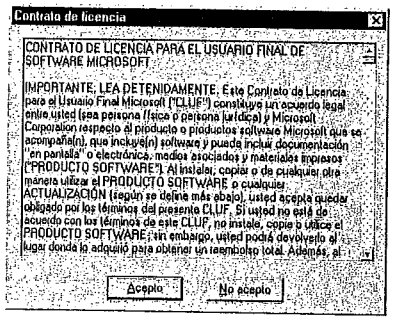


18. *Haga clic en el botón* [Aceptar].

Durante unos segundos el programa de instalación examinará los componentes que hay instalados en su equipo.



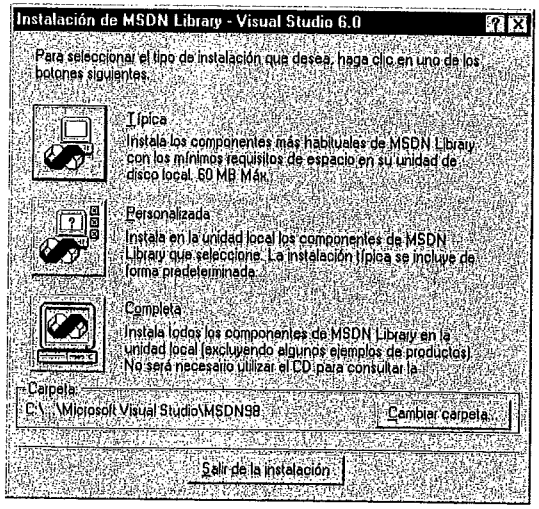
En la siguiente pantalla se muestra el contrato de licencia de usuario.



19. Haga clic en el botón [Acepto].

Si por el contrario no desea aceptar las condiciones del contrato haga clic en el botón [No acepto].

A la hora de seleccionar el tipo de instalación podrá escoger entre:

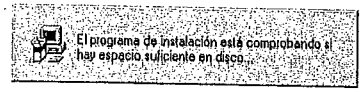


- **Típica:** Permite ejecutar MSDN Library desde el CD. Sólo se copiará el conjunto mínimo de archivos en el disco duro
- **Personalizada:** Permite especificar la instalación local de un subconjunto de MSDN Library.

  Se copiarán en el disco duro los archivos que seleccione y los archivos de la instalación Típica. Podrá tener acceso a la documentación de los productos sin tener que buscar en el CD de MSDN Library.
- **Completa:** Permite ejecutar MSDN Library casi exclusivamente desde el disco duro. El programa de instalación copiará toda la documentación y algunos archivos de ejemplo en el disco duro.

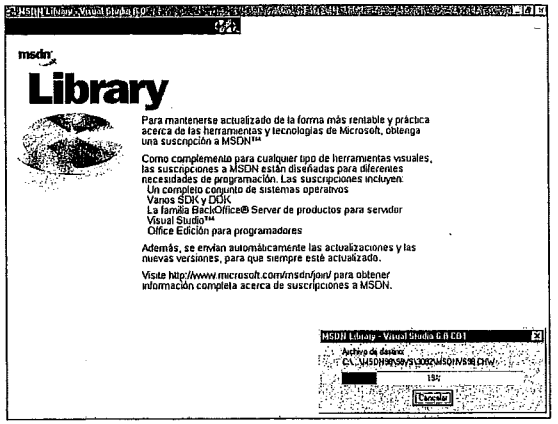
20. Seleccione el tipo de instalación que crea más adecuado pulsando sobre cualquiera de los botones de la izquierda.

Si desea que Visual Basic se instale en otra carpeta que no sea la predeterminada por el programa haga clic en [Cambiar carpeta...].

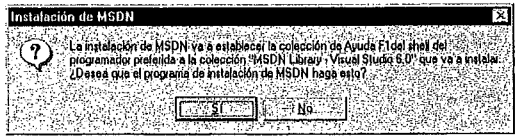


Una vez haya seleccionado el tipo de instalación, el ordenador comprobará si dispone de espacio suficiente para continuar la instalación.

Una vez seleccionado el tipo de instalación, aparece en la parte inferior derecha una nueva ventana indicando el tanto por ciento de aplicación que se lleva instalado.

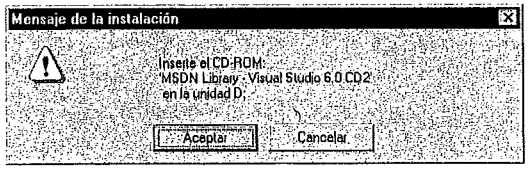


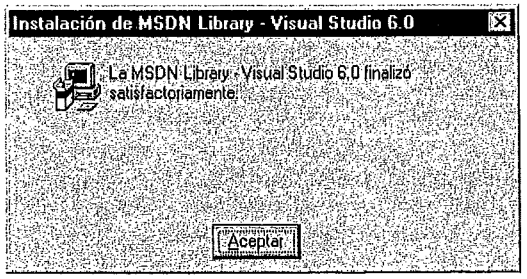
A continuación el programa preguntará si deseamos que por medio de la tecla de función **F1** se pueda obtener ayuda cuando esté en el entorno de Visual Basic y así poder acceder a MSDN Library.



21. Responda afirmativamente pulsando sobre el botón [Sí].

Si ha elegido una instalación completa, en un momento dado aparecerá la ventana de la derecha solicitándole que introduzca el CD número 2 de MSDN Library y a continuación pulse la tecla [Aceptar].



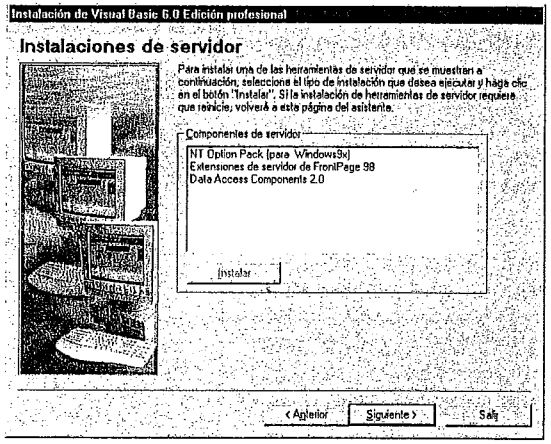


La figura de la izquierda muestra la ventana que aparece una vez que la instalación de MSDN Library ha finalizado satisfactoriamente.

22. Haga clic sobre el botón [Aceptar].

La siguiente ventana permite seleccionar la instalación de componentes de servidor. Si desea instalar alguno de ellos deberá seleccionarlos y pulsar el botón [Instalar].

23. *Haga clic con el ratón en* [Siguiente >].



La última ventana que aparece permite registrar su programa de Visual Basic 6 a través de la Web.

Si deja activada la opción **Registrarse ahora** que aparece por defecto activada, se conectará con el área de usuarios registrados donde podrá encontrar información actualizada de todos los productos de Visual Basic 6.

24. *Realice un clic sobre* [Finalizar].

Si dejó activada la opción **Registrarse ahora** entrará en la Web y el programa de instalación finalizará. De no ser así el programa también finalizará pero sin entrar en la Web.

*Si usted ha instalado MSDN Library con instalación **Típica**, cuando desde Visual Basic solicite ayuda con la tecla **F1**, la aplicación le informará que debe introducir el CD-ROM etiquetado como disco 2 de MSDN Library si quiere obtener ayuda.*

*Si no quiere tener el CD de MSDN en la unidad de CD-ROM cuando solicite ayuda, deberá realizar una instalación **Completa** o una instalación **Personalizada** habiendo seleccionado con anterioridad las siguientes casillas:*

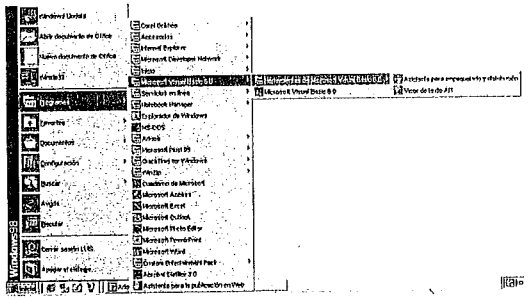
- Documentación de VB.

- Documentación compartida de VS.

- Documentación de VSS.

Programas instalados

Una vez finalizada la instalación las aplicaciones disponibles dentro de la carpeta de Visual Basic son:



- **Herramientas de Microsoft Visual Basic 6.0.** Estas herramientas están formadas por:
  - *Asistente para empaquetado y distribución.* El Asistente de empaquetado y distribución es una herramienta que le ayuda a crear paquetes de instalación para sus aplicaciones de Visual Basic e instalarlas en los equipos de usuarios finales. También puede usar el Asistente de empaquetado y distribución para crear archivos de dependencias (.dep).

Es necesario haber guardado y compilado el proyecto antes de poder ejecutar este asistente.

El Asistente de empaquetado y distribución ofrece numerosas opciones para la distribución de su aplicación, incluyendo la creación de:

- Paquetes estándar, diseñados para ser instalados mediante un programa de instalación (setup.exe).
- Paquetes de Internet, diseñados para ser descargados desde un sitio Web.
- Archivos de dependencia que puede distribuir con sus componentes.

  - *Visor de texto API.* El Visor de API le permite explorar las declaraciones, constantes y tipos incluidos en un archivo de API de texto o en una base de datos de Jet. Es posible copiar elementos al Portapapeles y, por tanto, copiarlos al código de Visual Basic.

- **Microsoft Visual Basic 6.0.** Ejecuta la aplicación de Visual Basic con todas las características que se han explicado en capítulos anteriores.

Agregar o quitar componentes de Visual Basic

Puede ejecutar el programa de instalación tantas veces como le sea necesario.

Si desea agregar o quitar componentes deberá realizar los siguientes pasos:

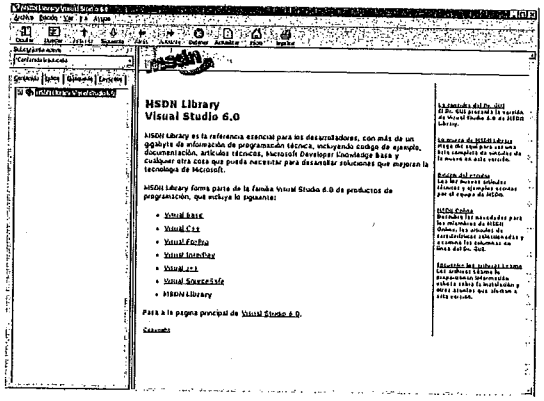
1. *Inserte el disco en la unidad de CD-ROM.*
2. *Haga clic en el botón **Personalizada** del cuadro de diálogo **Instalación de Microsoft Visual Basic 6.0.***
3. *Seleccione los componentes que va a instalar o anule los componentes que desea desinstalar en el cuadro de lista **Opciones** del cuadro de diálogo **Personalizada**.*
4. *Siga las instrucciones que aparecen en pantalla.*

Cómo obtener ayuda

Visual Basic dispone de diferentes formas para obtener ayuda mientras se programa. Esta ayuda consta de:

- **Contenido.** Esta opción muestra toda la documentación de Visual Basic en una misma ventana. Para acceder a esta opción realice los siguientes pasos:

  1. *Abra el menú **Ayuda**.*
  2. *Seleccione la opción **Contenido**.*

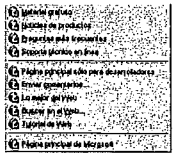


- **Índice.** Esta es la típica ayuda disponible en la mayoría de aplicaciones. Esta opción se utiliza si busca unos temas en concreto.

- **Búsqueda.** Permite buscar una palabra, funcion, método, etc. de cualquiera de los temas, es decir, en toda la ayuda.

- **Soporte técnico.** Si tiene la necesidad de hacer alguna pregunta técnica sobre Microsoft Visual Basic, consulte en primer lugar la documentación en pantalla. También encontrará información técnica y de última hora en el archivo *Léame* incluido en los CD-ROM del producto.

  Si no encuentra la respuesta, Microsoft ofrece soporte técnico y servicios que van desde servicios gratuitos de información en línea hasta planes anuales de soporte con un ingeniero técnico de Microsoft.



- **Microsoft en el Web.** Permite obtener ayuda en las páginas Web de Microsoft siempre y cuando disponga de una conexión a Internet.

- **Ejecutar ejemplos de código.** Siempre que solicite ayuda sobre una función, evento, propiedad, etc., Visual Basic mostrará una pantalla de ayuda, en la que aparecerá una opción denominada **Ejemplo**. Esta opción le mostrará un pequeño programa relacionado con el tema que ha seleccionado, este ejemplo lo puede copiar y seguidamente pegar en un proyecto de Visual.

- **Obtener ayuda interactiva desde el entorno.** Visual Basic dispone de ayudas interactivas, es decir, se obtiene ayuda directamente sin tener que acceder al menú de ayuda; esto se consigue pulsando la tecla **F1**.

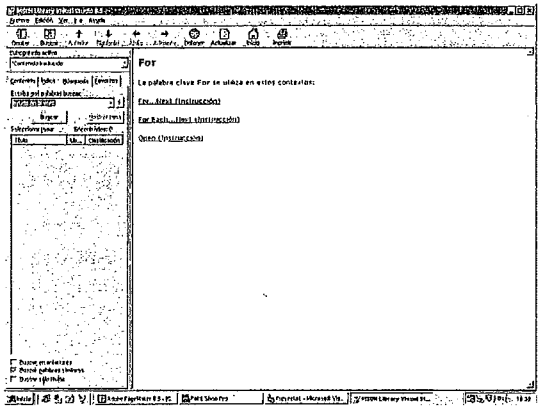
  La tecla **F1** se puede pulsar cuando el cursor esté situado en las siguientes partes de una aplicación:

  - *Todas las ventanas de la aplicación de Visual Basic.*
  - *Los controles que forman el cuadro de Herramientas.*
  - *Todos los objetos de los formularios.*
  - *Las propiedades que se incluyen en la ventana de Propiedades.*
  - *Procedimientos de la ventana de Código.*
  - *Instrucciones, funciones, propiedades, métodos, etc.*

  Realice los siguientes pasos:

1. *Abra la ventana de código realizando un doble clic sobre el formulario que tenga en pantalla.*

2. *Teclee la palabra **For** y pulse la tecla **F1**.*



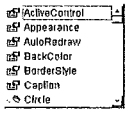
Observe como en el interior de la ventana se muestra información sobre la palabra que ha tecleado.

Por último y para terminar disponemos de una ayuda que nos será de mucha utilidad a la hora de escribir código. Esta ayuda nos informará sobre las propiedades, métodos, eventos, etc. de cualquier objeto que estemos utilizando.

Efectúe estos pasos para ver el funcionamiento de esta ayuda:

1. *Abra la ventana de código.*

2. *Teclee en su interior **Form1**..*



A continuación Visual Basic mostrará la ventana de la izquierda. Observe como esta ventana incluye todas las propiedades del objeto tecleado, en este caso las propiedades del formulario.

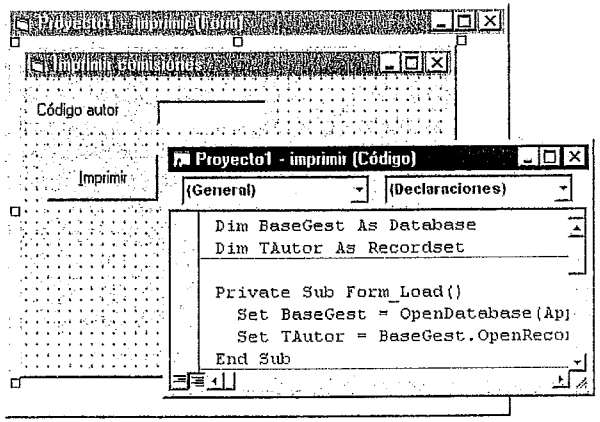
3. Fundamentos de Programación

En este capítulo veremos los componentes básicos del lenguaje Visual Basic, principalmente los relacionados con el código que determine el comportamiento de la aplicación profundizando en la declaración y tipos de datos, así como las estructuras de control, es decir, conceptos que tiene que conocer antes de empezar a diseñar formularios, módulos, etc.

Estructura de una aplicación en Visual Basic

La estructura de una aplicación es la forma en que se organizan las instrucciones, es decir, dónde se almacenan las instrucciones y el orden en que han de ser ejecutadas.

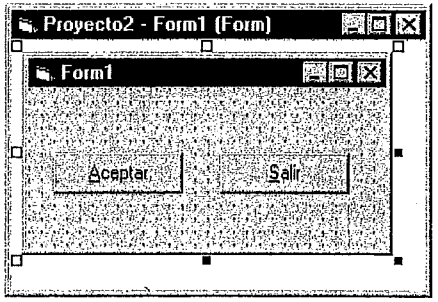
Podemos definir una aplicación como el conjunto de instrucciones que se le da a un equipo (ordenador) para que realice una o varias tareas.



Un aspecto importante a la hora de realizar una aplicación es saber estructurarla u organizarla de modo que sea fácil para el programador encontrar las partes del programa que desea consultar o modificar.

Una aplicación de Visual Basic se basa en objetos, la estructura de su código se

aproxima mucho a su representación física en la pantalla. De modo que por cada formulario hay un módulo relacionado que contiene su código.



Cada módulo de formulario contiene secciones de código (también llamados *procedimientos de evento*) donde se colocan las instrucciones que se ejecutarán cuando se produzca un determinado evento.

Por cada control de un formulario, debe existir el correspondiente conjunto de procedimientos de evento en el módulo de formulario. Si por ejemplo tenemos un formulario con los botones (controles) de *Aceptar* y *Salir*, deberá existir el código correspondiente relacionado con cada uno de los botones.

Además de procedimientos de evento, los módulos de formulario pueden contener procedimientos generales que se ejecutan como respuesta a una llamada desde cualquier procedimiento de evento.

Aplicaciones controladas por eventos

Las aplicaciones controladas por eventos ejecutan el código de Visual Basic que tengan asociado a ese evento. Si bien Visual Basic reconoce automáticamente un conjunto de eventos predefinidos (como por ejemplo el evento *Click* con el ratón) es usted quien decide cómo y cuándo se responderá a ese evento escribiendo el código relacionado con ese evento.

La secuencia que sigue una aplicación controlada por eventos es la siguiente:

1. Se inicia la aplicación, se carga y muestra un formulario.
2. El formulario o un control de éste (por ejemplo un botón) espera a recibir un evento. Este evento puede estar causado por:
   - El usuario (realizando un clic con el ratón sobre un botón).
   - El sistema (si es una fecha determinada).

   - Por el código de forma indirecta (carga de un formulario).
3. Si hay código asociado con ese evento, se ejecuta.
4. La aplicación se queda en espera hasta recibir otro evento.

Consideraciones a tener en cuenta antes de codificar

La parte más importante en la creación de una aplicación en Visual Basic es la fase de diseño. La forma en que estructure una aplicación puede hacer que varíe su funcionamiento, así como el mantenimiento y la facilidad de uso del código.

El código de una aplicación de Visual Basic está organizado jerárquicamente. Una aplicación típica está formada por uno o más módulos y cada módulo contiene a su vez uno o más procedimientos que contienen código.

Además del diseño de la estructura del código, es importante establecer convenciones de nomenclatura.

Tipos de módulos

En Visual Basic existen tres tipos de módulos:

llevan códigos y objetos

- ***Módulos de formulario***. Son la base de la mayoría de las aplicaciones de Visual Basic. Pueden contener procedimientos que controlen eventos, procedimientos generales y declaraciones a nivel de formulario de variables, constantes, tipos y procedimientos externos.

No tienen controles solamente de código.

- ***Módulos estándar***. Son contenedores de los procedimientos y declaraciones a los que tienen acceso otros módulos de la aplicación. Pueden contener declaraciones globales o a nivel de módulo de variables, constantes, tipos, procedimientos externos y procedimientos globales.

No es buen utilizarlos mucho.

- ***Módulos de clase***. Son la base de la programación orientada a objetos (OOP) en Visual Basic. Estos objetos nuevos pueden incluir propiedades y métodos personalizados.

Parametro por valor → parametro de entrada
" " referencia → " " salida
Parametro actual → el que está actualmente (principal)
" formal →

Conceptos básicos del código

En este apartado explicaremos algunos conceptos básicos que se han de tener en cuenta a la hora de escribir el código de un programa.

Introducir comentarios en el código

Un comentario es una anotación muy útil en un programa ya sea corto o extenso. Servirá para realizar una pequeña explicación de lo que hace ese trozo de código de tal manera que si pasado un tiempo se desea volver a examinar ese programa ya no hará falta *adivinar* la tarea que realiza esa parte del código.

Siempre que desee introducir un comentario en un programa deberá utilizar el símbolo (') que se encuentra situado debajo de la tecla de función **F9**.

Los comentarios pueden ocupar una o más líneas completas siempre y cuando comiencen con el símbolo ('). Pero también pueden aparecer en la misma línea de código, siempre y cuando aparezcan después del código y estén precedidos del símbolo (').

Veamos algunos ejemplos:

Comentario Válido

a)

```
'z almacena el valor de x + y
z = x + y
```

b)

```
z = x + y  'z almacena el
           'valor de x + y
```

Comentario Inválido

a)

```
z almacena el valor de x + y
z = x + y
```

b)

```
z = x + y    z almacena el
             'valor de x + y
```

*Siempre que inicie un comentario con el símbolo (') el texto que aparezca a continuación lo hará en color verde una vez que haya pulsado la tecla **Retorno**.*

Combinar instrucciones en una línea

Normalmente se suele escribir una instrucción en cada línea de código, pero si quiere puede colocar más de una instrucción por línea, para ello deberá utilizar el símbolo de dos puntos (:) entre cada una de las instrucciones.

Veamos un ejemplo:

Código en más de una línea:

```
Color = 255
Mensaje1.Text = "Hola"
Mensaje1.FontSize = 25
```

Código en una línea:

```
Color = 255 : Mensaje1.Text = "Hola" : Mensaje1.FontSize = 25
```

Dividir una instrucción en varias líneas

Puede darse el caso de que el código de la instrucción que está escribiendo sea muy largo y desee dividirlo en varias líneas, para ello se utilizará un espacio en blanco seguido del carácter de continuación de línea (_), de este modo el programa no devolverá un error cuando procese esta línea de código.

Veamos un ejemplo:

```
"SELECT * FROM Agenda, Socios" _
& "WHERE Agenda.Cod = Socios.Cod" _
& "AND Agenda.Provincia = 'Barcelona' "
```

Nunca se puede poner un comentario después de un carácter de continuación (_) de línea en la misma línea.

Reglas de nomenclatura en Visual Basic

Siempre que declare procedimientos, variables y constantes en Visual Basic deberá seguir una serie de reglas:

- Deben comenzar por una letra.
- No pueden contener puntos o caracteres de declaración de tipos.
- No pueden superar los 255 caracteres.
- Los nombres de controles, formularios, clases y módulos no excederán los 40 caracteres.
- No pueden tener el mismo nombre de una palabra clave.

Declaración de variables

Cuando se programa, a menudo es necesario almacenar valores temporalmente para realizar cálculos, comparaciones, etc. para ello utilizaremos las variables.

La sintaxis para asignar un valor a una variable es la siguiente:

Nombre de Variable = Valor de la variable

Veamos algunos ejemplos:

```
Nombre = "Juan"          'Asigna el valor Juan a la variable Nombre
Edad = 30                'Asigna el valor 30 a la variable Edad
Total = Total + 100      'Incrementa el valor de Total en 100
```

En Visual Basic la declaración de una variable se divide en:

- ***Nombre de Variable.*** Etiqueta que identifica el lugar donde se almacena el valor de la variable.
- ***Tipo de dato.*** Determina el tipo de datos que puede almacenar la variable.

Para declarar una variable en Visual Basic es necesario utilizar la palabra ***Dim*** seguida del nombre/s de variable/s y del tipo de dato. Su sintaxis es la siguiente:

Dim *Nombre de Variable* (**As** *tipo*)

Veamos algunos ejemplos:

```
Dim X As Integer            'Variable de tipo entero
Dim Y As Real, Z As Real    'Variables de tipo real
Dim Nombre As String        'Variable de tipo cadena
```

*Como habrá podido observar cuando declara una variable, la declaración de tipo es opcional (por eso aparece entre corchetes en la sintaxis [**As** tipo]). Si no especifica el tipo de variable el programa asume por defecto que es del tipo* Variant *y selecciona el tipo que cree más adecuado.*

Además del modo que hemos visto de declarar una variable, existen otras dos formas:

- ***Declaración implícita.*** Permite utilizar una variable sin haberla declarado, esto puede resultar cómodo, pero también conlleva errores, veamos un ejemplo:

Hay q. declarar al principio del pgrama

```
Function Raiz(Numero)
    Absoluto = Abs(Numero)
    Raiz = Sqr (Absolut)
End Function
```

A simple vista el código parece correcto, sin embargo se producirá un error que Visual Basic no detectará puesto que en la tercera línea hemos escrito *Absolut* en lugar de *Absoluto*.

Todo lo q. se utiliza hay q. declararlo

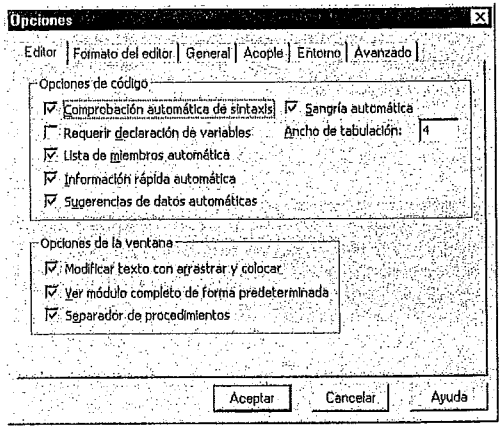
- ***Declaración explícita.*** Evita el problema de la declaración implícita, es decir, siempre que Visual Basic encuentre una variable que previamente no ha sido declarada provocará un mensaje de error.

Si utilizásemos el ejemplo anterior se produciría un error en la tercera línea puesto que la variable *Absolut* no ha sido declarada con anterioridad.

Para realizar una declaración explícita de variables existen dos métodos:

- Incluir la instrucción ***Option Explicit*** en la sección Declaraciones del módulo de clase, formulario o estándar.

- La otra forma sería realizando los pasos que se describen a continuación:



1. *Abra el menú **Herramientas** de Visual Basic y seleccione **Opciones**.*
2. *Seleccione la carpeta **Editor**.*
3. *Active la opción **Requerir declaración de variables**.*
4. *Pulse el botón* Aceptar *para finalizar.*

Alcance de las variables

Una variable puede tener distintos *alcances*, es decir, define qué partes del código son conscientes de su existencia. Así podemos distinguir entre:

- ***Variables locales***. Sólo son reconocidas en el procedimiento en que se han declarado. Para poder declararlas deberá usar la siguiente instrucción:

  Dim *Nombre de variable* (**As** *tipo*)

  También podrá utilizar la siguiente instrucción:

  Static *Nombre de variable* (**As** *tipo*)

IMPORTANTE!!

La diferencia entre una variable de tipo **Dim** *y tipo* Static *está en que las variables declaradas* **Static** *existen mientras se ejecuta la aplicación, las variables declaradas* **Dim** *sólo existen mientras se ejecuta el procedimiento.*

- ***Variables privadas***. Son reconocidas por todos los procedimientos de un mismo módulo. Para poder declararlas deberá usar la siguiente instrucción al principio del módulo.

  Private *Nombre de variable* (**As** *tipo*)

- ***Variables públicas***. Son reconocidas por todos los procedimientos de cualquier módulo. Para poder declararlas deberá usar la siguiente instrucción al principio del módulo.

  Public *Nombre de variable* (**As** *tipo*)

Si hay variables públicas en distintos módulos con el mismo nombre, es posible diferenciarlas en el código haciendo referencia al módulo y a los nombres de variable.

Supongamos que tenemos una variable pública (**Public**) de tipo entero (**Integer**) cuyo nombre es **Valor** y que está declarada en un formulario (**Form1**) y en un módulo (**Module1**). La forma de hacer referencia a cada una de ellas sería:

```
Form1.Valor        'Variable del formulario
Module1.Valor      'Variable del módulo
```

Aunque Visual Basic permita utilizar variables con el mismo nombre en distintos módulos, es recomendable declarar dichas variables con nombres diferentes para evitar confusiones.

Declaración de constantes

En todo código suelen aparecer valores constantes que se repiten una y otra vez, es decir, valores que por sí mismos no tienen un significado obvio.

Una constante es un nombre significativo que sustituye a un número o a una cadena que no varía.

IMPORTANTE!!

La diferencia que existe entre una variable y una constante radica en que la variable puede tomar diferentes valores mientras se ejecuta la aplicación, mientras que el valor de la constante permanece fijo.

Existen dos tipos de constantes:

- ***Constantes definidas por el sistema.*** Son proporcionadas por aplicaciones y controles. Las constantes de Visual Basic se muestran en Visual Basic, Visual Basic para Aplicaciones y las bibliotecas de objetos en el Examinador de objetos.

- ***Constantes definidas por el usuario.*** Se declaran mediante la instrucción ***Const.***

La sintaxis para asignar un valor a una constante es la siguiente:

(Public/Private) Const *Nombre de Constante* **(As** *tipo***)** = *Expresión*

Estos tipos de constantes reciben otro nombre, en el caso de las constantes definidas por el sistema son las denominadas **intrínsecas** *y las definidas por el usuario* **simbólicas.**

El argumento *Expresión* está compuesto por constantes y operadores de cadena o numéricos. Algunos ejemplos son:

```
Const Gravedad = 9.81                'Asigna el valor 9.81 a la constante Gravedad
Private Const Nombre = "Antonio"     'Asigna el valor Antonio a la constante Nombre
Public Const IVA = 0.16              'Asigna el valor 0.16 a la constante IVA
```

Una constante tiene el mismo alcance que una variable y se le aplican las mismas reglas.

- *Para crear una constante que sólo exista en un procedimiento, debe declararse dentro del procedimiento.*
- *Para crear una constante disponible para todos los procedimientos, pero no para el código que está fuera del módulo, debe declararse en la sección Declaraciones del módulo.*
- *Para crear una constante disponible en toda la aplicación, debe declarar la constante en la sección Declaraciones de un módulo estándar y colocar delante de* ***Const*** *la palabra* ***Public.***

Declaración de tipos

El tipo de dato de una variable indica cómo se almacenan los bits que representan esos valores en la memoria del ordenador. Visual Basic dispone de un tipo de datos predeterminado si no proporciona uno concreto, es el tipo ***Variant*** que puede representar diferentes tipos de datos en distintas situaciones.

Es aconsejable utilizar siempre un tipo de datos determinado, ya que Visual Basic tratará de forma más eficiente los datos si son declarados con un tipo específico que si no lo son.

Cuando declare una variable con un tipo de dato, deberá utilizar las instrucciones **Dim, Static, Private** o **Public** como vimos en el apartado correspondiente.

Tipos de datos numéricos

Visual Basic proporciona diferentes tipos de datos numéricos, pudiéndose distinguir los siguientes tipos:

- ***Integer.*** Variable entera de 2 bytes, su rango va desde -32.768 a +32.767. El carácter de declaración de tipo para este tipo de datos es el signo **%**.

- ***Long.*** Variable entera larga de 4 bytes, su rango va desde -2.147.483.648 a +2.147.483.647. El carácter de declaración de tipo para este tipo de datos es el signo **&**.

- ***Single.*** Variable real simple de 4 bytes, su rango va desde -3.40E+308 a +1.79E+308. El carácter de declaración tipo para este tipo de datos es el signo **!**.

- ***Double.*** Variable real de doble precisión de 8 bytes, su rango va desde -1,79769313486232E+308 a -4,94065645841247E-324 para valores negativos y de +4,94065645841247E-324 a 1,79769313486232E+308 para valores positivos. El carácter de declaración tipo para este tipo de datos es el signo **#**.

- ***Currency.*** Variable con punto decimal fijo de 8 bytes, su rango va desde -922.337.203.685.477,5808 a 337.203.685.477,5807. El carácter de declaración tipo para este tipo de datos es el signo **@**.

Veamos algunos ejemplos:

```
Dim x As Integer                 'Variable x tipo Integer
Public y As Long                 'Variable y tipo Long
Private b As Single c As Double  'Variable b tipo Single variable c tipo Double
```

```
Static z As Currency          'Variable z tipo Currency
```

Los valores de signo flotante se pueden expresar como **mmmEeee** *o* **mmmDeee**, *donde* **mmm** *es la mantisa y* **eee** *el exponente. Si utiliza la letra* **D** *para separar la mantisa del exponente, el valor será tratado como* **Double**, *en cambio si utiliza la letra* **E** *el valor será tratado como* **Single**.

Tipos de datos Byte

El tipo **Byte** se utilizará cuando la variable almacene datos binarios. Todos los operadores que funcionan con enteros son válidos con el tipo Byte excepto la resta unaria. Su rango va desde 0 a +255.

Tipos de datos String

Las variables de tipo **String** son cadenas de longitud variable, esto quiere decir que la longitud aumenta o disminuye según se le asignen nuevos datos. También se pueden definir variables de este tipo con longitud fija. Veamos algunos ejemplos:

```
Dim Nombre As String          'Variable Nombre con longitud variable
Dim Nombre As String * 20     'Variable Nombre con longitud fija 20 caracteres
```

Emplee este tipo de datos cuando utilice una variable que siempre contenga una cadena y nunca un valor numérico.

Tipos de datos Boolean

Utilice este tipo de datos cuando trabaje con variables que contengan sólo información con los valores *Verdadero* o *Falso, Sí* o *No, Activado* o *Desactivado*. El valor por defecto de una variable de este tipo es ***False***. A continuación veremos un ejemplo de como se usa este tipo de variable:

```
Dim x As Integer
```



```
Dim Continuar As Boolean
If x = 10 Then
    Continuar = True
End If
```

Tipos de datos Fecha

Utilice este tipo de datos cuando desee almacenar valores de fecha y hora. Cuando se convierten otros tipos de datos numéricos en **Date**, los valores que hay a la izquierda del signo decimal representan la información de fecha, mientras que los valores que hay a la derecha representan la hora.

Tipos de datos Object

Las variables **Object** se almacenan como direcciones de 4 bytes que hacen referencia a objetos dentro de una aplicación o de cualquier otra.

Tipos de datos Variant

Este tipo de variables es capaz de almacenar todos los tipos de datos definidos en el sistema. No tiene que convertir estos tipos de datos ya que Visual Basic realiza automáticamente el tipo de conversión necesaria.

Si realiza operaciones aritméticas o funciones sobre un **Variant** *procure que éste contenga un número.*

Si concatena cadenas, utilice el operador **&** *en lugar del operador* **+**.

Además de los valores estándar, las variables **Variant** pueden contener tres valores especiales:

- ***Empty.*** Es el valor que tiene una variable del tipo ***Variant*** antes de asignarle cualquier otro.
- ***Null.*** Este valor se utiliza en aplicaciones de bases de datos para indicar datos desconocidos o que faltan.
- ***Error.*** Se utiliza para indicar que se ha producido una condición de error en un procedimiento.

Se producirá un error si intenta realizar una operación o función sobre un tipo **Variant** que no contiene un número o algo que se puede interpretar como numérico.

Cuando Visual Basic convierte una representación que no es numérica (por ejemplo una cadena que contenga un número) a un valor numérico, utiliza la configuración regional para asignar el separador de miles, decimal y el símbolo monetario.

Introducción a los Procedimientos

Existe un aforismo en informática que dice ***"divide y vencerás"***, pues bien, para simplificar las tareas de programación los programas se dividirán en componentes lógicos más pequeños llamados *procedimientos*.

Los procedimientos resultan muy útiles para:

- Condensar las tareas repetitivas o compartidas.
- Cálculos utilizados frecuentemente.
- Manipulación de texto y controles.
- Operaciones con bases de datos.

Hay dos ventajas a destacar cuando se programa con procedimientos:

- Los procedimientos permiten dividir programas en unidades lógicas discretas, cada una de las cuales se puede depurar más fácilmente que un programa entero sin procedimientos.
- Los procedimientos que se utilizan en un programa pueden actuar como bloques de construcción de otros programas, normalmente con pocas o ninguna modificación.

Los nombres de las variables privadas y públicas a nivel de módulo también pueden entrar en conflicto con los nombres de procedimiento. Una variable del módulo no puede tener el mismo nombre que los procedimientos o tipos definidos en el módulo. Sin embargo, pueden tener el mismo nombre que los procedimientos, tipos o variables públicas definidas en otros módulos. En este caso, cuando se tiene acceso a la variable desde otro módulo, debe calificarse con el nombre del módulo.

En Visual Basic se utilizan varios tipos de procedimientos:

- Procedimientos **Sub**. void < eventos
- Procedimientos **Function**. → de función.
- Procedimientos **Property**. → de propiedades.

INSTITUTO DE EDUCACIÓN SECUNDARIA "ARROYO HARINA" ALMENDRALEJO (Badajoz)

Procedimientos Sub

Estos procedimientos se ejecutan como respuesta a un *evento* y no devuelven ningún valor. Su sintaxis es la siguiente:

(Private / Public) (Static) Sub *Nombre Procedimiento (argumentos)*

instrucciones

End Sub

Cada vez que se llama a un procedimiento de este tipo se ejecuta todo el código que haya dentro de *instrucciones*.

Para llamar a un procedimiento se utilizará la palabra **Call** siendo su sintaxis la siguiente:

Call *Nombre Procedimiento (argumentos)*

Dentro de este tipo de procedimiento podemos distinguir dos subtipos:

- **Procedimientos Generales.** Indican a la aplicación cómo realizar una tarea específica. Una vez definido, se le debe llamar específicamente desde la aplicación.
- **Procedimientos de Evento.** Permanecen inactivos hasta que son llamados para responder a eventos provocados por el usuario o desencadenados por el sistema.

Procedimientos Function

Estos procedimientos pueden devolver un valor al procedimiento que realiza la llamada. Su sintaxis es la siguiente:

(Private / Public) (Static) Function *Nombre Procedimiento (argumentos)* (As *tipo*)

instrucciones

End Function

Para llamar a un procedimiento de función se usa la misma forma en que se llama a una función intrínseca de Visual Basic, es decir, utilizando su nombre en una expresión. Su sintaxis es la siguiente:

Nombre Variable = Nombre Función (*argumentos*)

Existen tres diferencias entre los procedimientos **Sub** y **Function**:

- Generalmente se llama a una función incluyendo el nombre y los argumentos del procedimiento en la parte derecha de una instrucción o expresión.
- Los procedimientos **Function** tienen dos tipos de datos al igual que las variables.
- Se devuelve un valor asignándolo al propio nombre del procedimiento.

Procedimientos Property

Son procedimientos que pueden crear y manipular las propiedades para un módulo de clase. Un procedimiento de este tipo comienza con la instrucción **Property Let**, **Property Get** o **Property Set** y termina con la instrucción **End Property**.

Property Let

Declara el nombre, los argumentos y el código que forman el cuerpo de un procedimiento Property, el cual asigna un valor a una propiedad. Su sintaxis es:

(Private/Public/Friend) (Static) Property Let *Nombre* (*(argumentos,)* valor)

(instrucciones)

(Exit Property)

(instrucciones)

End Property

*Cualquier instrucción de este tipo debe definir por lo menos un argumento para el procedimiento que define. Ese argumento (o el último si hay más de uno) contiene el valor real que se asigna a la propiedad cuando se invoca el procedimiento definido por la instrucción **Property Let**.*

Property Get

Declara el nombre, los argumentos y el código que forman el cuerpo de un procedimiento Property, el cual obtiene el valor de una propiedad. Su sintaxis es:

(Private/Public/Friend) (Static) Property Get *Nombre* (*(argumentos)*) (As Tipo)

(instrucciones)

(nombre = expresión)

(Exit Property)

(instrucciones)

(nombre = expresión)

End Property

*La instrucción **Exit Property** causa la inmediata salida del procedimiento. La ejecución del programa continúa con la instrucción que sigue a la instrucción que llamó al procedimiento.*

Property Set

Declara el nombre, los argumentos y el código que componen el cuerpo de un procedimiento Property, el cual obtiene el valor de una propiedad. Su sintaxis es:

(Private/Public/Friend) (Static) Property Get *Nombre* (*(argumentos)*) (As Tipo)

(instrucciones)

(nombre = expresión)

(Exit Property)

(instrucciones)

(nombre = expresión)

End Property

Forzar la salida de un Procedimiento

A veces nos puede interesar forzar la salida de un procedimiento **Sub** o **Function** antes de que éste finalice, pues bien, para ello utilizaremos las siguientes sentencias:

Procedimiento	Instrucción de salida
Sub	Exit Sub
Function	Exit Function
Property	Exit Property

Pasar argumentos a Procedimientos

Normalmente el código de un procedimiento necesita cierta información sobre el estado del programa para realizar su trabajo. Esta información consiste en variables que se pasan al procedimiento cuando es llamado.

IMPORTANTE!!

*La variable que es pasada a un procedimiento recibe el nombre de **argumento**.*

Los argumentos pueden ser pasados de dos formas:

- **Pasar argumentos por valor.** Se pasa una copia del contenido de la variable que se pasa como argumento. Si el procedimiento cambia el valor, el cambio sólo afecta a la copia. Deberá utilizar la palabra **ByVal** para indicar que el argumento es pasado por valor.

  Veamos un ejemplo:

```
Dim A As Integer
A = 5                    'Valor inicial de la variable A
Prueba(A)                'Llamada al procedimiento con paso de argumento
Print A                  'Se imprime el valor 5

Sub Prueba (ByVal Num As Integer)    'Num toma el valor de A, o sea 5
    Num = Num + 1
    Print Num                        'Num toma el valor 6
End Sub
```


- **Pasar argumentos por referencia.** Da al procedimiento acceso al contenido real de la variable en su posición de memoria. Si el procedimiento cambia el valor este cambio si que afecta al contenido de la variable.

  Veamos un ejemplo:

```
Dim A As Integer
A = 5                    'Valor inicial de la variable A
Prueba(A)                'Llamada al procedimiento con paso de argumento
Print A                  'Se imprime el valor 6

Sub Prueba (Num As Integer)    'Num toma el valor de A, o sea 5
    Num = Num + 1
    Print Num                  'Num toma el valor 6
End Sub
```

*Si desea que un procedimiento pueda recibir argumentos opcionales, deberá colocar la palabra clave **Optional** delante de cada uno de los argumentos que desee que sean opcionales.*

*En cambio si quiere que el número de argumentos sea indefinido, deberá utilizar la palabra clave **ParamArray** delante de los argumentos.*

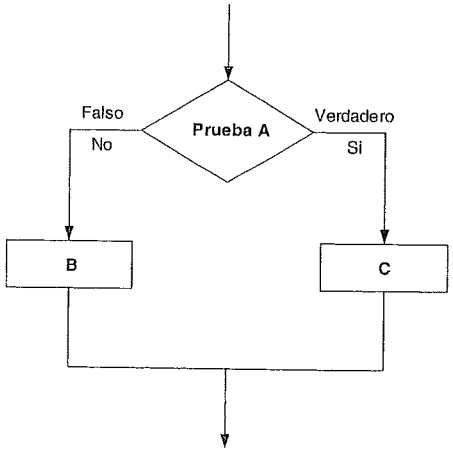
Estructuras de control de un programa

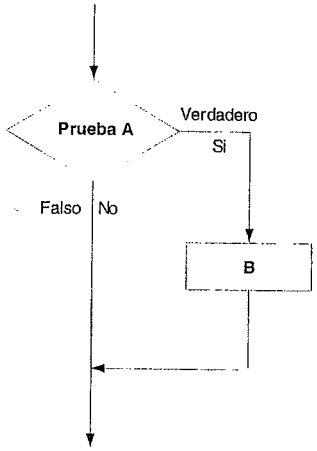
Mediante las estructuras de control se puede controlar el flujo de ejecución de un programa. Es casi imposible no encontrar una sentencia de control en cualquier programa que se realice.

Las estructuras de control se dividen en dos grupos:

- ***Estructuras de decisión.*** Son las encargadas de comprobar condiciones, y dependiendo de los resultados, realizar diferentes operaciones.
- ***Estructuras de bucle.*** También son llamadas repetitivas. Se utilizan para ejecutar una o más líneas de código un número determinado de veces.

Estructuras de decisión





Dentro de este apartado existen diferentes estructuras de decisión, estas son:

If ... Then

Esta estructura se utilizará siempre que se desee ejecutar una o más instrucciones dependiendo de que se cumpla o no una condición. Las sintaxis de esta estructura es la siguiente:

If *condición* **Then** *instrucción*

Veamos un ejemplo:

```
If A = 10 Then Y = -5     'Si A vale 10 entonces Y valdrá -5
```

En cambio si se desea que se ejecute más de una instrucción cuando se cumpla la condición, la sintaxis será la siguiente:

If *condición* **Then**
instrucciones
End If

Veamos un ejemplo:

```
If A = 10 Then          'Si A vale 10
    Y = -5              'Y toma el valor -5
    Z = Y * 2           'Z toma el valor de Y multiplicado por 2
End If
```


Tanto si se cumple o no la condición el programa continuará ejecutándose en la línea siguiente a **End If**.

Esta última sintaxis también es válida para un bloque de una sola instrucción como pasaba en el primer ejemplo.

If ... Then ... Else

Esta estructura se utiliza para definir varios bloques de instrucciones, uno de los cuales se ejecutará dependiendo de la condición que se cumpla. Su sintaxis es la siguiente:

If *condición 1* **Then**
instrucciones 1
ElseIf *condición 2* **Then**
instrucciones 2
.....
(Else
instrucciones n)
End If

El funcionamiento de esta estructura es el siguiente. Visual Basic evalúa la primera condición. Si no se cumple evalúa la siguiente condición y así sucesivamente hasta que encuentra una condición que se cumpla; entonces ejecutará el bloque de instrucciones correspondientes a la condición que se cumple. En caso de que no se cumpla alguna condición y si se ha incluido la cláusula opcional **Else**, el programa ejecutará el bloque de instrucciones que haya después de esta cláusula.

Veamos un ejemplo:

```
If Valor = 1 Then          'Si Valor vale 1
    X = 7                      'X tomará el valor 7
    Y = 6                      'Y tomará el valor 6
```

```
ElseIf Valor = 2 Then          'Si Valor vale 2
    X = 9                          'X tomará el valor 9
    Y = 8                          'Y tomará el valor 8
ElseIf Valor = 3 Then          'Si Valor vale 3
    X = 11                         'X tomará el valor11
    Y = 10                         'Y tomará el valor 10
Else                           'Si no se cumple ninguna de las condiciones anteriores
    Print "Error"              'Imprime el mensaje "Error"
End If
```

Select Case → Nació con la programación modular.

Esta estructura se utiliza como alternativa a la estructura **If ... Then ... Else** para ejecutar selectivamente un bloque de instrucciones entre varios bloques de instrucciones puesto que hace el código más legible cuando hay varias opciones. Su sintaxis es la siguiente:

```
Select Case expresión
    Case lista expresiones 1
        instrucciones 1
    Case lista expresiones 2
        instrucciones 2
    ...........
    (Case Else
        instrucciones n)
    End Select
```

Cada *lista de expresiones* es una lista de uno o más valores. Si hay más de un valor, éstos deben estar separados por comas (,). Cada bloque de *instrucciones* debe estar formado por ninguna o más instrucciones. Si ha incluido la cláusula opcional **Case Else** esta se ejecutará en caso de que ningún valor de la lista de expresiones coincida.

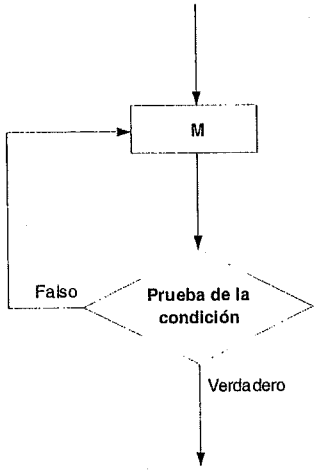
IMPORTANTE !!

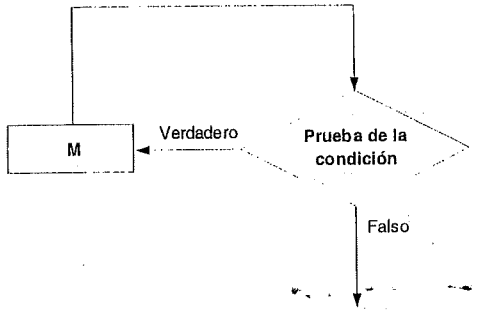
*En caso de que más de una sentencia **Case** coincida con una expresión sólo se ejecutará la primera de ellas.*

Veamos un ejemplo:

```
Select Case Mes                 'Comprueba el valor de Mes
    Case 4, 6, 9, 11                'Si Mes vale 4, 6, 9 u 11
        Dias = 30                   'Dias almacena el valor 30
    Case 1, 3, 5, 7, 8, 10, 12      'Si Mes vale 1, 3, 5, 7, 8, 10 o 12
        Dias = 31                   'Dias almacena el valor 31
    Case 2                          'Si Mes vale 2
        Dias = 28                   'Dias almacena el valor 28
    Case Else
        Print "Error mes incorrecto"  'Imprime el mensaje "Error mes incorrecto"
End Select
```

Estructuras de bucle





En este apartado tenemos las siguientes estructuras de bucle:

Do ... Loop

Esta estructura permite ejecutar un bucle de instrucciones un número indeterminado de veces. Su sintaxis es la siguiente:

```
Do While condición
    instrucciones
Loop
```

Este bucle primero evalúa **Do While** *condición*. Si no se cumple salta a la instrucción que sigue a **Loop**. Si se cumple *condición* se ejecutan *instrucciones* y se vuelve a evaluar **Do While** *condición* hasta que no se cumpla, es decir, mientras sea verdadera (**True**). Veamos un ejemplo:

```
A = 5                 'A se inicializa con el valor 5
Do While A > 1        'Comprueba que A sea mayor de 1
    Print A           'Imprime el valor de A
    A = A - 1         'Disminuye en 1 el valor de A
Loop                  'Vuelve a Do While
```

En cambio si desea que al menos se ejecute una vez el bucle, la sintaxis de la estructura es la siguiente:

```
Do
    instrucciones
Loop While condición
```

Veamos un ejemplo:

```
A = 5                 'A se inicializa con el valor 5
Do                    'Inicio del bucle
    Print A           'Imprime el valor de A
    A = A - 1         'Disminuye en 1 el valor de A
Loop While A > 1      'Comprueba que A sea mayor que 1
```

Existen dos variantes del bucle **Do ... Loop**, éstas son:

Do Until ... Loop

Se ejecuta cero o más veces. Es decir, si la condición no se cumple (**False**), la primera vez no se ejecuta. Su sintaxis es la siguiente:

```
Do Until condición
    instrucciones
Loop
```

Veamos un ejemplo:

```
A = 5                 'A se inicializa con el valor 5
Do Until A > 1        'Comprueba que A sea mayor de 1
    Print A           'Imprime el valor de A
    A = A - 1         'Disminuye en 1 el valor de A
```



```
Loop                  'Vuelve a Do While
```

Este bucle no se ejecutará ninguna vez puesto que la condición se cumple (A es mayor que 1) y por lo tanto es verdadera (**True**).

Do ... Loop Until

Se ejecuta una o más veces. Se diferencia de la anterior en que la condición se comprueba después de haberse ejecutado una vez *instrucciones*. Su sintaxis es la siguiente:

```
Do
    instrucciones
Loop Until condición
```

Veamos un ejemplo:

```
A = 5                 'A se inicializa con el valor 5
Do                    'Inicio del bucle
    Print A           'Imprime el valor de A
    A = A - 1         'Disminuye en 1 el valor de A
Loop Until A > 1      'Comprueba que A sea mayor que 1
```

Este bucle sólo se ejecutará una vez puesto que la condición se cumple (A es mayor que 1) y por lo tanto es verdadera (**True**).

For ... Next

Esta estructura se utilizará cuando sepamos el número de veces que se van a ejecutar las instrucciones. A diferencia del bucle **Do**, éste utiliza una variable llamada contador que incrementa o disminuye su valor en cada repetición. Su sintaxis es la siguiente:

```
For contador = inicio To fin (Step incremento)
    instrucciones
Next (contador)
```

Su funcionamiento es el siguiente:

1. Establece *contador* con un valor inicial (*inicio*).
2. Comprueba si *contador* es mayor o menor que *fin* (dependiendo de si incremento es positivo o negativo). Si lo es sale del bucle.
3. Ejecuta *instrucciones*.

4. Incrementa *contador* en 1 a no ser que se especificase otro valor en el parámetro opcional **Step**.

5. Repite los pasos 2 a 4.

Veamos un ejemplo:

```
For X = 1 To 20 Step 2      'X se inicializa con el valor 1 y se incrementa en 2 unidades
    Print X                 'Imprime el valor de X
Next                        'Vuelve al principio del bucle
```

For Each ... Next

Esta sentencia es similar a **For ... Next**, con la diferencia que esta sentencia repite un grupo de instrucciones por cada elemento de una colección de objetos o de una matriz. Su sintaxis es la siguiente:

For Each ***elemento*** **In** ***grupo***
    instrucciones
Next (*elemento*)

Este tipo de bucle es útil cuando no se sabe el número de elementos que hay en una colección.

Hay que tener en cuenta las siguientes restricciones cuando se trabaja con el bucle **For Each ... Next**.

El parámetro elemento *tiene que ser una variable de tipo* **Variant** *o una variable tipo* **Object**.

El parámetro elemento *sólo puede ser de tipo* **Variant** *cuando se trate de matrices.*

No se puede utilizar el bucle **For Each ... Next** *para una matriz de tipos definidos por el usuario, ya que el tipo* **Variant** *no puede contener un tipo definido por el usuario.*

Tampoco hay que olvidar uno de los puntos importantes cuando se trabaja con bucles, nos referimos a cómo provocar la salida de un bucle antes de que se cumpla la condición de finalización del mismo, las sentencias que nos permiten hacer esto son las que aparecen a continuación:

Bucle	Instrucción de Salida
For ... Next	Exit For
Do ... While ... Loop	Exit Do
Do ... Until ... Loop	Exit Do

Conversión de tipos de datos

Visual Basic proporciona una serie de funciones que se usan para convertir valores en tipos de datos específicos:

Función de conversión	Convierte la expresión a
CBool	Boolean
CByte	Byte
CCur	Currency
CDate	Date o Fecha.
CDbl	Double
CInt	Integer
CLng	Long
CSng	Single
CStr	String
CVar	Variant
CVErr	Error

Ha de tener en cuenta que los valores que se pasen a estas funciones han de ser válidos para el tipo de datos destino, de no ser así, se producirá un error.

Sistemas de numeración

Aunque siempre se utiliza la notación decimal (base 10) cuando se ha de utilizar un valor, Visual Basic también puede trabajar con número octales (base 8) y número hexadecimales (base 16).

*Para utilizar la notación octal es necesario colocar el prefijo **&O** delante del número. Su formato sería **&Oxxx***

*Para utilizar la notación hexadecimal es necesario colocar el prefijo **&H** delante del número. Su formato sería **&Hxxx***

En la siguiente tabla se muestran algunos ejemplos de números codificados en los distintos sistemas de numeración:

DECIMAL	OCTAL	HEXADECIMAL
9	&O11	&H9
15	&O17	&HF
16	&O20	&H10
20	&O24	&H14
255	&O377	&HFF

Hasta aquí se han visto las instrucciones y estructuras necesarias para poder programar en Visual Basic, no obstante a lo largo del libro irá aprendiendo más sobre la sintaxis del programa, como los objetos, a los cuales se dedicará un capítulo entero.

4. Desarrollar una aplicación con Visual Basic

Para crear una aplicación sencilla con Visual Basic sólo se necesitan unos pocos minutos. Se puede crear la interfaz de usuario *dibujando* controles (cuadros de texto, botones de comando, etc.) en un formulario. A continuación deberá establecer las propiedades del formulario y los controles (título, color, tamaño, etc.). Y por último deberá escribir el código que pónga en funcionamiento la aplicación.

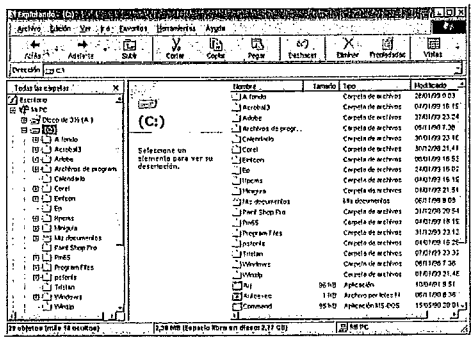
En este capítulo veremos una introducción al proceso de desarrollo de aplicaciones, la descripción de los términos y conocimientos necesarios para usar Visual Basic.

Conceptos de Visual Basic

Para entender el proceso de desarrollo de una aplicación, es útil conocer algunos conceptos alrededor de los cuales está construido Visual Basic.

Si no conoce todavía ningún lenguaje de programación que funcione bajo el entorno Windows como por ejemplo Visual Basic, será necesario que se familiarice con tres conceptos clave.

Ventanas



Una ventana es una región rectangular con sus propios límites. Pueden existir diferentes tipos de ventanas: ventana de documento, ventana del Explorador de Windows, un cuadro de diálogo, un botón de comando, un icono, un cuadro de texto, un botón de opción, etc.

Botón Comando

Icono

Botón Opción

El sistema Windows administra estas ventanas asignándole a cada una de ellas un número identificador, siendo el sistema quien controla cada una de estas ventanas para ver si existen signos de actividad en cada una de ellas (pulsar un botón, cerrar una ventana, etc.).

Eventos

Un evento se puede producir de tres formas distintas:

- El usuario realiza una acción en el programa como por ejemplo efectuar un *clic* con el ratón.
- Mediante programación. La aplicación puede disponer de una parte de código que obligue al programa a realizar una copia de seguridad en una fecha determinada.
- Como resultado de acciones producidas en otras ventanas.

Cada vez que se produce un evento se envía un mensaje al sistema operativo. El sistema procesa el mensaje y lo transmite al resto de las ventanas del programa y éstas realizan la acción apropiada basándose en las instrucciones que tienen cada una de ellas para tratar ese mensaje en particular.

IMPORTANTE !!

Un evento es todo acontecimiento o suceso que estimula al sistema y a los que éste debe responder.

Mensajes

El mensaje es la respuesta generada por un evento. Debido a que el número de mensajes producidos puede ser muy elevado, Visual Basic evita tener que tratar todos los controladores de mensaje de bajo nivel ya que los controla automáticamente.

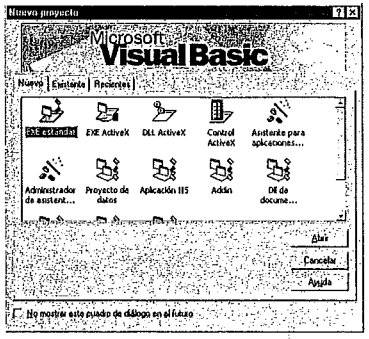
El entorno de desarrollo (IDE)

El entorno de trabajo de Visual Basic se llama entorno integrado de desarrollo o IDE ya que integra muchas funciones diferentes como el diseño, modificación, compilación y depuración en un entorno común.

A continuación pasaremos a describir los pasos necesarios para ejecutar Visual Basic 6.

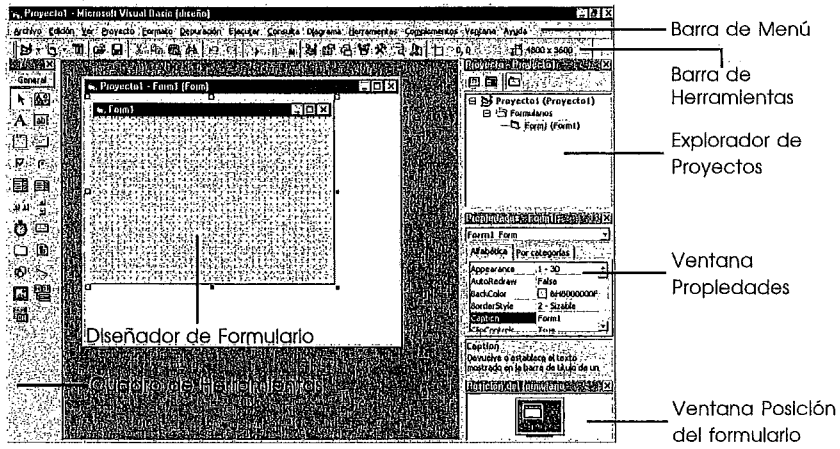
1. *Haga clic sobre el botón* Inicio *de la barra de tareas.*
2. *Seleccione* ***Programas*** *y a continuación* ***Microsoft Visual Basic 6.0.***
3. *Por último seleccione la opción* ***Visual Basic 6.0.***

Cuando se inicie Visual Basic aparecerá la ventana **Nuevo proyecto**, donde se muestran todos los tipos de proyecto que se pueden crear con Visual Basic.



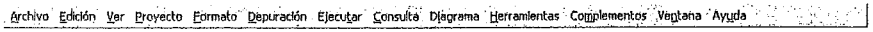
1. *Seleccione el tipo de proyecto* ***EXE estándar.***

Si es la primera vez que inicia Visual Basic, el entorno de desarrollo que aparece es como el de la figura siguiente:



El entorno de desarrollo de Visual Basic consta de los elementos enumerados en la figura anterior y que a continuación pasamos a describir.

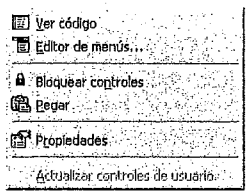
Barra de menús



Esta barra presenta los comandos que se usan para trabajar con Visual Basic, también incorpora los típicos menús de **Archivo, Edición, Ver, Ventana** y **Ayuda**, además de las otras opciones para trabajar con funciones de programación como **Proyecto, Formato** o **Depuración.**

Menús contextuales

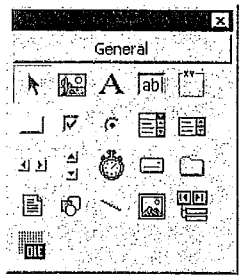
Contienen accesos directos a acciones que se realizan con frecuencia. Para abrir un menú contextual haga clic en el botón secundario del ratón en el objeto que se está usando. La lista específica de opciones depende de la parte del entorno en la que se hace clic con el botón secundario del ratón.



Barras de herramientas

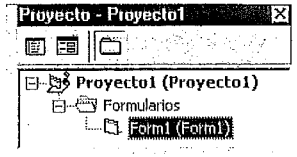
Esta barra de herramientas contiene un acceso rápido a las opciones más comunes en el entorno de programación. Ésta es la barra que presenta Visual Basic por defecto, sin embargo dispone de otras barras de herramientas que puede activar o desactivar con la opción **Barras de herramientas** que se encuentra en el interior del menú **Ver.**

Cuadro de herramientas



Este cuadro proporciona un conjunto de herramientas que puede usar en tiempo de diseño del formulario. Además del diseño del cuadro de herramientas predeterminado, puede crear su propio diseño personalizado si selecciona **Agregar ficha** en el menú contextual y agrega controles a la ficha resultante.

Ventana Explorador de proyectos

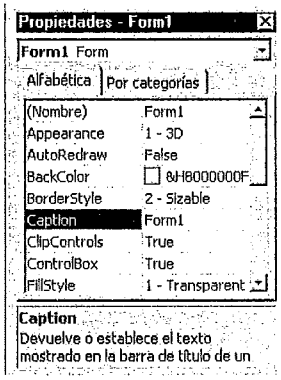


Esta ventana informa de los formularios que componen el proyecto que se está utilizando.

IMPORTANTE !!

Un proyecto es un conjunto de archivos que se usa para generar un programa o aplicación. También es posible generar un grupo de proyectos.

Ventana Propiedades



Esta ventana muestra las propiedades del control o formulario que esté seleccionado en ese momento.

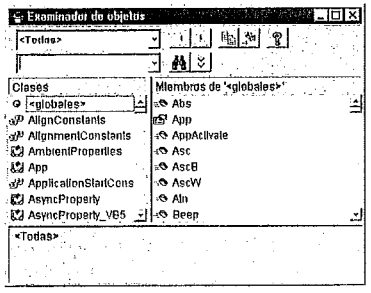
IMPORTANTE !!

Una propiedad es una característica de un objeto, como su color, tamaño, fuente de letra, etc.

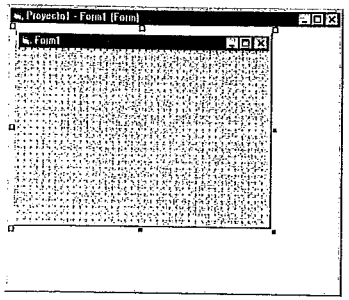
Examinador de objetos

Enumera los objetos disponibles que puede usar en su proyecto proporcionando una manera rápida de desplazarse a través del código.

Se puede usar el Examinador de objetos para explorar objetos de Visual Basic y otras aplicaciones, ver qué métodos y propiedades están disponibles para esos objetos, y pegar código de procedimientos en su aplicación.

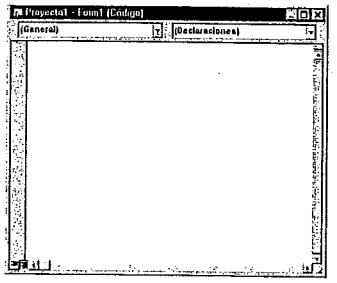


Diseñador de formularios



Este diseñador funciona como una ventana en la que se personaliza el diseño de la interfaz de su aplicación pudiendo añadir controles, gráficos e imágenes para darle la apariencia que desee. Cada formulario de la aplicación tiene su propia ventana de diseño.

Ventana Editor de código



Esta ventana funciona igual que un procesador de textos solo que es un editor muy especializado con cierto número de características que facilita en gran medida la escritura de código de Visual Basic.

Se utiliza para teclear el código de cada uno de los controles que componen nuestra aplicación. Se crea una ventana Editor de código diferente para cada formulario o módulo de la aplicación.

EL código de cada módulo está subdividido en distintas secciones para cada objeto del módulo. Para pasar de un cuadro a otro se utiliza el cuadro de lista **Objeto**.

Cada sección de código puede contener varios procedimientos distintos, a los que se tiene acceso mediante el cuadro de lista **Procedimiento**. Puede ver el código de un solo procedimiento o verlos todos.

Visual Basic hace que la escritura de código sea mucho más sencilla mediante características que llenan automáticamente instrucciones, propiedades y argumentos.

A medida que va introduciendo código, el editor muestra lista de elecciones adecuadas, prototipos de instrucciones o funciones, o valores. Estas opciones se pueden activar o desactivar por medio del menú **Opciones** eligiendo la opción **Herramientas** y seleccionando la ficha **Editor**.

Ventana Posición del formulario

Permite colocar los formularios de su aplicación utilizando una pequeña representación gráfica de la pantalla.

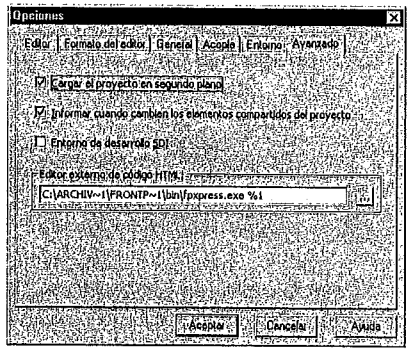
Ventanas Inmediato, Locales e Inspección

Estas ventanas adicionales se proporcionan para la depuración de la aplicación. Sólo están disponibles cuando ejecuta la aplicación dentro del IDE.

Tenga presente que usted puede mover libremente las diferentes ventanas que componen el IDE, para ello use el mismo método que utilizaba en Windows.

Opciones de entorno

Visual Basic le permite configurar el entorno de trabajo de forma que se adapte a su forma de trabajar. Podrá elegir entre una interfaz de documentos simple (SDI) o una interfaz de documentos múltiples (MDI). Un vez haya configurado su entorno de trabajo, éste se mantendrá en futuras sesiones de Visual Basic a no ser que usted lo vuelva a cambiar.



Para alternar entre un entorno SDI o MDI, ha de efectuar los siguientes pasos:

1. *Abra el menú **Herramientas** y seleccione la opción **Opciones**.*
2. *Selecciona la ficha **Avanzado**.*
3. *Active o desactive la casilla **Entorno de desarrollo SDI**.*

4. *Pulse el botón* [Aceptar].

*La diferencia entre un entorno **SDI** y un entorno **MDI** es que en el primero (SDI) todas las ventanas IDE se pueden mover libremente por cualquier lugar de la pantalla siempre y cuando la aplicación de Visual Basic sea la activa; mientras que en el segundo (MDI) todas las ventanas del IDE están contenidas en una única ventana primaria de tamaño ajustable.*

Creación de una aplicación

Para crear una aplicación con Visual Basic se han de seguir los pasos que a continuación se detallan:

- Crear la interfaz, es decir, la parte visual de la aplicación con la que el usuario ha de interactuar.
- Establecer las propiedades de los controles y formularios.
- Teclear el código necesario para cada uno de los controles y formularios.

Los formularios y controles son los elementos de desarrollo básicos que se usan cuando se crea la interfaz.

Los formularios son objetos que exponen las propiedades que determinan su apariencia, los métodos que definen su comportamiento y los eventos que establecen la forma en que interactúan con el usuario.

Al establecer las propiedades del formulario y al escribir el código del programa para responder a los eventos del mismo, se personaliza el objeto para cubrir las necesidades de la aplicación.

Los controles son objetos que están contenidos en los objetos de formularios. Cada tipo de control tiene sus propias propiedades, métodos y eventos dependiendo de la finalidad para la que sean utilizados.

Descripción de las propiedades, métodos y eventos

Los formularios y controles de Visual Basic son objetos que exponen sus propios métodos, propiedades y eventos. Las propiedades se pueden considerar como atributos de un objeto, los métodos como sus acciones y los eventos como sus respuestas.

Sin duda, lo explicado en el párrafo anterior le parecerá un poco confuso, pero no se preocupe a continuación lo veremos con un ejemplo del mundo real, y para no alejarnos del tema que nos ocupa supondremos que el objeto en cuestión es un ordenador.

Las propiedades del ordenador podrían ser:

- Modelo.
- Velocidad.
- Memoria RAM.
- Memoria Caché.

Si trasladáramos estas propiedades al lenguaje de programación Visual Basic el código podría ser el siguiente:

```
Ordenador.Modelo = Pentium II
Ordenador.Velocidad = 300 MHz
Ordenador.MemoriaRAM = 64 Gb
Ordenador.MemoriaCache = 512 Kb
```

Un ordenador tiene también métodos o acciones que puede realizar como por ejemplo imprimir o listar. En código de Visual Basic sería:

```
Ordenador.Imprimir
Ordenador.Listar
```

Para describir un evento, es decir, una respuesta el código describirá el comportamiento del ordenador cuando se produzca un evento invocando a uno o más métodos. El ordenador podría responder a un evento como se muestra a continuación:

```
Sub Ordenador_Imprimir ()
    Ordenador.ComprobarImpresora
    Ordenador.Listar
End Sub
```

En el siguiente capítulo empezaremos a construir nuestra primera aplicación con Visual Basic.

5. La primera aplicación

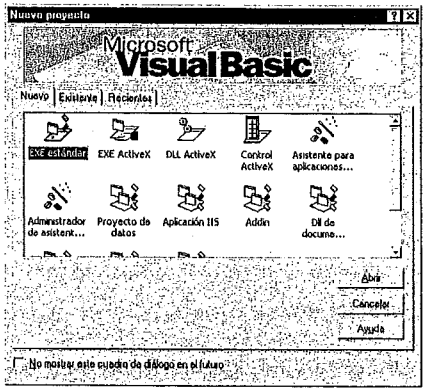
Si usted se ha leído los dos capítulos anteriores, ya está en disposición de realizar su primera aplicación. Para ello escribiremos un programa sencillo que le sirva de toma de contacto.

Le recomendamos que aprenda bien los pasos que se van a describir para insertar controles en un formulario, ya que en los próximos capítulos no haremos mención a ello, simplemente le diremos que inserte un determinado control en un formulario.

Creación de un nuevo proyecto

Lo primero que deberá hacer para crear una aplicación será entrar dentro del entorno de Visual Basic. Para ello:

1. *Haga clic sobre el botón* Inicio *de la barra de tareas.*
2. *Seleccione* ***Programas*** *y a continuación* ***Microsoft Visual Basic 6.0.***
3. *Por último seleccione la opción* ***Visual Basic 6.0.***

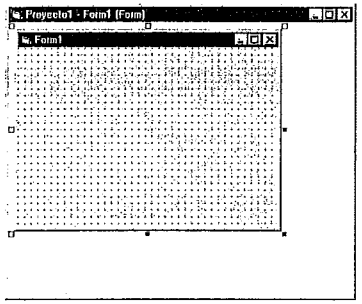


Si tiene creado un icono de acceso directo en el escritorio bastará con hacer doble clic sobre él.

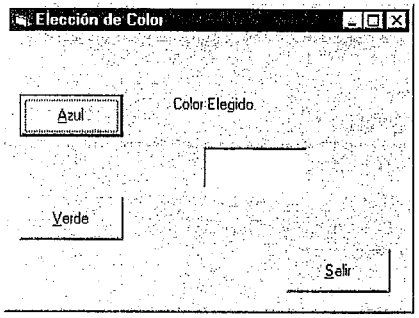
4. *Seleccione de la ventana* ***Nuevo proyecto*** *el tipo de proyecto* ***EXE estándar.***

5. *Haga clic sobre el botón* Abrir.

En estos momentos el formulario que hay en su pantalla debe presentar una apariencia igual a la figura de la derecha.



Creación de la aplicación



Para nuestra primera aplicación seguiremos los pasos especificados en el capítulo 4 cuyo apartado se titula **"Creación de una aplicación"**.

La aplicación que vamos a crear consiste en un formulario con tres botones de comando, dos de ellos con un nombre de color determinado y el tercero con la opción de salir; una ventana de texto en la cual se escribirá el nombre del color elegido y una etiqueta con un título.

Diseñar el Formulario

Puesto que los objetos de un formulario son los elementos de desarrollo fundamentales de una aplicación en Visual Basic, comenzaremos por dibujar los objetos que contendrá el formulario. La aplicación contendrá los siguientes controles:

Botón de comando.

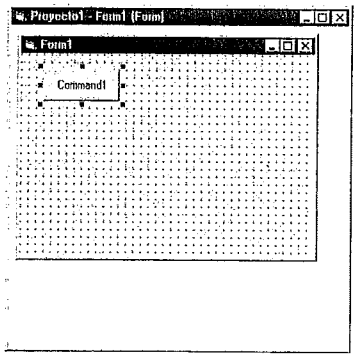
Cuadro de texto.

Etiqueta.

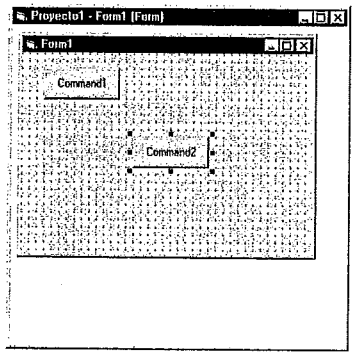
Cuando quiera insertar un control en un formulario, lo puede realizar de forma manual o de forma automática.

❍ Forma manual:

1. *Haga clic sobre el control* .
2. *Desplace el puntero del ratón dentro del formulario y observe como éste se convierte en una cruz* +.
3. *Sitúe la cruz donde desee que aparezca el control, haga clic y sin soltar el botón arrastre el ratón hasta que el control tenga el tamaño deseado.*
4. *Suelte el botón del ratón.*



❍ Forma automática:



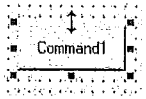
1. *Realice un doble clic sobre el control* .

Observe como Visual Basic inserta el control en el centro del formulario.

Si quiere desplazar el control a otro lugar del formulario basta con efectuar un clic sobre él y sin soltar desplazarlo hasta el lugar deseado.

En cambio si desea modificar su tamaño realice los siguientes pasos:

1. *Efectúe un clic sobre el control que desea modificar para seleccionarlo.*
2. *Sitúe el puntero del ratón sobre uno de los puntos que enmarcan el control, cuando el puntero se transforme en una doble flecha* ↕ *, efectúe un clic y sin soltar arrastre hasta que el control tenga el tamaño que desee.*



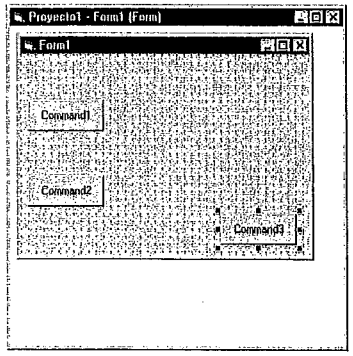
3. *Suelte el botón del ratón.*

Ahora que ya sabe cómo insertar un control, le enseñaremos la forma de eliminarlo. Para ello:

1. *Efectúe un clic sobre el control que desea eliminar para seleccionarlo.*
2. *Pulse la tecla* ***Supr****.*

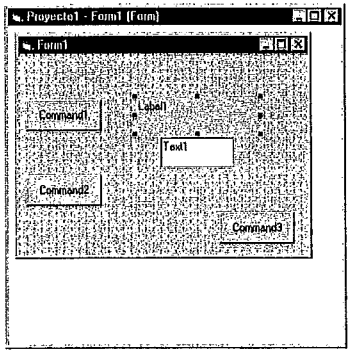
Antes de comenzar a insertar los controles de la aplicación borre, los que haya en el formulario hasta que quede vacío.

A partir de este momento consideraremos que usted ya sabe cómo se inserta, mueve, modifica o borra un control, por lo tanto no se explicarán todos los pasos necesarios para realizarlo. Si tiene alguna duda vuelva a consultar este apartado.



Ahora comenzaremos a diseñar el formulario. Para ello:

1. *Sitúe los tres botones de comando (****CommandButton****) tal como aparecen en la figura de la izquierda.*



2. *Inserte el cuadro de texto (****TextBox****) en el mismo lugar en que aparece en la figura de la derecha.*
3. *Haga lo mismo con la etiqueta (****Label****).*

Una vez diseñado el entorno gráfico pasaremos a establecer las propiedades del formulario y de los controles.

Establecer las propiedades del Formulario y los Controles

Las propiedades de un formulario se pueden establecer en *tiempo de diseño* o en *tiempo de ejecución*.

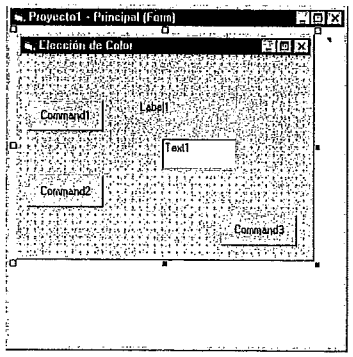
Tiempo de diseño: *Cualquier momento mientras se está desarrollando una aplicación en el entorno de Visual Basic trabajando con formularios y controles, se establecen propiedades y se escribe código para los eventos.*

Tiempo de ejecución: *Cualquier momento mientras se ejecuta la aplicación y se interactúa con ella como lo haría un usuario.*

La tarea de establecer las propiedades a los controles que ha insertado es sencilla, ya que dispone de la ventana Propiedades. Las propiedades que estableceremos a continuación son en *tiempo de diseño*. En el siguiente apartado veremos cómo se hace en *tiempo de ejecución*.

a) Establecer las propiedades del formulario (**Form**).

1. *Efectúe un clic sobre una zona vacía del formulario.*
2. *Active la ventana Propiedades y haga doble clic sobre la propiedad* ***Caption****.*
3. *Teclee:* **Elección de Color**.
4. *Haga doble clic sobre la propiedad* ***(Nombre)*** *y teclee:* **Principal**.



La propiedad ***Caption*** *hace referencia al texto que aparece en la barra de título del formulario.*

La propiedad ***(Nombre)*** *hace referencia al nombre que se le da al formulario.*

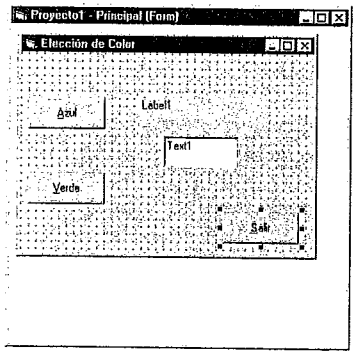
Si desea saber más cosas sobre cada una de las propiedades de un control determinado, efectúe un clic con el ratón sobre ella y pulse la tecla ***F1****.*

b) Establecer las propiedades de los botones (**CommandButton**).

Ahora que ya sabe como se establecen las propiedades de un control pasaremos a indicarle las propiedades que debe establecer en cada uno de los botones de una forma más simplificada:

Objeto	Propiedad	Valor
Command1	(Nombre)	Azul

Objeto	Propiedad	Valor
	Caption	&Azul
Command2	(Nombre)	Verde
	Caption	&Verde
Command3	(Nombre)	Salir
	Caption	&Salir



*Observe como en el valor de la propiedad **Caption** hemos usado el carácter **&**. Este carácter recibe el nombre de acelerador y su función es permitir acceder a este control por medio del teclado.*

*Para ello se ha de utilizar conjuntamente la tecla **Alt** junto con la letra que aparece subrayada. Por ejemplo para acceder a la tecla con el rótulo **Verde** deberá pulsar la tecla **Alt** y sin soltarla la tecla **V**.*

*El carácter **&** debe colocarse antes de la letra que desee que aparezca subrayada. Por ejemplo si en la palabra Salir desea que salga subrayada la letra **S** deberá escribir en la propiedad **Caption** el texto **&Salir**; en cambio si quiere que salga subrayada la letra **a** deberá escribir en la propiedad **Caption** el texto **S&alir**.*

Nunca deberá tener más de un control al que se pueda acceder por la misma letra, ya que entonces Visual Basic no sabrá a cuál de ellos quiere acceder y por lo tanto no hará nada.

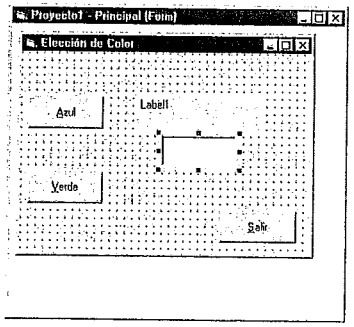
Para resolver este problema evite que se pueda acceder a más de un control con la misma letra de la siguiente manera:

*Supongamos hay dos controles que comienzan por la misma letra **Amarillo** y **Azul**. Una forma de resolverlo sería escribir en la propiedad **Caption** del botón amarillo el texto **&Amarillo** y en la propiedad **Caption** del botón azul el texto **A&zul**. De este modo el texto que aparecerá en cada uno de los controles será **Amarillo** y **Azul**.*

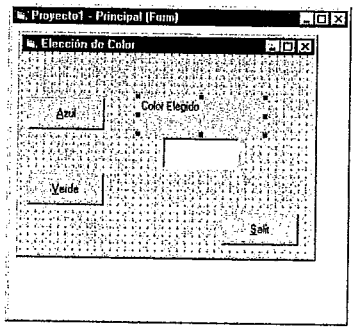
c) Establecer las propiedades de la caja de texto (**TextBox**).

Objeto	Propiedad	Valor
Text1	(Nombre)	Color
	Text	(Vacío)

El aspecto de su formulario debe ser como el de la figura de la derecha.



d) Establecer las propiedades de la etiqueta (**Label**).

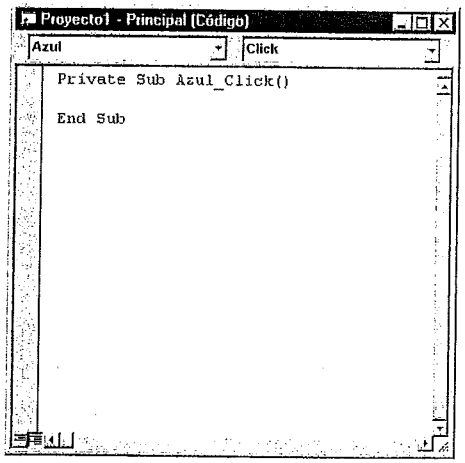


Objeto	Propiedad	Valor
Label1	(Nombre)	Titulo
	Caption	Color Elegido

Una vez finalizado el diseño del formulario, el aspecto que debe presentar tiene que ser como el de la figura de la izquierda.

Escritura del código

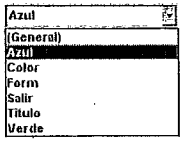
Por último tendrá que teclear el código necesario para que la aplicación funcione. El funcionamiento de la aplicación es sencillo, se trata de que cuando usted pulse un botón de comando (*CommandButton*), aparezca en el cuadro de texto (*TextBox*) el nombre del color que ha seleccionado.



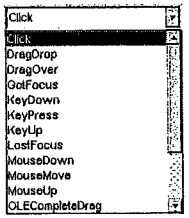
Efectúe los pasos que a continuación se detallan para teclear el código.

1. *Haga doble clic sobre el botón etiquetado **Azul**.*

A continuación aparecerá la ventana de Código de la derecha.



En la parte superior de esta ventana encontramos dos listas desplegables, la de la izquierda (figura de la izquierda) muestra todos los controles disponibles en el formulario actual ordenados alfabéticamente.

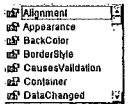


La lista de la derecha (figura de la derecha) muestra los eventos ordenados alfabéticamente referentes al control seleccionado.

*Visual Basic siempre mostrará un evento por defecto, dependiendo del control sobre el que se haya hecho doble clic. Normalmente será el evento **Click**, que es el que se produce cuando efectuamos un clic sobre el control.*

2. *Teclee el siguiente código para el evento **Click** del botón **Azul**.*

```
Private Sub Azul_Click()
        Color.Text = "AZUL"          'Asigna a la propiedad Text el valor AZUL
End Sub
```



Observe como después de teclear el punto (.) aparece una lista desplegable con todos las propiedades de ese control ordenadas alfabéticamente. De este modo si no quiere perder tiempo tecleándola, bastará con que haga doble clic sobre ella.

3. *Seleccione de la lista desplegable superior izquierda de la ventana Código el control **Verde**.*

4. *Teclee el siguiente código para el evento **Click** del botón **Verde**.*

```
Private Sub Verdel_Click()
        Color.Text = "VERDE"         'Asigna a la propiedad Text el valor VERDE
End Sub
```

5. *Seleccione de la lista desplegable superior izquierda de la ventana Código el control **Salir**.*

6. *Teclee el siguiente código para el evento **Click** del botón **Salir**.*


```
Private Sub Salir_Click()
        End                          'Finaliza el programa
End Sub
```

*A partir de ahora y en adelante, recuerde que siempre sólo deberá copiar las líneas de código que aparezcan en **negrita**.*

EDUCACIÓN SECUNDARIA INSTITUTO ALVARO ALEDO Badajoz

7. *Cierre la ventana de Código.*

Guardar un proyecto

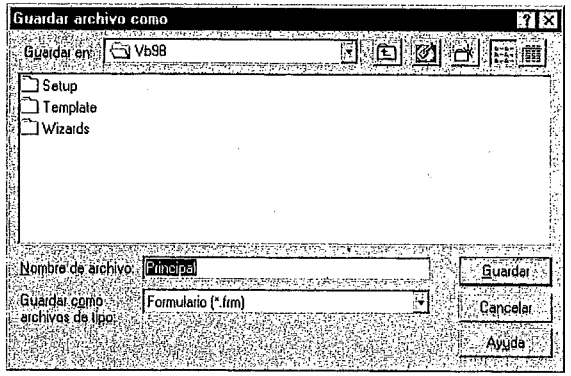
Como es habitual en cualquier aplicación de Windows, disponemos de dos maneras de almacenar la aplicación, por medio de la barra de menús o por medio de los botones de las barra de herramientas.

1. *Abra el menú **Archivo** y elija la opción **Guardar Proyecto**.*

O bien:

1. *Pulse el botón de la barra de herramientas.*

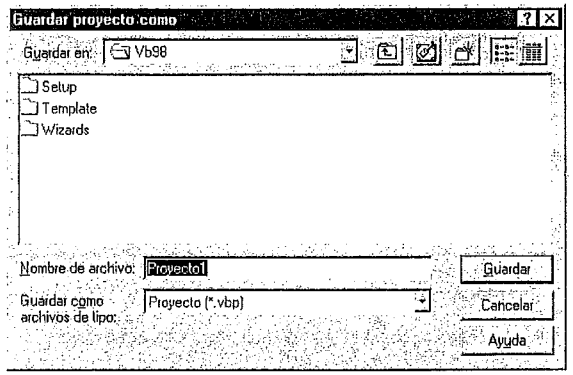
A continuación aparecerá el cuadro de diálogo que se muestra a continuación:



Observe como Visual Basic asigna el nombre **Principal** al archivo, es decir, el mismo nombre que le dimos a la propiedad **(Nombre)** del formulario.

Siempre que archive un proyecto por primera vez, Visual Basic solicitará un nombre de archivo por formulario y otro nombre para el proyecto que no tiene por qué ser el mismo.

2. *Pulse sobre el botón* [Guardar], *sin modificar el nombre que nos ofrece Visual Basic ya que lo consideramos adecuado.*



A continuación aparecerá la misma pantalla, pero solicitando el nombre del proyecto, fíjese que el nombre que sugiere es **Proyecto1.vbp**.

3. *Borre el nombre sugerido y teclee* ***Colores***.

4. *Haga clic en el botón* [Guardar].

Cuando se desarrolla una aplicación, y se almacena en disco, se crean diferentes tipos de archivos dependiendo del tipo de ficheros por los que esté formada dicha aplicación. Estos archivos son:

.vbp: *Archivo de proyecto.*

.frm: *Archivo de formulario.*

.frx: *Archivo de datos binarios.*

.cls: *Archivo de módulo de clase.*

.bas: *Archivo de módulo estándar.*

.ocx: *Archivo con controles ActiveX.*

.res: *Archivo de recursos.*

No olvide guardar cada cierto tiempo su aplicación mientras la está desarrollando. Aunque parezca una tontería, no lo es, ya que de producirse algún problema (corte de electricidad, bloqueo del ordenador), si usted almacenó con anterioridad su aplicación, evitará tener que volver a comenzar de nuevo desde el principio.

Ejecución de la aplicación

Por fin ha llegado el momento que esperaba con ansiedad, ejecutar la aplicación y ver el resultado de la misma. Esta tarea la puede realizar de tres formas distintas:

a) Por medio del menú.

1. *Abra el menú* ***Ejecutar*** *y elija la opción* ***Iniciar***.

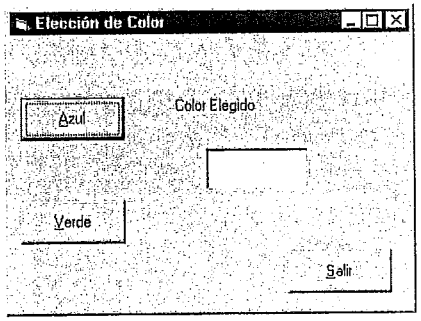
b) Por medio de la barra de herramientas.

1. *Haga clic sobre el botón* [▶].

c) Por medio de una tecla de función.

1. *Pulse la tecla de función* ***F5.***

La pantalla que le aparecerá será como la de la figura de la derecha.



2. *Efectúe un clic sobre el botón* [Azul].

Observe como en la caja de texto aparece el rótulo **AZUL**.

3. *Efectúe un clic sobre el botón* [Verde].

Observe como en la caja de texto se muestra el rótulo **VERDE**. Practique un poco para ver como cambia el rótulo que aparece en la caja de texto.

En estos momentos la ventana de la aplicación funciona de la misma manera que una ventana de Windows, es decir, puede moverla, cambiarle el tamaño, maximizarla, minimizarla y cerrarla.

4. *Pulse sobre el botón* [Salir] *para finalizar la aplicación.*

Descargar un proyecto de la memoria

Si desea eliminar un proyecto de la memoria RAM de su equipo para, por ejemplo, iniciar otro nuevo proyecto, deberá:

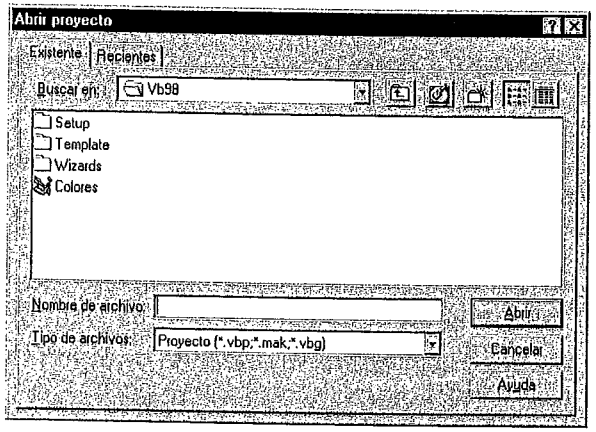
1. *Abrir el menú **Archivo** y elegir la opción **Quitar proyecto**.*

Si hay cambios pendientes de efectuar en el proyecto, se le pedirá que los guarde y después se cerrará el proyecto y se quitará del grupo de proyectos.

Recuperar un proyecto

La manera de recuperar un proyecto es similar a la de guardarlo. Siempre que desee recuperar un proyecto, deberá seguir estos pasos:

1. *Abra el menú **Archivo** y elija la opción **Abrir proyecto**.*



O bien:

1. *Efectúe un clic sobre el botón* .

A continuación aparecerá una pantalla como la figura de la izquierda.

Observe como esta ventana es similar a la que aparecía cuando guardaba un proyecto. Esta ventana cuenta con dos pestañas, ***Existente*** donde aparecen todos los proyectos que hay grabados en la carpeta **Vb98** y ***Recientes*** donde se muestran los proyectos que se han abierto últimamente.

2. *Seleccione por medio de un clic el proyecto que desea recuperar y pulse el botón* Abrir.

O bien:

2. *Efectúe un doble clic sobre el proyecto que desea recuperar.*

En unos momentos el proyecto estará cargado en la memoria de su equipo. Sin embargo no verá ni el formulario del proyecto ni el código, esto no tiene importancia si sólo desea ejecutar la aplicación, ya que para ello:

1. *Realice un clic sobre el botón **Iniciar*** .

Si quiere ver el formulario:

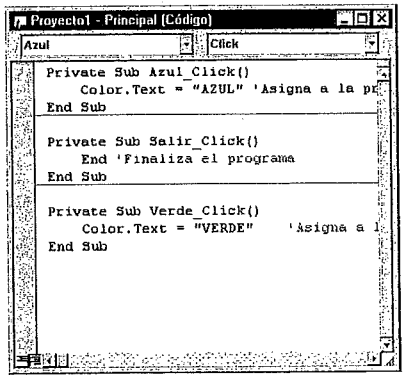
1. *Efectúe un doble clic sobre la carpeta* Formularios *que se encuentra en la ventana **Proyecto**.*
2. *Seleccione el formulario **Principal** y pulse sobre el botón* *para visualizarlo.*

Mejorando la aplicación

Si bien la aplicación que acabamos de crear funciona correctamente, vamos a realizar algunas mejoras. Haremos que el texto que vemos en el cuadro de texto con el nombre del color aparezca escrito con ese color. Para ello:

1. *Efectúe un clic en el botón **Ver código*** .

El aspecto de la pantalla es el que se muestra en la siguiente figura donde aparece el código perteneciente a cada uno de los botones de comando.



Añada las siguientes líneas de código a cada uno de los eventos:

2. *Teclee el siguiente código para el evento **Click** del botón **Azul**.*

```
Private Sub Azul_Click()
    Color.ForeColor = RGB(0, 0, 255)  'Color Azul
```

```
        Color.Text = "AZUL"          'Asigna a la propiedad Text el
                                     'valor AZUL
    End Sub
```

3. *Teclee el siguiente código para el evento **Click** del botón **Verde**.*

```
    Private Sub Verde_Click()
        Color.ForeColor = RGB(0, 255, 0)  'Color Verde
        Color.Text = "VERDE"         'Asigna a la propiedad Text el
                                     'valor VERDE
    End Sub
```

*Un valor de color RGB especifica la intensidad relativa de rojo, verde y azul que hacen que se muestre un color determinado. Su sintaxis es: **RGB(rojo, verde, azul)**. Cuanto mayor sea el valor de cada uno de los argumentos más claro será el color.*

El valor de cualquier argumento para la función RGB oscila entre 0-255. Cualquier valor que sea superior a 255 se considerará como 255.

*Si se quieren obtener otros colores hay que realizar una combinación de los tres. Por ejemplo para obtener el color amarillo se deben mezclar el rojo y el verde, por lo tanto la combinación será **RGB(255, 255, 0)**.*

4. *Archive de nuevo el proyecto haciendo clic en el botón* .

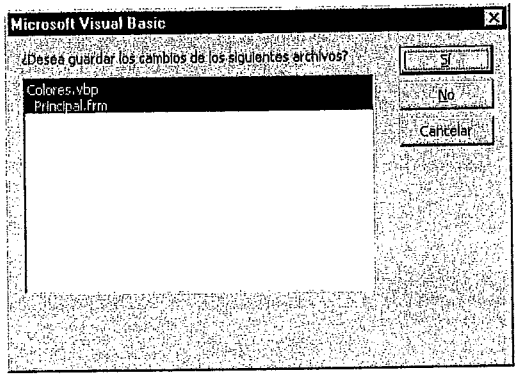
Si ejecuta de nuevo la aplicación observará que cuando efectúe un clic sobre cada uno de los botones correspondientes, el texto aparecerá escrito con su respectivo color.

Finalizar la sesión con Visual Basic

En este apartado le enseñaremos a finalizar una sesión de Visual Basic. Para ello deberá:

1. *Abrir el menú **Archivo** y seleccionar la opción **Salir**.*

Una vez haya seleccionado esta opción pueden suceder dos cosas:



a) Que se cierre Visual Basic.

b) Si alguno de los archivos que componen la aplicación ha sido modificado desde la última vez que se archivó el proyecto, aparecerá la siguiente pantalla:

En esta ventana Visual Basic preguntará si desea guardar los cambios realizados desde la última grabación.

2. *Pulse el botón* Sí *o* No *según crea conveniente.*

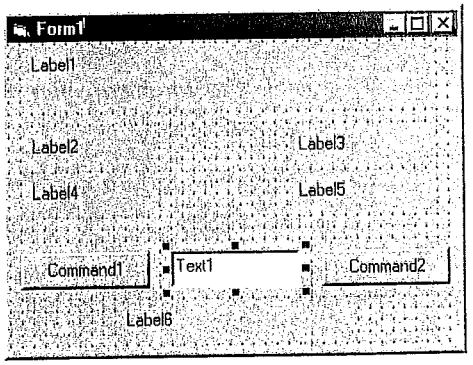
En caso de que hubiera varios archivos, Visual Basic permite desactivar alguno de ellos realizando un clic sobre dicho archivo.

La aplicación de la Euro-Calculadora

A continuación y aprovechando que en el año 2000 entrará en circulación el famoso *Euro* realizaremos una aplicación que consistirá en una Euro-Calculadora que le permitirá realizar la conversión de monedas entre pesetas y euros o entre euros y pesetas.

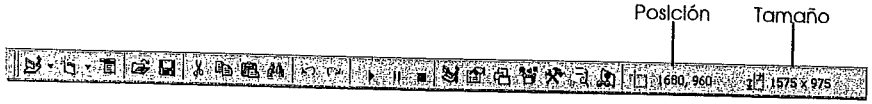
1. *Ejecute la aplicación **Visual Basic 6.0** si no la tiene en funcionamiento.*

2. *Seleccione de la ventana **Nuevo proyecto** el tipo de proyecto **EXE estándar**.*



Para ver la posición del control utilice el indicador que aparece en la barra de herramientas 2640, 960, el segundo de la derecha.

Para ver el tamaño del control utilice el indicador que aparece también en la barra de herramientas 1215 x 975, el primero de la derecha.



Seguidamente diseñe un formulario como el de la página anterior. Asígnele a los controles la posición y el tamaño especificado en la siguiente tabla:

Objeto	Posición	Tamaño
Label1	240, 120	4095 x 255
Label2	240, 960	1215 x 255
Label3	3000, 960	1335 x 255
Label4	240, 1440	1215 x 255
Label5	3000, 1440	1335 x 255
Label6	1200, 2760	2295 x 255
Command1	120, 2160	1335 x 375
Command2	3240, 2160	1335 x 375
Text1	1680, 2160	1335 x 375

1. *Pulse el botón de la barra de herramientas.*
2. *Teclee **Calculadora** como nombre del formulario y pulse el botón Guardar.*
3. *Teclee **Calculadora** como nombre del proyecto y pulse el botón Guardar.*

Una vez ha construido el diseño del formulario, pasaremos a establecer cada una de las propiedades del formulario y los controles:

a) Propiedades del formulario (**Form**).

Objeto	Propiedad	Valor
Form1	(Nombre)	Calculadora
	Caption	EURO CALCULADORA

b) Establecer las propiedades de los botones (**CommandButton**).

Objeto	Propiedad	Valor
Command1	(Nombre)	Pesetas
	Caption	&Pesetas a Euros
Command2	(Nombre)	Euros
	Caption	&Euros a Pesetas

c) Establecer las propiedades de las etiquetas (**Label**).

Objeto	Propiedad	Valor
Label1	(Nombre)	Titulo
	Alignment	2 - Center
	Caption	EURO CALCULADORA
	Font	Arial Black
	Tamaño Font	16
Label2	(Nombre)	Pesetas
	Alignment	2 - Center
	BorderStyle	1 - Fixed Single
	Caption	PESETAS
Label3	(Nombre)	Euros
	Alignment	2 - Center
	BorderStyle	1 - Fixed Single
	Caption	EUROS
Label4	(Nombre)	ValorPesetas

Objeto	Propiedad	Valor
	Caption	(Vacío)
Label5	(Nombre)	ValorEuros
	Caption	(Vacío)
Label6	(Nombre)	Enunciado
	Alignment	2 - Center
	BorderStyle	1 - Fixed Single
	Caption	INTRODUZCA CANTIDAD

*La propiedad **Alignment** devuelve o establece un valor que determina la alineación de un control. Los valores que puede tomar esta propiedad son:*

0 - El texto está alineado a la izquierda.

1 - El texto está alineado a la derecha.

2 - El texto está centrado.

*La propiedad **BorderStyle** devuelve o establece el estilo del borde de un objeto. Los valores que puede tomar para este tipo de control son:*

0 - Ninguno.

1 - Simple fijo.

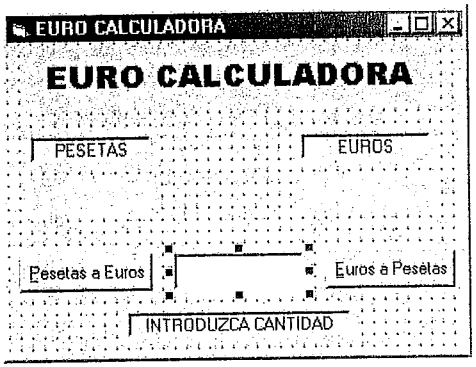
c) Establecer las propiedades del cuadro de texto (**TextBox**).

Objeto	Propiedad	Valor
Text1	(Nombre)	Valor
	TabIndex	1
	Text	(Vacío)

*La propiedad **TabIndex** de un control determina el lugar que ocupa en el orden de tabulación.*

Una vez establecidas todas las propiedades, el aspecto de su formulario debe ser como el de la figura de la derecha de la siguiente página.

Ya sólo falta teclear el código de la aplicación. Efectúe los pasos que a continuación se detallan:



1. *Realice un doble clic sobre el botón etiquetado **Pesetas a Euros**.*

2. *Teclee el siguiente código:*

```
Private Sub Pesetas_Click()
        ValorEuros.Alignment = 2
        ValorPesetas.Caption = Format(Valor.Text, "#,##0.00")
        ValorPesetas.Alignment = 2
        ValorEuros.Caption = Format(Val(Valor.Text) / 166.386, "#,##0.00")
        Valor.Text = ""
        Valor.SetFocus
End Sub
```

*La función **Format** devuelve un tipo **Variant (String)** que contiene una expresión formateada de acuerdo a las instrucciones contenidas en una expresión de formato. Su **sintaxis** es **Format(expresión[, formato[, primerdíadesemana[, primerdíadeaño]]]).***

***expresión** es cualquier expresión válida.*

***formato** es una expresión de formato definida por el usuario o con nombre válida.*

***primerdíadesemana** es una constante que especifica el primer día de la semana.*

***primerdíadeaño** es una constante que especifica la primera semana del año.*

3. *Realice un doble clic sobre el botón etiquetado **Euros a Pesetas**.*

4. *Teclee el siguiente código:*

```
Private Sub Euros_Click()
```

```
        ValorEuros.Alignment = 2
        ValorEuros.Caption = Format(Valor.Text, "#,##0.00")
        ValorPesetas.Alignment = 2
        ValorPesetas.Caption = Format(Val(Valor.Text) * 166.386, "#,##0.00")
        Valor.Text = ""
        Valor.SetFocus
End Sub
```

*La función **Val** devuelve los números contenidos en una cadena como un valor numérico del tipo adecuado. Su sintaxis es **Val(cadena)**.*

*La función **Val** deja de leer la cadena en el primer carácter que no puede reconocer como parte de un número. Los símbolos y caracteres que se consideran a menudo parte de valores numéricos, como signos de moneda y comas, no se reconocen. Sin embargo, la función reconoce los prefijos de base &O (para octal) y &H (para hexadecimal). Los espacios en blanco, los tabuladores y los avances de línea se eliminan del argumento.*

*La función **Val** sólo reconoce el punto (.) como separador decimal válido.*

*El método **SetFocus** mueve el enfoque al control o al formulario especificado.*

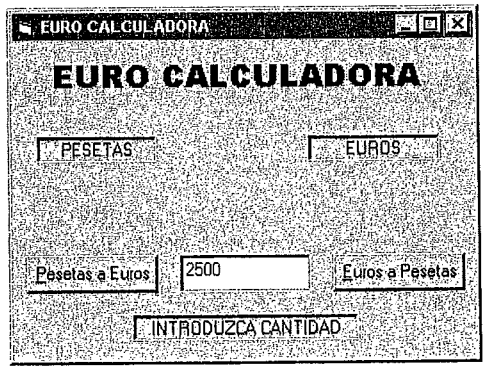
*El **enfoque** es la capacidad para recibir clics del ratón o entradas por teclado en cualquier momento. En el entorno de Microsoft Windows, solamente una ventana, un formulario o un control tienen esta capacidad simultáneamente. El objeto que tiene el foco se indica normalmente por un título o una barra de títulos resaltados. Se puede establecer el foco tanto por el usuario como por la aplicación.*

5. *Archive de nuevo el proyecto haciendo clic en el botón* .

6. *Haga clic sobre el botón* .

Observe como nada más ejecutar la aplicación el cursor aparece en el interior del cuadro de texto esperando que teclee una cantidad. Explicaremos su funcionamiento.

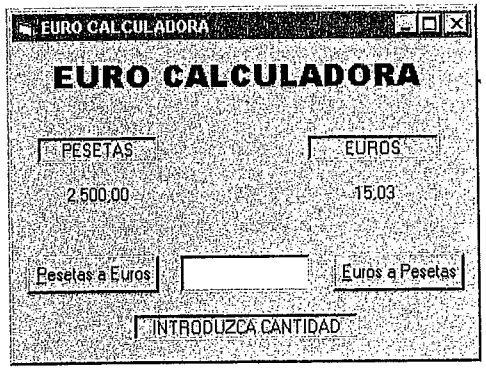
Una vez tecleada la cantidad a convertir sólo es necesario pulsar en el botón correspondiente (**Pesetas a Euros** para convertir la cantidad teclada a euros o **Euros a Pesetas** para convertir la cantidad tecleada a pesetas). Después de haberse realizado la conversión, la cantidad que se había tecleado con anterioridad para convertir desaparece del cuadro de texto y queda listo para realizar la próxima conversión. Veámoslo con un ejemplo.

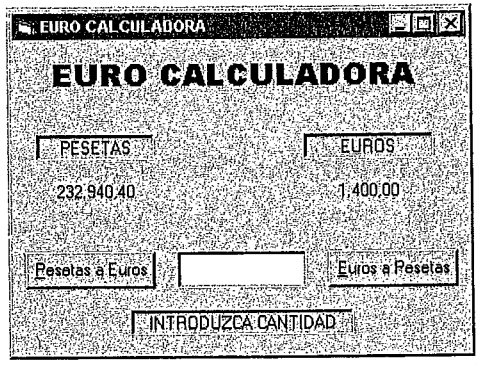


1. *Teclee **2500** en el cuadro de texto.*
2. *Pulse el botón **Pesetas a Euros**.*
3. *Teclee **1400** en el cuadro de texto.*
4. *Pulse el botón **Euros a Pesetas**.*

5. *Cierre la aplicación pulsando sobre el botón de cierre.*

En el siguiente capítulo continuaremos explicando más controles para utilizarlos en una aplicación.





6. Controles más usuales

En este capítulo y en el siguiente estudiaremos los controles típicos que se suelen encontrar en cualquier aplicación. Nos referimos a:

- Barra de desplazamiento horizontal.
- Barra de desplazamiento vertical.
- Casilla de verificación.
- Botón de opción.
- Cuadro contenedor.
- Cuadro combinado.
- Cuadro de lista.

Aplicación con barras de desplazamiento y cuadros contenedores

La siguiente aplicación que crearemos consistirá en:

1. Un formulario donde se irán mostrando los caracteres del código ASCII comprendidos entre 32 y 255 por medio de una barra de desplazamiento horizontal.
2. Un cuadro contenedor donde aparecerán los tres colores básicos (Rojo, Verde y Azul) y donde se podrá ir variando el grado de intensidad de los mismos por medio de tres barras de desplazamiento horizontal, para así poder conseguir los distintos colores existentes.
3. Una barra de desplazamiento vertical que permitirá modificar el tamaño del carácter ASCII seleccionado.

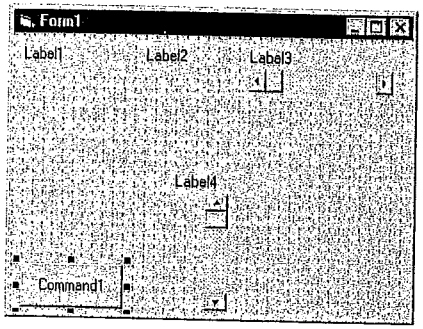
4. Un cuadro contenedor que permitirá visualizar en tamaño y color real el carácter ASCII seleccionado.

   1. *Haga clic sobre el botón* Inicio *de la barra de tareas.*

   2. *Seleccione* ***Programas*** *y a continuación* ***Microsoft Visual Basic 6.0.***

   3. *Por último seleccione la opción* ***Visual Basic 6.0.***

Si tiene creado un icono de acceso directo en el escritorio bastará con hacer doble clic sobre él.



4. *Seleccione de la ventana* ***Nuevo proyecto*** *el tipo de proyecto* ***EXE estándar.***

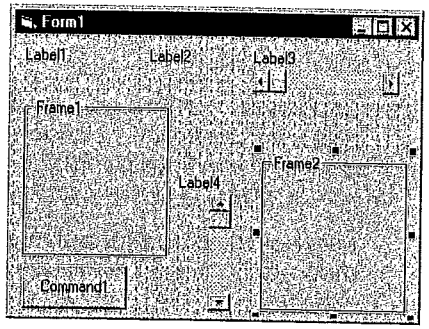
A continuación diseñe un formulario como el de la figura. Para que no le sea difícil saber la posición y el tamaño de cada uno de los controles del formulario, se lo mostraremos en una tabla.

Objeto	Posición	Tamaño
Label1	120, 120	1215 x 255
Label2	1560, 120	495 x 255
Label3	2760, 120	1695 x 255
Label4	1920, 1560	1215 x 255
Command1	120, 2640	1215 x 495

Objeto	Posición	Tamaño
HScroll1	2760, 360	1695 x 255
VScrollVar	2280, 1800	255 x 1335

A continuación hay que colocar las dos cuadros contenedores (**Frames**).

Un control **Frame** proporciona un agrupamiento identificable para controles. También puede usar un control



Frame para subdividir un formulario funcionalmente; por ejemplo, para separar grupos de controles como **OptionButton** como tendrá oportunidad de ver más adelante.

Los controles **Frames** deberá colocarlos en:

Objeto	Posición	Tamaño
Frame1	120, 720	1695 x 1815
Frame2	2880, 1320	1695 x 1815

El paso siguiente consistirá en agrupar dentro del primer *frame* el resto de las barras de scroll.

Para agrupar controles, dibuje primero el control ***Frame*** *y, a continuación, dibuje los controles dentro del* ***Frame****. De este modo podrá mover al mismo tiempo el* ***Frame*** *y los controles que contiene. Si dibuja un control fuera del* ***Frame*** *y después intenta moverlo dentro de éste, el control se colocará sobre el* ***Frame*** *y deberá mover el* ***Frame*** *y los controles por separado.*

1. *Seleccione el control* ***Frame1*** *y haga clic sobre el objeto* *y desplácese al interior del* frame.

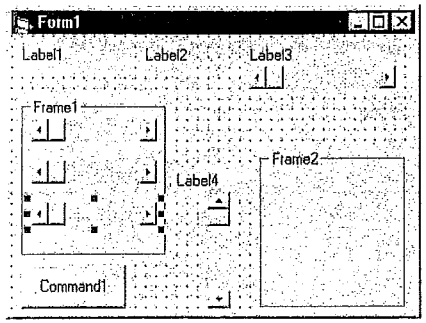
Observe como el puntero del ratón se ha convertido en una cruz (+), ahora ya puede dibujar la barra de scroll.

2. *Repita el mismo paso para dibujar el resto de barras de scroll y de etiquetas.*

El tamaño y posición de las barras de scroll viene indicado en la tabla siguiente:

Objeto	Posición	Tamaño
HScroll2	120, 240	1455 x 255
HScroll3	120, 720	1455 x 255
HScroll4	120, 1200	1455 x 255

En estos momentos el aspecto de su formulario debe ser igual al de la figura de la página siguiente:



Observe que si hace un clic sobre el control **Frame1** y lo mueve de su posición actual, también se mueven los controles situados en su interior. En caso de no ser así compruebe que ha efectuado correctamente los pasos que se detallaban en la página anterior y repítalos de nuevo.

Ahora ya sólo nos falta colocar tres etiquetas en el interior del control **Frame1** para tenerlo completo. Para ello:

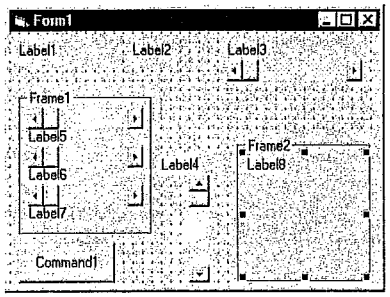
1. *Seleccione el control **Frame1** y haga clic sobre el objeto* A *y desplácese al interior del* frame.
2. *Repita el mismo paso para dibujar el resto de barras de scroll y de etiquetas.*

El tamaño y las posición de las etiquetas es el siguiente:

Objeto	Posición	Tamaño
Label5	120, 480	1455 x 255
Label6	120, 960	1455 x 255
Label7	120, 1440	1455 x 255

*Para seleccionar múltiples controles en un **Frame**, mantenga presionada la tecla CTRL mientras utiliza el mouse (ratón) para dibujar un cuadro alrededor de los controles.*

Para finalizar ya sólo falta colocar una etiqueta dentro del control **Frame2**. Será en esta etiqueta donde se visualice en color y tamaño real el carácter determinado por el código ASCII que se haya seleccionado. La posición y el tamaño de la etiqueta se detallan a continuación:



Objeto	Posición	Tamaño
Label8	120, 240	1455 x 1455

Antes de continuar con el diseño del formulario, almacenaremos lo que hemos hecho hasta ahora. Para ello:

1. *Pulse el botón* [Guardar] *de la barra de herramientas.*
2. *Teclee **Código ASCII** como nombre de formulario y pulse sobre el botón* [Guardar].
3. *Teclee **Código ASCII** como nombre de proyecto y pulse sobre el botón* [Guardar].

Seguidamente pasaremos a especificar las propiedades del formulario y de los controles:

a) Propiedades del formulario (**Form**).

Objeto	Propiedad	Valor
Form1	(Nombre)	Principal
	Caption	Código ASCII

b) Establecer las propiedades de los botones (**CommandButton**).

Objeto	Propiedad	Valor
Command1	(Nombre)	Salir
	Caption	&Salir

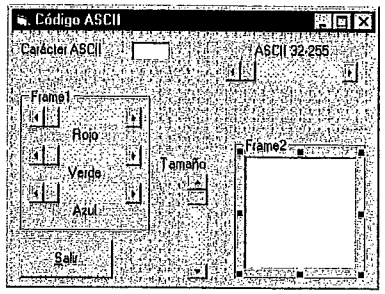
c) Establecer las propiedades de las etiquetas (**Label**).

Objeto	Propiedad	Valor
Label1	(Nombre)	Caracter
	Caption	Carácter ASCII
Label2	(Nombre)	ASCII
	Caption	(Vacío)
	BackColor	&H00FFFFFF&
	BorderStyle	1 - Fixed Single

Objeto	Propiedad	Valor
Label3	(Nombre)	Codigo
	Alignment	2 - Center
	Caption	ASCII 32-255
Label4	(Nombre)	Tamaño
	Caption	Tamaño
Label5	(Nombre)	Rojo
	Alignment	2 - Center
	Caption	Rojo
Label6	(Nombre)	Verde
	Alignment	2 - Center
	Caption	Verde
Label7	(Nombre)	Azul
	Alignment	2 - Center
	Caption	Azul
Label8	(Nombre)	VerASCII
	Alignment	2 - Center
	BackColor	&H00FFFFFF&
	BorderStyle	1 - Fixed Single
	Caption	(Vacío)

En estos momentos el aspecto del formulario debe ser como el de la figura de la derecha.

Para finalizar con el diseño del formulario estableceremos las propiedades de los cuadros contenedores y de las barras de desplazamiento.



d) Establecer las propiedades de los cuadros contenedores (**Frame**).

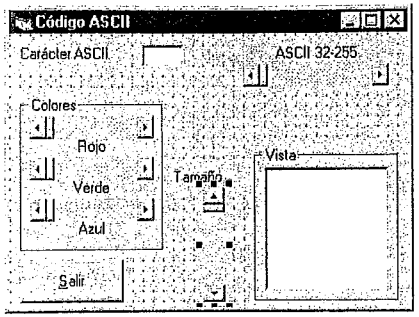
Objeto	Propiedad	Valor
Frame1	(Nombre)	Colores
	Caption	Colores
Frame2	(Nombre)	Vista
	Caption	Vista

e) Establecer las propiedades de las barras de desplazamiento (**HScrollBar** y **VScrollBar**).

Objeto	Propiedad	Valor
HScroll1	(Nombre)	BarraCodigo
	Max	255
	Min	32
HScroll2	(Nombre)	ColorRojo
	Max	255
	Min	0
HScroll3	(Nombre)	ColorVerde
	Max	255
	Min	0
HScroll4	(Nombre)	ColorAzul
	Max	255
	Min	0
VScroll1	(Nombre)	TamCaracter
	Max	80
	Min	6

INSTITUTO DE EDUCACION SECUNDARIA "ARROYO HARINA" ALMENDRALEJO - (Badajoz) -

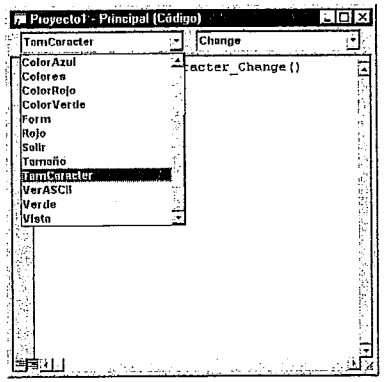
Una vez especificadas todas las propiedades del formulario, éste debe presentar un aspecto como el de la figura de la izquierda de la página siguiente:



Ahora ya sólo falta teclear el código necesario para cada uno de los controles del formulario. Para ello:

1. *Haga clic sobre el control* .
2. *Seleccione de la lista desplegable de objetos el objeto* ***Form***.

Lo que vamos a realizar a continuación es llamar al evento **Form_Load()** y especificar en *tiempo de ejecución* el tipo de letra en que debe aparecer el carácter ASCII en el recuadro **Vista**.



3. *Teclee el siguiente código:*

```
Private Sub Form_Load()
        'Tipo de letra en Vista
        VerASCII.FontName = _
        "Times New Roman"
End Sub
```

IMPORTANTE!!

*La función **Form_Load()** se ejecuta automáticamente cada vez que se inicia una aplicación. Esta función es útil para inicializar variables, abrir bases de datos con tablas, inicializar listas, etc.*

4. *Seleccione de la lista desplegable de objetos el objeto* ***Salir***.
5. *Teclee el siguiente código:*

```
Private Sub Salir_Click()
        End    'Finalización del programa
End Sub
```

6. *Seleccione de la lista desplegable de objetos el objeto* ***BarraCodigo***.

7. *Teclee el siguiente código:*

```
'Se asigna a la propiedad Caption de las etiquetas ASCII y
'VerASCII el carácter que tiene como valor la barra de código
Private Sub BarraCodigo_Change()
        ASCII.Caption = Chr(BarraCodigo.Value)
        VerASCII.Caption = Chr(BarraCodigo.Value)
End Sub
```

IMPORTANTE!!

*La función **Chr** devuelve una cadena que contiene el carácter asociado al código de carácter especificado. Su sintaxis es **Chr(CharCode)**.*

8. *Seleccione de la lista desplegable de objetos el objeto* ***TamCaracter***.
9. *Teclee el siguiente código:*

```
Private Sub TamCaracter_Change()
        VerASCII.FontSize = TamCaracter.Value   'Tamaño del carácter
End Sub
```

10. *Seleccione de la lista desplegable de objetos el objeto* ***ColorRojo***.
11. *Teclee el siguiente código:*

```
Private Sub ColorRojo_Change()
        VerASCII.ForeColor = RGB(ColorRojo.Value, ColorVerde.Value, _
                ColorAzul.Value)    'Intensidad de color Rojo
End Sub
```

12. *Seleccione de la lista desplegable de objetos el objeto* ***ColorVerde***.
13. *Teclee el siguiente código:*

```
Private Sub ColorVerde_Change()
        VerASCII.ForeColor = RGB(ColorRojo.Value, ColorVerde.Value, _
                ColorAzul.Value)    'Intensidad de color Verde
End Sub
```

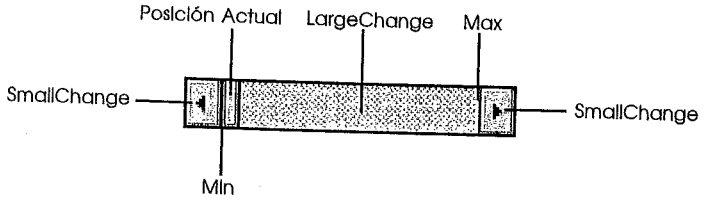
14. *Seleccione de la lista desplegable de objetos el objeto* ***ColorAzul****.*

15. *Teclee el siguiente código:*

```
Private Sub ColorAzul_Change()
        VerASCII.ForeColor = RGB(ColorRojo.Value, ColorVerde.Value, _
                ColorAzul.Value)  'Intensidad de color Azul
End Sub
```

*La propiedad **Max** de los controles **HScrollBar** y **VScrollBar** devuelve o establece el valor máximo de la propiedad **Value** cuando el cuadro de desplazamiento se encuentra en el extremo inferior o derecho.*

*La propiedad **Min** de los controles **HScrollBar** y **VScrollBar** devuelve o establece el valor mínimo de la propiedad **Value** cuando el cuadro de desplazamiento se encuentra en el extremo superior o izquierdo.*



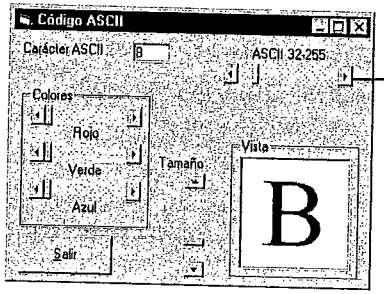
*La propiedad **LargeChange** devuelve o establece el cambio que se producirá en el valor de la propiedad **Value** de un control de barra de desplazamiento (**HScrollBar** o **VScrollBar**) cuando el usuario haga clic en el área situada entre el cuadro de desplazamiento y la flecha de desplazamiento.*

*La propiedad **SmallChange** devuelve o establece el cambio que se producirá en el valor de la propiedad **Value** de un control de barra de desplazamiento (**HScrollBar** o **VScrollBar**) cuando el usuario haga clic en la flecha de desplazamiento.*

16. *Ejecute la aplicación haciendo clic sobre el botón* .

Observe que cuando efectúa un clic en un botón de los extremos (*SmallChange*) de cualesquiera de las barras de desplazamiento (*HScrollBar* y *VScrollBar*) o en una zona intermedia (*LargeChange*) de las mismas, el incremento que se produce es el mismo.

Para entenderlo mejor veámoslo con un ejemplo:



1. *Seleccione el carácter ASCII correspondiente a la letra* ***"B"****.*

2. *Efectúe un clic en el botón derecho de la barra de desplazamiento correspondiente al código ASCII.*

Observe como el carácter que aparece corresponde a la letra **"C"**. Es decir ha avanzado una posición.

3. *Efectúe un clic en el botón izquierdo de la barra de desplazamiento correspondiente al código ASCII.*

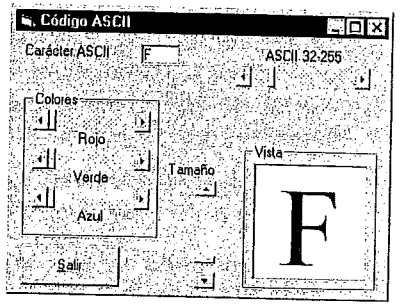
Observe como vuelve a aparecer la letra **"B"**. Es decir ha retrocedido una posición.

4. *Haga clic en una zona vacía de la parte derecha de la barra de desplazamiento.*

Observe como aparece de nuevo el carácter de la letra **"C"**.

5. *Haga clic en una zona vacía de la parte izquierda de la barra de desplazamiento.*

Fíjese como vuelve a aparecer la letra **"B"**. Esto es debido a que el valor que tienen por defecto las propiedades **SmallChage** y **LargeChange** es **1**. A continuación efectuaremos una pequeña modificación en la aplicación para que cuando se efectúe un clic en una zona vacía de las barras de desplazamiento el incremento sea **5**.



6. *Seleccione el control* ***BarraCodigo*** *y teclee* **5** *en la propiedad* ***LargeChange****.*

7. *Ejecute de nuevo la aplicación.*

8. *Seleccione el carácter ASCII correspondiente a la letra* **"F"** .

9. *Efectúe un clic en una zona vacía de la parte derecha de la barra de desplazamiento del código ASCII.*

Observe como el carácter que aparece corresponde a la letra **"K"**. Es decir ha avanzado cinco posiciones.

10. *Efectúe un clic en una zona vacía de la parte izquierda de la barra de desplazamiento correspondiente al código ASCII.*

De nuevo aparece la letra **"K"**. Seguidamente modificaremos la propiedad **LargeChange** de todos los controles de barras de desplazamiento del formulario.

11. *Seleccione el control* ***ColorRojo*** *y teclee* **5** *en la propiedad* ***LargeChange***.

12. *Seleccione el control* ***ColorVerde*** *y teclee* **5** *en la propiedad* ***LargeChange***.

13. *Seleccione el control* ***ColorAzul*** *y teclee* **5** *en la propiedad* ***LargeChange***.

14. *Seleccione el control* ***TamCaracter*** *y teclee* **5** *en la propiedad* ***LargeChange***.

15. *Guarde de nuevo el proyecto.*

Llegados a este punto, se podría dar por finalizada la aplicación, sin embargo vamos a mejorarla.

Mejorando la aplicación

Comenzaremos por colocar en las etiquetas (*Label*) y en el botón (*CommandButton*) un rótulo de ayuda de modo que cuando sitúe el puntero del ratón sobre el rótulo de una etiqueta o de un botón aparezca un texto explicativo debajo del mismo.

1. *Seleccione el control* ***Rojo*** *y teclee:* **Intensidad de color Rojo** *en la propiedad* ***ToolTipText***.

2. *Seleccione el control* ***Verde*** *y teclee:* **Intensidad de color Verde** *en la propiedad* ***ToolTipText***.

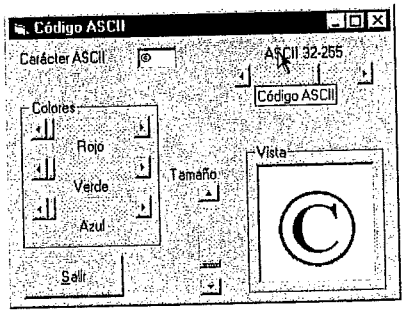
3. *Seleccione el control* ***Azul*** *y teclee:* **Intensidad de color Azul** *en la propiedad* ***ToolTipText***.

4. *Seleccione el control* ***Salir*** *y teclee:* **Finaliza la aplicación** *en la propiedad* ***ToolTipText***.

5. *Seleccione el control* ***Tamaño*** *y teclee:* **Tamaño de la letra** *en la propiedad* ***ToolTipText***.

6. *Seleccione el control* ***VerASCII*** *y teclee:* **Aspecto real** *en la propiedad* ***ToolTipText***.

7. *Seleccione el control* ***Codigo*** *y teclee:* **Código ASCII** *en la propiedad* ***ToolTipText***.



8. *Ejecute la aplicación pulsando la tecla* **F5**.

9. *Sitúe el puntero del ratón encima del rótulo* **ASCII 32-255**.

Observe en la figura de la izquierda como debajo del puntero aparece el texto explicativo que tecleó con anterioridad.

Compruebe que funcione correctamente en el resto de los controles la propiedad **ToolTipText**.

IMPORTANTE !!

La propiedad ***ToolTipText*** *permite que aparezca un texto explicativo debajo del puntero del ratón, cuando éste se sitúa sobre un control que disponga de esta propiedad y se le haya asignado un valor.*

Para finalizar realizaremos una pequeña modificación en la aplicación de modo que cuando efectuemos un clic sobre cualquiera de las barras de scroll de la aplicación, aparezca en una etiqueta el valor que tiene cada una de ellas. Para ello siga los pasos que se detallan a continuación:

1. *Detenga el programa si lo tiene en funcionamiento.*

2. *Active el formulario en modo diseño.*

3. *Inserte un nuevo control con las siguientes características:*

Objeto	Posición	Tamaño
Label1	2880, 720	1455 x 255

Objeto	Propiedad	Valor
Label1	(Nombre)	Valor
	Caption	(Vacío)
	Alignment	2 - Center

4. *Active la ventana de Código y seleccione el objeto* ***Form***.

5. *Teclee el siguiente código:*

```
Private Sub Form_Load()
    'Tipo de letra en Vista
    VerASCII.FontName = _
    "Times New Roman"
    'Valor del color Rojo
    Rojo.Caption = "Rojo - " + Str(ColorRojo.Value)
    'Valor del color Verde
    Verde.Caption = "Verde - " + Str(ColorVerde.Value)
    'Valor del color Azul
    Azul.Caption = "Azul - " + Str(ColorAzul.Value)
    'Valor del Tamaño
    Tamaño.Caption = "Tamaño - " + Str(TamCaracter.Value)
    'Valor ASCII
    Valor.Caption = "Valor ASCII - " + Str(BarraCodigo.Value)
End Sub
```

IMPORTANTE !!

*La función **Str** convierte un valor numérico en una cadena de caracteres. Su sintaxis es **Str(Number)**.*

*Nótese que se usa el carácter de concatenación (+) para así poder unir una cadena de texto junto con un valor numérico que ha sido convertido en carácter previamente mediante el uso de la función **Str**.*

Al añadir este código lo que se hace es inicializar los valores de las etiquetas cada vez que se ejecuta la aplicación.

6. *Seleccione el objeto* ***BarraCodigo***.

7. *Teclee el siguiente código:*

```
'Se asigna a la propiedad Caption de la etiqueta ASCII
'el carácter que tiene como valor la barra de código
Private Sub BarraCodigo_Change()
    ASCII.Caption = Chr(BarraCodigo.Value)
    VerASCII.Caption = Chr(BarraCodigo.Value)
    'Valor ASCII
    Valor.Caption = "Valor ASCII - " + Str(BarraCodigo.Value)
End Sub
```

8. *Seleccione el objeto* ***ColorRojo***.

9. *Teclee el siguiente código:*

```
Private Sub ColorRojo_Change()
    VerASCII.ForeColor = RGB(ColorRojo.Value, ColorVerde.Value, _
        ColorAzul.Value)    'Intensidad de color Rojo
    'Valor del color Rojo
    Rojo.Caption = "Rojo - " + Str(ColorRojo.Value)
End Sub
```

10. *Seleccione el objeto* ***ColorVerde***.

11. *Teclee el siguiente código:*

```
Private Sub ColorVerde_Change()
    VerASCII.ForeColor = RGB(ColorRojo.Value, ColorVerde.Value, _
        ColorAzul.Value)    'Intensidad de color Verde
    'Valor del color Verde
    Verde.Caption = "Verde - " + Str(ColorVerde.Value)
End Sub
```

12. *Seleccione el objeto* ***ColorAzul***.

13. *Teclee el siguiente código:*

```
Private Sub ColorAzul_Change()
    VerASCII.ForeColor = RGB(ColorRojo.Value, ColorVerde.Value, _
        ColorAzul.Value)    'Intensidad de color Verde
    'Valor del color Azul
```

Change → Evento.

5103


```
        Azul.Caption = "Azul - " + Str(ColorAzul.Value)
    End Sub
```

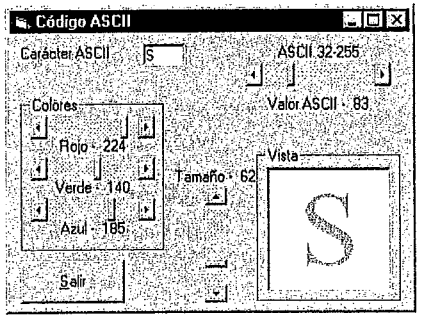
14. *Seleccione el objeto* ***TamCaracter****.*

15. *Teclee el siguiente código:*

```
    Private Sub TamCaracter_Change()
        VerASCII.FontSize = TamCaracter.Value   'Tamaño del carácter
        'Valor del Tamaño
        Tamaño.Caption = "Tamaño - " + Str(TamCaracter.Value)
    End Sub
```

16. *Almacene de nuevo el proyecto.*

17. *Ejecute de nuevo la aplicación.*



Fíjese como ahora aparece el valor que tiene cada una de las etiquetas junto con su rótulo correspondiente. También puede comprobar que cuando efectúa un clic en una zona vacía de las barras de desplazamiento el incremento es de cinco.

Con este ejemplo daremos por finalizado el ejemplo de aplicación con barras de desplazamiento y cuadros contenedores.

7. Matrices

95

Si ha programado con anterioridad en algún otro lenguaje de programación, ya sabrá lo que es una matriz. Pero no se preocupe si no es este su caso ya que en este apartado le explicaremos qué son y cómo funcionan las matrices.

IMPORTANTE !!

Una matriz es un conjunto de valores de datos llamados elementos, a los que se hace referencia mediante un número. Dado que las matrices existen en memoria, proporcionan un rápido acceso a los datos y son fáciles de manipular.

Se pueden especificar, situar o manipular elementos en una matriz de forma muy sencilla. Las matrices le permiten hacer referencia por el mismo nombre a una serie de variables y usar un número denominado índice para distinguirlas. Esto le ayuda a crear código más pequeño y simple. Las matrices tienen un límite superior e inferior y los elementos son contiguos dentro de esos límites.

Hay tres formas de declarar una matriz de tamaño fijo, dependiendo del alcance que desee tenga la matriz:

- Para declarar una *matriz pública*, utilice la instrucción **Public** en la sección declaraciones de un módulo.
- Para declarar una *matriz a nivel de módulo*, use la instrucción **Private** en la sección declaraciones de un módulo.
- Para crear una *matriz local*, emplee la instrucción **Private** en un procedimiento.

Existen tres tipos de matrices.

Matrices unidimensionales

Este tipo de matrices tienen un sólo índice, es decir se componen de un número determinado de filas.

Posición	*TABLA*	*Codificación*
1		Tabla (1)
2		Tabla (2)
3		Tabla (3)
...		...
n		Tabla (n)

Las matrices tienen un límite inferior y superior o lo que es lo mismo, el número mínimo y el número máximo de elementos que puede contener.

Todos los elementos de una matriz tienen el mismo tipo de datos y se declaran del mismo modo que las variables y las constantes. Algunos ejemplos de declaración válidos son los siguientes:

```
Dim Tabla (10) As Integer          'Matriz de 11 elementos 0-10
Dim Valores (1 To 5) As String     'Matriz de 5 elementos 1-5
```

Cuando se trabaja con matrices, el uso de los bucles es muy útil, sobre todo a la hora de recorrer una matriz, ya sea para inicializarla con un valor determinado, borrar su contenido, imprimir su contenido o buscar un valor dentro de ella. Veamos algunos ejemplos:

a) Inicialización de una matriz.

```
Dim Tabla (5) As Integer          'Matriz de 6 elementos 0-5
Dim i As Integer
For i = 0 To 5
        Tabla (i) = i
Next
```

b) Borrado de una matriz.

```
Dim Tabla (5) As Integer          'Matriz de 6 elementos 0-5
Dim i As Integer
For i = 0 To 5
        Tabla (i) = 0
Next
```

c) Impresión de una matriz.

```
Dim Tabla (5) As Integer          'Matriz de 6 elementos 0-5
```



```
Dim i As Integer
For i = 0 To 5
        Print Tabla (i)
Next
```

d) Búsqueda de un valor dentro de la matriz.

```
Dim Tabla (5) As Integer          'Matriz de 6 elementos 0-5
Dim i As Integer
For i = 0 To 5
        If Tabla (i) = 6 Then             'El elemento a buscar es el 6
                Print "ELEMENTO ENCONTRADO"
        End If
Next
```

Matrices multidimensionales

Este tipo de matrices son las que se componen de una serie de filas y una serie de columnas. Veamos el ejemplo de una matriz bidimensional:

Posición	*TABLA*		*Codificación*
1	(1, 1)	(1, 2)	Tabla (1, 1) Tabla (1, 2)
2	(2, 1)	(2, 2)	Tabla (2, 1) Tabla (2, 2)
3	(3, 1)	(3, 2)	Tabla (3, 1) Tabla (3, 2)
...	...	...	
n	(n, 1)	(n, 2)	Tabla (n, 1) Tabla (n, 2)

Algunos ejemplos de declaración válidos son los siguientes:

```
Dim Tabla (2, 10) As Integer          'Matriz de 3 Filas (0-2) y 11 Columnas (0-10)
Dim Valores (1 To 5,  1 To 2) As Byte 'Matriz de 5 Filas (1-5) y 2 Columnas (1-2)
```

El modo de trabajar con una matriz de dos dimensiones es similar al de una matriz unidimensional, sólo que es necesario el uso de dos bucles. Veamos algunos ejemplos:

a) Inicialización de una matriz.

```
Dim Tabla (5, 2) As Integer          'Matriz de 6 filas y 3 columnas
Dim i, j As Integer
For i = 0 To 5
        For j = 0 To 2
                Tabla (i, j) = i + j
        Next j
Next i
```

b) Borrado de una matriz.

```
Dim Tabla (5, 2) As Integer          'Matriz de 6 filas y 3 columnas
Dim i, j As Integer
For i = 0 To 5
        For j = 0 To 2
                Tabla (i, j) = 0
        Next j
Next i
```

c) Impresión de una matriz.

```
Dim Tabla (5, 2) As Integer          'Matriz de 6 filas y 3 columnas
Dim i, j As Integer
For i = 0 To 5
        For j = 0 To 2
                Print "Elemento ", i, ",", j, " = ", Tabla (i, j)
        Next j
Next i
```

d) Búsqueda de un valor dentro de la matriz.

```
Dim Tabla (5, 2) As Integer          'Matriz de 6 filas y 3 columnas
Dim i, j As Integer
For i = 0 To 5
        For j = 0 To 2
                If Tabla (i, j) = 6 Then            'El elemento a buscar es el 6
                        Print "ELEMENTO ENCONTRADO"
                End If
        Next j
Next i
```


Matrices dinámicas

A veces necesitará saber exactamente lo grande que debe ser una matriz. Puede que desee poder cambiar el tamaño de la matriz en tiempo de ejecución.

IMPORTANTE !!

Una matriz dinámica se puede cambiar de tamaño en cualquier momento ayudándole a administrar de forma eficiente la memoria.

Para crear una matriz dinámica deberá realizar los siguientes pasos:

1. *Declare la matriz con cualquiera de estas instrucciones* ***Public***, ***Static*** *o* ***Dim*** *dependiendo del alcance que quiera que tenga la variable.*

2. *Declare la matriz como dinámica proporcionándole una lista de dimensiones vacía.*

   ```
   Dim Matriz()        'Declaración de la matriz
   ```

3. *Asigne el número real de elementos con la instrucción* ***ReDim***.

   ```
   ReDim Matriz(60)    'Nuevo tamaño de la matriz a 60 elementos
   ```

Preservar el contenido de las matrices dinámicas

Cada vez que utilice la instrucción **ReDim** perderá todos los valores almacenados hasta ese momento en la matriz. Si quiere preservar los datos que contiene la matriz hasta el momento de redimensionarla deberá utilizar la palabra clave **Preserve**.

Si utiliza la palabra clave **Preserve**, puede cambiar el tamaño sólo de la última dimensión de la matriz, y no puede cambiar el número de dimensiones. Por ejemplo, si la matriz tiene sólo una dimensión, puede cambiar el tamaño de esa dimensión porque es la última y única dimensión. Sin embargo, si la matriz tiene dos o más dimensiones, puede cambiar únicamente el tamaño de la última dimensión y mantener los contenidos de la matriz. Veamos un ejemplo:

```
ReDim Matriz(3, 4)
...
ReDim Preserve(3, 7)
```

Todos los elementos de una matriz deben ser del mismo tipo, de no ser así se produciría un error.

Registros

En muchas situaciones se necesita procesar elementos que están relacionados entre sí, pero que no son del mismo tipo. Por ejemplo una fecha consta de un nombre de mes (cadena de caracteres), un día y un año (número entero). A estos elementos se les conoce con el nombre de **Registros**. Visual Basic puede procesar este tipo de datos (*registros*) cuyos elementos pueden ser de diferentes tipos. En los registros se puede almacenar diferente información sobre una persona (nombre, estado civil, fecha de nacimiento, etc...), una empresa (nombre de la empresa, dirección, etc...), etc. Cada elemento se almacena en un campo de registro independiente y se puede hacer referencia a cada uno de los elementos almacenados de un campo por su nombre.

*Los componentes de un registro se denominan **campos**. Cada campo tiene un nombre elegido por el programador.*

Su sintaxis es la siguiente:

(Private | Public)Type *Nombre de variable*

Nombre de elemento (((*subíndices*))) As *tipo*

(*Nombre de elemento* (((*subíndices*))) As *tipo*)

. . .

End Type

Global *Variable registro* As *Nombre de variable*

*La instrucción **Type** sólo puede utilizarse en el nivel de módulo. Una vez que ha declarado un tipo definido por el usuario con la instrucción **Type**, puede declarar una variable de ese tipo en cualquier lugar dentro del alcance de la declaración.*

Un ejemplo de declaración de registro podría ser:

```
Type Datos
   Nombre As String * 20      'Variable de 20 caracteres
   Domicilio As String * 30   'Variable de 30 caracteres
   Edad As Integer            'Variable entera
End Type

Global Persona As Datos
```

Acceso a los campos de un registro

Se puede acceder a cada campo de un registro directamente utilizando la siguiente sintaxis:

Variable de registro.Nombre de elemento

Veamos un ejemplo:

```
'Supongamos que deseamos inicializar el registro Persona con
'unos valores  determinados continuando con el ejemplo anterior
Persona.Nombre = "Lorenzo Sánchez"
Persona.Domicilio = "Rbla. San Sebastián, 23"
Persona.Edad = 34
```

Matriz de registros

Los registros simples se utilizan raramente. En general se agrupan en conjuntos conocidos como *arrays de registros*. Veamos un ejemplo:

```
Type Datos
   Nombre As String * 20      'Variable de 20 caracteres
   Domicilio As String * 30   'Variable de 30 caracteres
   Edad As Integer            'Variable entera
End Type
```

```
Global Persona (1 To 25) As Datos
Global i As Integer

'Inicialización del array con unos valores determinados
For i = 1 To 25
    Persona (i).Nombre = " "
    Persona (i).Domicilio = " "
    Persona(i).Edad = 0
Next
```

En el siguiente apartado crearemos una aplicación donde usaremos una matriz de registros y aparecerán casillas de verificación, botones de opciones, cuadros combinados y cuadros de lista.

Aplicación con casillas de verificación, botones de opción, cuadros contenedores y cuadros de lista

La aplicación que crearemos a continuación tratará de simular la gestión de una librería llevando el control de los libros. Los registros que se introduzcan se grabarán en memoria, de modo que una vez que se finalice la ejecución de la aplicación, éstos se perderán. Más adelante ya se estudiará cómo trabajar con bases de datos.

En esta aplicación también aprenderemos a trabajar con módulos. Para empezar comenzaremos diseñando el formulario.

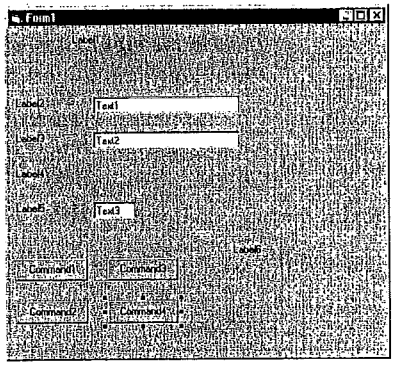
1. *Seleccione de la ventana **Nuevo proyecto** el tipo de proyecto **EXE estándar**.*
2. *Seleccione la propiedad **Height** y teclee el valor **6030**.*
3. *Seleccione la propiedad **Width** y teclee el valor **6600**.*
4. *Modifique el tamaño de la ventana **Proyecto1** para que sea visible todo el formulario.*
5. *Inserte los siguientes controles en la posición y con el tamaño que se indica a continuación:*

Objeto	Posición	Tamaño
Label1	1080, 120	3855 x 615

Objeto	Posición	Tamaño
Label2	120, 1200	1215 x 255
Label3	120, 1800	1215 x 255
Label4	120, 2400	1215 x 255
Label5	120, 3000	1215 x 255
Label6	3840, 3720	1215 x 255
Text1	1440, 1200	2535 x 285
Text2	1440, 1800	2535 x 285
Text3	1440, 3000	735 x 285
Command1	120, 3960	1215 x 375
Command2	120, 4680	1215 x 375
Command3	1680, 3960	1215 x 375
Command4	1680, 4680	1215 x 375

INSTITUTO DE EDUCACION SECUNDARIA

Si ha situado correctamente los controles, su formulario debe presentar el mismo aspecto que la figura de la derecha.



Seguidamente insertaremos dos nuevos controles **ComboBox** y **ListBox** en el formulario.

6. *Inserte los siguientes controles en la posición y con el tamaño que se indica a continuación:*

Objeto	Posición	Tamaño
Combo1	1440, 2400	1695 x 315
List1	3840, 3960	2415 x 1425

Después de haber insertado estos dos controles, el aspecto de su formulario debe ser igual que el de la figura de la página siguiente.

El control de cuadro de lista (**ListBox**) y el de cuadro combinado (**ComboBox**) también presenta al usuario una lista de opciones. De forma

```
ReDim Matriz(3, 4)
...
ReDim Preserve(3, 7)
```

Todos los elementos de una matriz deben ser del mismo tipo, de no ser así se produciría un error.

Registros

En muchas situaciones se necesita procesar elementos que están relacionados entre sí, pero que no son del mismo tipo. Por ejemplo una fecha consta de un nombre de mes (cadena de caracteres), un día y un año (número entero). A estos elementos se les conoce con el nombre de **Registros**. Visual Basic puede procesar este tipo de datos (*registros*) cuyos elementos pueden ser de diferentes tipos. En los registros se puede almacenar diferente información sobre una persona (nombre, estado civil, fecha de nacimiento, etc...), una empresa (nombre de la empresa, dirección, etc...), etc. Cada elemento se almacena en un campo de registro independiente y se puede hacer referencia a cada uno de los elementos almacenados de un campo por su nombre.

*Los componentes de un registro se denominan **campos**. Cada campo tiene un nombre elegido por el programador.*

Su sintaxis es la siguiente:

(Private | Public)Type *Nombre de variable*

Nombre de elemento (((*subíndices*))) **As** *tipo*

(*Nombre de elemento* (((*subíndices*))) **As** *tipo*)

. . .

End Type

Global *Variable registro* **As** *Nombre de variable*

*La instrucción **Type** sólo puede utilizarse en el nivel de módulo. Una vez que ha declarado un tipo definido por el usuario con la instrucción **Type**, puede declarar una variable de ese tipo en cualquier lugar dentro del alcance de la declaración.*

Un ejemplo de declaración de registro podría ser:

```
Type Datos
    Nombre As String * 20        'Variable de 20 caracteres
    Domicilio As String * 30     'Variable de 30 caracteres
    Edad As Integer              'Variable entera
End Type

Global Persona As Datos
```

Acceso a los campos de un registro

Se puede acceder a cada campo de un registro directamente utilizando la siguiente sintaxis:

Variable de registro.Nombre de elemento

Veamos un ejemplo:

```
'Supongamos que deseamos inicializar el registro Persona con
'unos valores  determinados continuando con el ejemplo anterior
Persona.Nombre = "Lorenzo Sánchez"
Persona.Domicilio = "Rbla. San Sebastián, 23"
Persona.Edad = 34
```

Matriz de registros

Los registros simples se utilizan raramente. En general se agrupan en conjuntos conocidos como *arrays de registros*. Veamos un ejemplo:

```
Type Datos
    Nombre As String * 20        'Variable de 20 caracteres
    Domicilio As String * 30     'Variable de 30 caracteres
    Edad As Integer              'Variable entera
End Type
```

```
Global Persona (1 To 25) As Datos
Global i As Integer

'Inicialización del array con unos valores determinados
For i = 1 To 25
    Persona (i).Nombre = " "
    Persona (i).Domicilio = " "
    Persona(i).Edad = 0
Next
```

En el siguiente apartado crearemos una aplicación donde usaremos una matriz de registros y aparecerán casillas de verificación, botones de opciones, cuadros combinados y cuadros de lista.

Aplicación con casillas de verificación, botones de opción, cuadros contenedores y cuadros de lista

La aplicación que crearemos a continuación tratará de simular la gestión de una librería llevando el control de los libros. Los registros que se introduzcan se grabarán en memoria, de modo que una vez que se finalice la ejecución de la aplicación, éstos se perderán. Más adelante ya se estudiará cómo trabajar con bases de datos.

En esta aplicación también aprenderemos a trabajar con módulos. Para empezar comenzaremos diseñando el formulario.

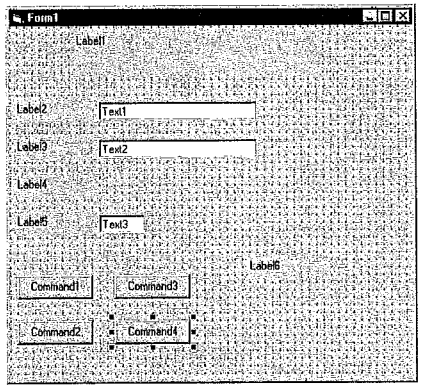
1. *Seleccione de la ventana **Nuevo proyecto** el tipo de proyecto **EXE estándar**.*
2. *Seleccione la propiedad **Height** y teclee el valor **6030**.*
3. *Seleccione la propiedad **Width** y teclee el valor **6600**.*
4. *Modifique el tamaño de la ventana **Proyecto1** para que sea visible todo el formulario.*
5. *Inserte los siguientes controles en la posición y con el tamaño que se indica a continuación:*

Objeto	Posición	Tamaño
Label1	1080, 120	3855 x 615

Objeto	Posición	Tamaño
Label2	120, 1200	1215 x 255
Label3	120, 1800	1215 x 255
Label4	120, 2400	1215 x 255
Label5	120, 3000	1215 x 255
Label6	3840, 3720	1215 x 255
Text1	1440, 1200	2535 x 285
Text2	1440, 1800	2535 x 285
Text3	1440, 3000	735 x 285
Command1	120, 3960	1215 x 375
Command2	120, 4680	1215 x 375
Command3	1680, 3960	1215 x 375
Command4	1680, 4680	1215 x 375

INSTITUTO DE EDUCACIÓN SECUNDARIA "ALMENDRALEJO" (Badajoz)

Si ha situado correctamente los controles, su formulario debe presentar el mismo aspecto que la figura de la derecha.



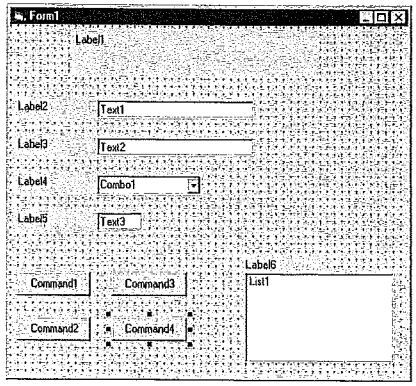
Seguidamente insertaremos dos nuevos controles **ComboBox** y **ListBox** en el formulario.

6. *Inserte los siguientes controles en la posición y con el tamaño que se indica a continuación:*

Objeto	Posición	Tamaño
Combo1	1440, 2400	1695 x 315
List1	3840, 3960	2415 x 1425

Después de haber insertado estos dos controles, el aspecto de su formulario debe ser igual que el de la figura de la página siguiente.

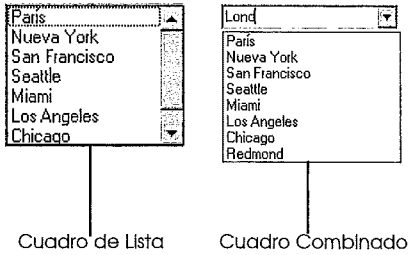
El control de cuadro de lista (**ListBox**) y el de cuadro combinado (**ComboBox**) también presenta al usuario una lista de opciones. De forma



Si el número de elementos es superior al que puede mostrar el cuadro combinado, entonces aparecen automáticamente barras de desplazamiento en el control. El usuario en ese momento se podrá desplazar por la lista de arriba a abajo y/o de izquierda a derecha.

predeterminada, las opciones se muestran en una única columna, si bien se pueden establecer varias columnas.

El cuadro combinado también aúna las características de un cuadro de texto (**TextBox**) y un cuadro de lista (**ListBox**). Este control permite al usuario seleccionar opciones si escribe texto en el cuadro combinado o selecciona un elemento de la lista.

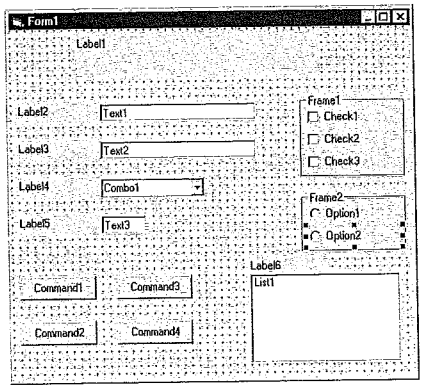


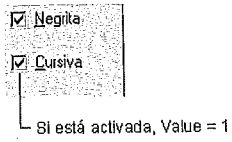
7. *Inserte los siguientes controles en la posición y con el tamaño que se indica a continuación:*

Objeto	Posición	Tamaño
Frame1	4680, 1080	1695 x 1335
Frame2	4680, 2640	1695 x 975
Check1	120, 240	1455 x 255
Check2	120, 600	1455 x 255
Check3	120, 960	1455 x 255
Option1	120, 240	1455 x 255
Option2	120, 600	1455 x 255

1. *Pulse el botón* [ ] *de la barra de herramientas.*

2. *Teclee* ***Biblioteca*** *como nombre de formulario y pulse el botón* [Guardar].
3. *Teclee* ***Biblioteca*** *como nombre de proyecto y pulse el botón* [Guardar].





Una casilla de verificación indica si una condición determinada está activada o desactivada.

*Como las casillas de verificación (**CheckBox**) funcionan independientemente una de otra, el usuario puede activar cualquier número de casillas de verificación al mismo tiempo.*

Los botones de opción (**OptionButton**) presentan al usuario un conjunto de dos o más opciones. Pero, a diferencia de las casillas de verificación, los botones de opción deben funcionar siempre como parte de un grupo; al activar un botón de opción se desactivan inmediatamente todos los demás botones del grupo.

Una vez que tenemos construido el diseño del formulario, pasaremos a establecer cada una las propiedades del formulario y los controles.

a) Propiedades del formulario (**Form**).

Objeto	Propiedad	Valor
Form1	(Nombre)	Biblioteca
	Caption	Gestión Libros

b) Establecer las propiedades de los botones (**CommandButton**).

Objeto	Propiedad	Valor
Command1	(Nombre)	Nuevo
	Caption	&Nuevo

Objeto	Propiedad	Valor
Command2	(Nombre)	Altas
	Caption	&Altas
Command3	(Nombre)	Bajas
	Caption	&Bajas
Command4	(Nombre)	Salir
	Caption	&Salir

c) Establecer las propiedades de las etiquetas (**Label**).

Objeto	Propiedad	Valor
Label1	(Nombre)	Cabecera
	Alignment	2 - Center
	Caption	LIBRERÍA
	Font	Arial Black
	Tamaño Font	28
	ForeColor	&H00C0000&
Label2	(Nombre)	Titulo
	Alignment	2 - Center
	Caption	TÍTULO
Label3	(Nombre)	Autor
	Alignment	2 - Center
	Caption	AUTOR
Label4	(Nombre)	Tema
	Alignment	2 - Center
	Caption	TEMA
Label5	(Nombre)	Pagina
	Alignment	2 - Center
	Caption	Num. PÁGINAS

Objeto	Propiedad	Valor
Label6	(Nombre)	Libros
	Caption	Libros

d) Establecer las propiedades de los cuadros de texto (**TextBox**).

Objeto	Propiedad	Valor
Text1	(Nombre)	TituloL
	Text	(Vacío)
Text2	(Nombre)	AutorL
	Text	(Vacío)
Text3	(Nombre)	PaginasL
	Text	(Vacío)

e) Establecer las propiedades del cuadro combinado (**ComboBox**).

Objeto	Propiedad	Valor
Combo1	(Nombre)	TemaL
	Text	(Vacío)

f) Establecer las propiedades del cuadro de lista (**ListBox**).

Objeto	Propiedad	Valor
List1	(Nombre)	Disponible

g) Establecer las propiedades de los cuadros contenedores (**Frame**).

Objeto	Propiedad	Valor
Frame1	(Nombre)	Formato
	Caption	Formato
Frame2	(Nombre)	Estado
	Caption	Estado

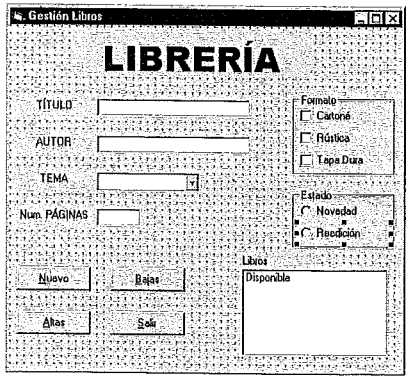
h) Establecer las propiedades de los casillas de verificación (**CheckBox**).

Objeto	Propiedad	Valor
Check1	(Nombre)	Cartone
	Caption	Cartoné
Check2	(Nombre)	Rustica
	Caption	Rústica
Check3	(Nombre)	TapaDura
	Caption	Tapa Dura

i) Establecer las propiedades de los botones de opción (**OptionButton**).

Objeto	Propiedad	Valor
Option1	(Nombre)	Novedad
	Caption	Novedad
Option2	(Nombre)	Reedicion
	Caption	Reedición

4. *Pulse el botón* [icono] *de la barra de herramientas para almacenar la aplicación.*



Si ha efectuado de forma correcta los pasos anteriores el aspecto que debe presentar su formulario debe ser como el que aparece a la izquierda.

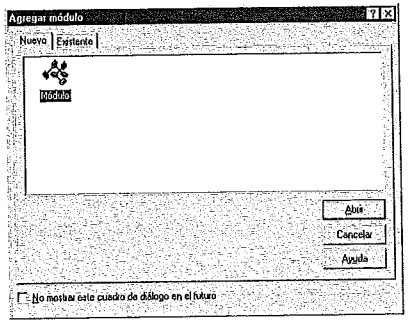
Una vez diseñado el formulario, necesitaremos insertar un módulo para definir el registro y las variables que nos hacen falta.

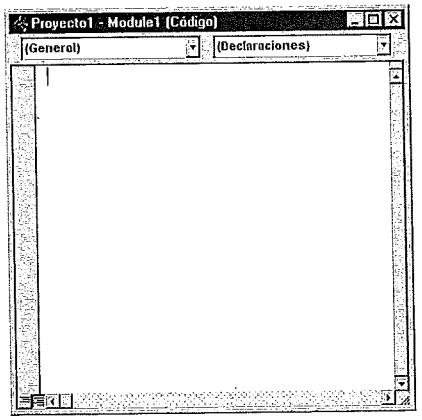
1. *Abra el menú* ***Proyecto*** *y seleccione la opción* ***Agregar módulo***.

La ventana que aparece permite insertar un módulo, ya sea nuevo (pestaña **Nuevo**), o uno que ya exista (pestaña **Existente**).

2. *Efectúe un clic sobre el botón* [Abrir].

Un módulo es un archivo en un proyecto de Visual Basic con extensión .BAS que puede contener descripciones gráficas de un formulario, sus controles y sus valores de propiedades, declaraciones de constantes, variables y procedimientos externos a nivel de formulario, y procedimientos de evento y generales.





Este módulo lo utilizaremos para definir variables globales (variables que se pueden llamar desde cualquier parte de un programa) y crear funciones que no estén relacionadas directamente con ningún evento.

3. *Teclee el siguiente código:*

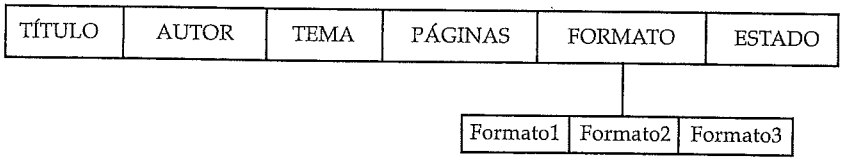
```
Option Explicit

'Declaración del registro
Type Libreria
        TTitulo As String * 25
        TAutor As String * 25
        TTema As String * 15
        TPaginas As Integer
        TFormato (1 To 3) As String * 10   'Tabla de formatos de libro
        TEstado As String * 10
End Type

Global TLibros (1 To 100) As Libreria      'Tabla de registros
Global Reg As Integer                      'Contador de registros
```

La sentencia ***Option Explicit*** *se utiliza para evitar escribir incorrectamente el nombre de una variable existente o para evitar confusiones en el código, donde el alcance de la variable no está claro y también para obligar que las variables estén definidas.*

*La sentencia **Global** permite que las variables **Libros** y **Reg** sean globales a todo el programa, es decir, se pueden utilizar desde cualquier función o formulario de la aplicación.*

Recuerde que al grabar el módulo, deberá asignarle un nombre.

Si representamos gráficamente la estructura del registro que hemos definido veremos que tiene el siguiente aspecto:



4. *Pulse el botón para almacenar el módulo.*
5. *Teclee **Librería** como nombre del módulo y efectúe un clic en Guardar.*
6. *Active la ventana del formulario.*
7. *Haga clic sobre el control.*
8. *Seleccione de la lista desplegable de objetos el objeto **Form**.*
9. *Teclee el siguiente código:*

```
Private Sub Form_Load()
    Reg = 0
    TemaL.AddItem "Acción"
    TemaL.AddItem "Aventuras"
    TemaL.AddItem "Biografía"
    TemaL.AddItem "Ciencia"
    TemaL.AddItem "Ciencia Ficción"
    TemaL.AddItem "Cine"
    TemaL.AddItem "Economía"
    TemaL.AddItem "Gastronomía"
    TemaL.AddItem "Historia"
```



```
    TemaL.AddItem "Informática"
    TemaL.AddItem "Medicina"
    TemaL.AddItem "Misterio"
    TemaL.AddItem "Naturaleza"
    TemaL.AddItem "Policíaco"
    TemaL.AddItem "Política"
    TemaL.AddItem "Romántica"
    TemaL.AddItem "Teatro"
    TemaL.AddItem "Terror"
End Sub
```

Esta manera de insertar los valores en el control **ComboBox**, es la denominada en tiempo de ejecución. Para ello se ha utilizado el método **AddItem** (añadir elemento). Su sintaxis es la siguiente:

***Control*.AddItem *elemento* (, *índice*)**

***elemento** viene representado por la cadena que se desea añadir al control, e **índice** es un entero que especifica la posición dentro del objeto donde se insertará el elemento.*

*Si se omite **índice**, **elemento** se agrega en la posición que le corresponda dentro del orden apropiado (si la propiedad **Sorted** es **True**) o al final de la lista (si **Sorted** es **False**).*

10. *Seleccione de la lista desplegable de objetos el objeto **Nuevo**.*
11. *Teclee el siguiente código:*

```
'Borra el contenido de la ficha inicializando los valores
Private Sub Nuevo_Click()
    TituloL.Text = ""
    AutorL.Text = ""
    PaginasL.Text = ""
    TemaL.Text = ""
    Cartone.Value = 0
    Rustica.Value = 0
    TapaDura.Value = 0
    Novedad.Value = False
    Reedicion.Value = False
```

```
        'Coloca el cursor en la caja de texto del título
        TituloL.SetFocus
End Sub
```

12. *Seleccione de la lista desplegable de objetos el objeto **Altas**.*

13. *Teclee el siguiente código:*

```
'Añade un nuevo libro
Private Sub Altas_Click()
        Reg = Reg + 1
        TLibros(Reg).TTitulo = TituloL.Text
        TLibros(Reg).TAutor = AutorL.Text
        TLibros(Reg).TTema = TemaL.Text
        TLibros(Reg).TPaginas = Val(PaginasL.Text)

        'Comprueba el tipo de formato del libro
        If Cartone.Value = 1 Then
                TLibros(Reg).TFormato(1) = "Cartoné"
        End If
        If Rustica.Value = 1 Then
                TLibros(Reg).TFormato(2) = "Rústica"
        End If
        If TapaDura.Value = 1 Then
                TLibros(Reg).TFormato(3) = "Tapa Dura"
        End If

        'Comprueba el estado del libro
        If Novedad.Value = True Then
                TLibros(Reg).TEstado = "Novedad"
        End If
        If Reedicion.Value = True Then
                TLibros(Reg).TEstado = "Reedición"
        End If

        'Añade el título del libro a la lista
        Disponible.AddItem TituloL.Text
End Sub
```


En el procedimiento anterior realiza las siguientes tareas:

1. Incrementar en 1 el contador de registros **Reg = Reg + 1**.
2. Copiar el contenido de las cajas de texto en la matriz de registros **TLibros(Reg).TTitulo = TituloL.Text**.
3. Añadir el título del libro al cuadro de lista **Disponible.AddItem Titulo.Text**.

Una vez aclarado el funcionamiento del procedimiento **Altas** continuaremos escribiendo el código de la aplicación.

14. *Seleccione de la lista desplegable de objetos el objeto **Disponible**.*

15. *Teclee el siguiente código:*

```
'Busca en la tabla el libro seleccionado de la lista
'para visualizar en la ficha todos sus datos
Private Sub Disponible_Click()
        Dim x As Integer   'Variable contador

        'Inicialización de casillas de verificación
        Cartone.Value = 0
        Rustica.Value = 0
        TapaDura.Value = 0
         For x = 1 To Reg
                If RTrim(Disponible.Text) = RTrim(TLibros(x).TTitulo) Then
                        TituloL.Text = TLibros(x).TTitulo
                        AutorL.Text = TLibros(x).TAutor
                        TemaL.Text = TLibros(x).TTema
                        PaginasL.Text = Str(TLibros(x).TPaginas)

                        'Comprueba el tipo de formato del libro
                        If RTrim(TLibros(x).TFormato(1)) = "Cartoné" Then
                                Cartone.Value = 1
                        End If
                        If RTrim(TLibros(x).TFormato(2)) = "Rústica" Then
                                Rustica.Value = 1
                        End If
                        If RTrim(TLibros(x).TFormato(3)) = "Tapa Dura" Then
                                TapaDura.Value = 1
                        End If
```

```
                'Comprueba el estado del libro
                If RTrim(TLibros(x).TEstado) = "Novedad" Then
                        Novedad.Value = True
                Else
                        Reedicion.Value = True
                End If

                Exit For    'Salida del bucle
        End If
    Next

    Posicion = x    'Guarda la posición del elemento buscado
End Sub
```

El código del procedimiento anterior lo utilizaremos para buscar un registro en la matriz, una vez encontrado lo trasladaremos a la ficha.

*La función **RTrim** devuelve un tipo **Variant (String)** que contiene una copia de una cadena determinada sin espacios a la derecha. Su sintaxis es **RTrim(cadena)**.*

*Si quiere eliminar los espacios de la izquierda, utilice la función **LTrim(cadena)** y si quiere eliminar los de ambos lados **Trim(cadena)**.*

Si observa el código se habrá dado cuenta que aparece una nueva variable (**Posicion**) que todavía no se ha declarado. Esta variable será utilizada para saber la posición en que se encuentra el registro que hemos buscado, para posteriormente utilizarla en el código del botón **Bajas**.

16. *Active el módulo **Libreria**.*

17. *Teclee el siguiente código:*

```
Option Explicit

'Declaración del registro
Type Libreria
        TTitulo As String * 25
        TAutor As String * 25
        TTema As String * 15
        TPaginas As Integer
```



```
        TFormato(1 To 3) As String * 10     'Tabla de formatos de libro
        TEstado As String * 10
End Type

Global TLibros(1 To 100) As Libreria     'Tabla de registros
Global Reg As Integer                    'Contador de registros
Global Posicion As Integer               'Posición registro a borrar
```

18. *Seleccione de la lista desplegable de objetos el objeto **Bajas**.*

19. *Teclee el siguiente código:*

```
'Elimina un elemento de la lista
Private Sub Bajas_Click()
        Disponible.RemoveItem Posicion - 1
        TituloL.Text = ""
        AutorL.Text = ""
        PaginasL.Text = ""
        TemaL.Text = ""
        Cartone.Value = 0
        Rustica.Value = 0
        TapaDura.Value = 0
        Novedad.Value = False
        Reedicion.Value = False
End Sub
```

Esta es la manera de borrar los valores de un control **ComboBox**. Para ello se ha utilizado el método **RemoveItem** (borrar elemento).

Su sintaxis es la siguiente:

***Control*.RemoveItem *elemento* (, *índice*)**

Seguidamente se borra el contenido de los controles de la ficha.

20. *Seleccione de la lista desplegable de objetos el objeto **Salir**.*

21. *Teclee el siguiente código:*

```
Private Sub Salir_Click()
        End    'Finaliza el programa
End Sub
```

Orden de tabulación

Antes de ejecutar el programa nos queda una tarea por realizar que si bien puede parecer una tontería, no deja de ser bastante importante. Se trata de establecer el orden de tabulación de los controles que forman el formulario.

*El **orden de tabulación** es el orden en que un usuario se mueve de un control a otro mediante la tecla **Tab**. Cada formulario tiene su propio orden de tabulación.*

Para cambiar el orden de tabulación de un control es necesario utilizar la propiedad **TabIndex** de éste.

*De forma predeterminada, el primer control dibujado toma el valor **0**, incrementándose el valor del mismo en **1** para el resto de los controles siguiendo el orden en que se hayan insertado.*

El orden de tabulación que estableceremos será el siguiente:

Control	Orden de Tabulación
TituloL	0
AutorL	1
TemaL	2
PaginasL	3
Cartone	4
Rustica	5
TapaDura	6
Novedad	7
Reedicion	8
Nuevo	9
Altas	10

Control	Orden de Tabulación
Bajas	11
Salir	12

1. *Seleccione de la lista desplegable de objetos el objeto **TituloL**.*
2. *Seleccione la propiedad **TabIndex** y teclee **0**.*
3. *Repita los pasos **1** y **2** para el resto de los controles que figuran en la tabla superior.*

El siguiente paso que realizaremos será anular el orden de tabulación del resto de los controles del formulario.

1. *Seleccione de la lista desplegable de objetos el objeto **Disponible**.*
2. *Seleccione la propiedad **TabStop** y elija **False**.*

Observe que el resto de los controles (**Label**, **Frame**) no tienen la propiedad **TabIndex**, ello es debido a que éstos no pueden obtener el enfoque y por lo tanto no disponen de esta propiedad.

*Un grupo de botones de opción tiene una única tabulación. El botón seleccionado como preestablecido tiene la propiedad **TabStop** establecida a **True** y el resto de opciones a **False**.*

*Si desea eliminar un control en el orden de tabulación bastará con establecer a **False** la propiedad **TabStop** de dicho control.*

Pequeños retoques

Llegados a este punto podemos decir que la aplicación ya está terminada, sin embargo vamos a mejorarla un poco más con unos ligeros retoques.

Comenzaremos por colocar en algunos controles un rótulo de ayuda de modo que cuando sitúe el puntero del ratón sobre ellos aparezca un texto explicativo debajo del mismo.

1. *Seleccione el control **TítuloL** y teclee:* **Título del libro** *en la propiedad **ToolTipText**.*
2. *Seleccione el control **AutorL** y teclee:* **Nombre del autor** *en la propiedad **ToolTipText**.*
3. *Seleccione el control **TemaL** y teclee:* **Seleccione el tema del libro** *en la propiedad **ToolTipText**.*
4. *Seleccione el control **PaginasL** y teclee:* **Número de páginas** *en la propiedad **ToolTipText**.*
5. *Seleccione el control **Formato** y teclee:* **Tipo de encuadernación** *en la propiedad **ToolTipText**.*
6. *Seleccione el control **Estado** y teclee:* **Estado del libro** *en la propiedad **ToolTipText**.*
7. *Seleccione el control **Disponible** y teclee:* **Libros disponibles** *en la propiedad **ToolTipText**.*
8. *Seleccione el control **Nuevo** y teclee:* **Introducción de nuevos libros** *en la propiedad **ToolTipText**.*
9. *Seleccione el control **Altas** y teclee:* **Altas de nuevos libros** *en la propiedad **ToolTipText**.*
10. *Seleccione el control **Bajas** y teclee:* **Borrado de libros** *en la propiedad **ToolTipText**.*
11. *Seleccione el control **Salir** y teclee:* **Salir de la aplicación** *en la propiedad **ToolTipText**.*

El siguiente paso consistirá en desactivar el botón **Bajas** y hacer que sólo se active cuando se seleccione algún elemento de la lista **Disponible**. Para ello:

1. *Seleccione de la lista desplegable de objetos el objeto **Form**.*
2. *Teclee el siguiente código:*

```
Private Sub Form_Load()
        Reg = 0
        TemaL.AddItem "Acción"
        TemaL.AddItem "Aventuras"
```



```
        TemaL.AddItem "Biografía"
        TemaL.AddItem "Ciencia"
        TemaL.AddItem "Ciencia Ficción"
        TemaL.AddItem "Cine"
        TemaL.AddItem "Economía"
        TemaL.AddItem "Gastronomía"
        TemaL.AddItem "Historia"
        TemaL.AddItem "Informática"
        TemaL.AddItem "Medicina"
        TemaL.AddItem "Misterio"
        TemaL.AddItem "Naturaleza"
        TemaL.AddItem "Policíaco"
        TemaL.AddItem "Política"
        TemaL.AddItem "Romántica"
        TemaL.AddItem "Teatro"
        TemaL.AddItem "Terror"

        'Desactivación del botón Bajas
        Bajas.Enabled = False
End Sub
```

*La propiedad **Enabled** devuelve o establece un valor que determina si un formulario o un objeto puede responder a eventos generados por el usuario. Su sintaxis es **objeto.Enabled = valor**.*

*El parámetro **valor** puede tomar los valores **True** (activado) o **False** (desactivado).*

Ahora deberemos indicar al programa que el botón **Bajas** deberá activarse cuando seleccionemos algún elemento de la lista **Disponible**.

1. *Seleccione de la lista desplegable de objetos el objeto **Disponible**.*
2. *Teclee el siguiente código:*

```
'Busca en la tabla el libro seleccionado de la lista
'para visualizar en la ficha todos sus datos
Private Sub Disponible_Click()
        Dim x As Integer    'Variable contador
```

```
'Activación del botón Bajas
Bajas.Enabled = True
'Inicialización de casillas de verificación
Cartone.Value = 0
Rustica.Value = 0
TapaDura.Value = 0

 For x = 1 To Reg
        If RTrim(Disponible.Text) = RTrim(TLibros(x).TTitulo) Then
                TituloL.Text = TLibros(x).TTitulo
                AutorL.Text = TLibros(x).TAutor
                TemaL.Text = TLibros(x).TTema
                PaginasL.Text = Str(TLibros(x).TPaginas)

                'Comprueba el tipo de formato del libro
                If RTrim(TLibros(x).TFormato(1)) = "Cartoné" Then
                        Cartone.Value = 1
                End If
                If RTrim(TLibros(x).TFormato(2)) = "Rústica" Then
                        Rustica.Value = 1
                End If
                If RTrim(TLibros(x).TFormato(3)) = "Tapa Dura" Then
                        TapaDura.Value = 1
                End If

                'Comprueba el estado del libro
                If RTrim(TLibros(x).TEstado) = "Novedad" Then
                        Novedad.Value = True
                Else
                        Reedicion.Value = True
                End If

                Exit For    'Salida del bucle
        End If
 Next

 Posicion = x    'Guarda la posición del elemento buscado
End Sub
```


A continuación introduciremos una pequeña modificación en el botón **Bajas** para evitar el borrado accidental de algún libro, para ello haremos que la aplicación nos pida confirmación antes de borrar algún libro. Para lograrlo usaremos la función **MsgBox**.

IMPORTANTE !!.

*La función **MsgBox** muestra un mensaje en un cuadro de diálogo, espera a que el usuario haga clic en un botón y devuelve un tipo **Integer** correspondiente al botón elegido por el usuario.*

La sintaxis de esta función es la siguiente:

MsgBox(*prompt*(, *buttons*)(, *title*)(, *helpfile, context*))

***prompt** es una expresión de cadena que representa el **prompt** en el cuadro de diálogo.*

***buttons** es una expresión numérica que corresponde a la suma de los valores que especifican el número y el tipo de los botones que se pretenden mostrar, el estilo de icono que se va a utilizar, la identidad del botón predeterminado y la modalidad del cuadro de mensajes. Si se omite este argumento, el valor predeterminado para **buttons** es 0.*

***title** es una expresión de cadena que se muestra en la barra de título del cuadro de diálogo. Si se omite **title**, en la barra de título se coloca el nombre de la aplicación.*

***helpfile** es una expresión de cadena que identifica el archivo de Ayuda que se utiliza para proporcionar ayuda interactiva en el cuadro de diálogo. Si se especifica **helpfile**, también se debe especificar **context**.*

***context** es una expresión numérica que es igual al número de contexto de Ayuda asignado por el autor al tema de Ayuda correspondiente. Si se especifica **context**, también se debe especificar **helpfile**.*

El parámetro **buttons** puede tener estos valores:

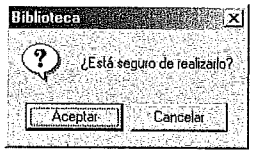
Constante	Valor	Descripción
VbOKOnly	0	Muestra solamente el botón **Aceptar**.
VbOKCancel	1	Muestra los botones **Aceptar** y **Cancelar**.
VbAbortRetryIgnore	2	Muestra los botones **Anular**, **Reintentar** e **Ignorar**.
VbYesNoCancel	3	Muestra los botones **Sí**, **No** y **Cancelar**.

Constante	Valor	Descripción
VbYesNo	4	Muestra los botones **Sí** y **No**.
VbRetryCancel	5	Muestra los botones **Reintentar** y **Cancelar**.
VbCritical	16	Muestra el icono de **mensaje crítico**.
VbQuestion	32	Muestra el icono de **pregunta de advertencia**.
VbExclamation	48	Muestra el icono de **mensaje de advertencia**.
VbInformation	64	Muestra el icono de **mensaje de información**.
VbDefaultButton1	0	El primer botón es el predeterminado.
VbDefaultButton2	256	El segundo botón es el predeterminado.
VbDefaultButton3	512	El tercer botón es el predeterminado.
VbDefaultButton4	768	El cuarto botón es el predeterminado.
VbApplicationModal	0	Aplicación modal; el usuario debe responder al cuadro de mensajes antes de poder seguir trabajando en la aplicación actual.
VbSystemModal	4096	Sistema modal; se suspenden todas las aplicaciones hasta que el usuario responda al cuadro de mensajes.
VbMsgBoxHelpButton	16384	Agrega el botón **Ayuda** al cuadro de mensaje.
VbMsgBoxSetForeground	65536	Especifica la ventana del cuadro de mensaje como la ventana de primer plano.
VbMsgBoxRight	524288	El texto se alínea a la derecha.
VbMsgBoxRtlReading	1048576	Especifica que el texto debe aparecer para ser leído de derecha a izquierda en sistemas hebreo y árabe.

El primer grupo de valores (0 a 5) describe el número y el tipo de los botones mostrados en el cuadro de diálogo; el segundo grupo (16, 32, 48, 64) describe el estilo del icono, el tercer grupo (0, 256, 512) determina el botón predeterminado y el cuarto grupo (0, 4096) determina la modalidad del cuadro de mensajes. Cuando se suman números para obtener el valor final del argumento ***buttons***, se utiliza solamente un número de cada grupo.

Así, por ejemplo, si desea que aparezcan los botones ***Aceptar*** y ***Cancelar*** junto con el icono de ***Pregunta de advertencia*** y con el texto ***¿Está seguro de realizarlo?*** Deberá escribir la siguiente línea de código:

```
MsgBox "¿Está seguro de realizarlo?", 33
```



El valor 33 se obtiene como resultado de sumar 1 (valor de **vbOKCancel**) y 32 (valor de **vbQuestion**).

Si además de que aparezca esta ventana, interesa que se realicen distintas operaciones dependiendo del botón que se haya pulsado, deberemos asignar esta función a una variable. Los valores que puede devolver la función **MsgBox** aparecen en la siguiente tabla:

Constante	Valor	Descripción
vbOK	1	Aceptar
vbCancel	2	Cancelar
vbAbort	3	Anular
vbRetry	4	Reintentar
vbIgnore	5	Ignorar
vbYes	6	Sí
vbNo	7	No

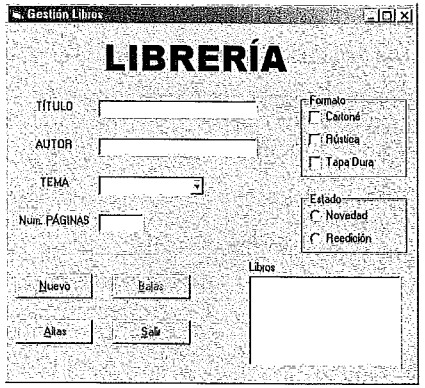
Una vez visto el funcionamiento de esta función pasaremos a modificar el código de la aplicación.

1. *Seleccione de la lista desplegable de objetos el objeto **Bajas**.*

2. *Teclee el siguiente código:*

```
'Elimina un elemento de la lista
Private Sub Bajas_Click()
    Dim Respuesta As Integer  'Almacena el valor de la respuesta
    Respuesta = MsgBox ("¿Está seguro de borrarlo?", vbYesNo, _
    "ATENCIÓN")
```

```
        If Respuesta = 6 Then
            Disponible.RemoveItem Posicion - 1
            TituloL.Text = ""
            AutorL.Text = ""
            PaginasL.Text = ""
            TemaL.Text = ""
            Cartone.Value = 0
            Rustica.Value = 0
            TapaDura.Value = 0
            Novedad.Value = False
            Reedicion.Value = False
        End If

        Bajas.Enabled = False        'Desactivación del botón Bajas
    End Sub
```



3. *Ejecute la aplicación.*

Observe que el botón **Bajas** inicialmente está desactivado como era nuestro propósito.

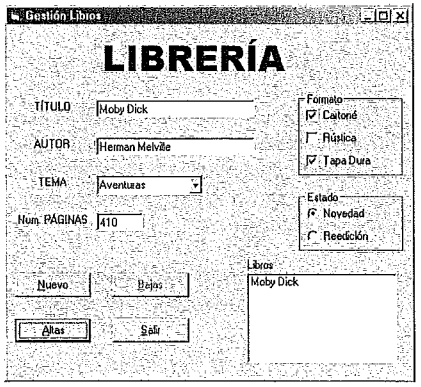
A continuación probaremos si funciona correctamente la aplicación.

4. *Introduzca los siguientes datos:*

Título:	***Moby Dick***
Autor:	***Herman Melville***
Tema:	***Aventuras***
Num. Páginas:	***410***
Formato:	***Cartoné***
	Tapa Dura
Estado:	***Novedad***

5. *Pulse el botón* [Altas] *para dar el libro de alta.*

Observe como automáticamente aparece el título del libro en la lista disponibles.



6. *Pulse el botón* [Nuevo] *para introducir un nuevo libro.*

Observe como aparece una ficha en blanco como al principio.

7. *Introduzca los siguientes datos:*

Título:	***La Tienda***
Autor:	***Stephen King***
Tema:	***Terror***
Num. Páginas:	***820***
Formato:	***Rústica***
	Tapa Dura
Estado:	***Reedición***

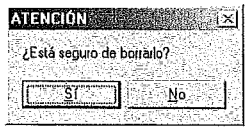
8. *Pulse el botón* [Altas] *para dar el libro de alta.*

Una vez introducidos los dos libros continuaremos probando la aplicación.

9. *Efectúe un clic sobre* ***Moby Dick*** *de la lista libros.*

Observe como en la ficha aparecen todos los datos correspondientes a este libro y como se activa el botón **Bajas**. Si hace otro clic sobre el otro libro, verá como también aparecen sus datos.

10. *Seleccione el libro* ***Moby Dick*** *y haga clic sobre el botón* [Bajas].



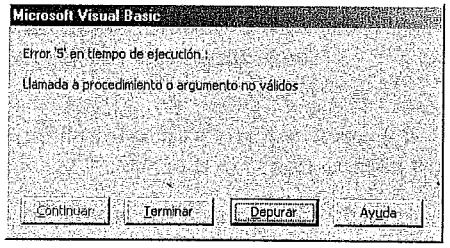
Acto seguido aparece la ventana de confirmación de borrado que se creó.

11. *Pulse el botón* [Sí].

Automáticamente desaparece de la lista **Libros**.

*12. Repita los pasos **10** y **11** con el libro que queda.*

Después de haber contestado afirmativamente a la pregunta de si quería borrar el libro, aparece la pantalla de la derecha con un mensaje de error.



No se preocupe vamos a explicar por qué se produce y lo solucionaremos.

13. Haga clic en el botón Terminar.

El error se produce debido a que cuando borramos un elemento de la lista **Disponible** (mediante el método **RemoveItem**), lo hacemos de verdad, mientras que de la tabla no lo hacemos, de modo que llega un momento en que el número de elementos de la tabla es mayor que el de la lista, al intentar borrar un elemento puede darse el caso de que el valor devuelto por la variable **Posicion** (**Posicion = X**) sea mayor que el número de elementos que hay en realidad en la lista **Disponible**, es entonces cuando se produce el error ya que se intenta borrar un elemento de una posición inexistente.

Para solucionar este problema siga los pasos que se detallan a continuación:

*14. Seleccione de la lista desplegable de objetos el objeto **Disponible**.*

15. Teclee el siguiente código:

```
'Busca en la tabla el libro seleccionado de la lista
'para visualizar en la ficha todos sus datos
Private Sub Disponible_Click()
    Dim x As Integer     'Variable contador
    Dim C As Integer     'Contador de elementos llenos de la tabla

    'Activación del botón Bajas
    Bajas.Enabled = True

    'Inicialización del contador de elementos llenos
    C = 0
```



```
    'Inicialización de casillas de verificación
    Cartone.Value = 0
    Rustica.Value = 0
    TapaDura.Value = 0
    For x = 1 To Reg
        If RTrim(Disponible.Text) = RTrim(TLibros(x).TTitulo) Then
            TituloL.Text = TLibros(x).TTitulo
            AutorL.Text = TLibros(x).TAutor
            TemaL.Text = TLibros(x).TTema
            PaginasL.Text = Str(TLibros(x).TPaginas)

            'Comprueba el tipo de formato del libro
            If RTrim(TLibros(x).TFormato(1)) = "Cartoné" Then
                Cartone.Value = 1
            End If
            If RTrim(TLibros(x).TFormato(2)) = "Rústica" Then
                Rustica.Value = 1
            End If
            If RTrim(TLibros(x).TFormato(3)) = "Tapa Dura" Then
                TapaDura.Value = 1
            End If

            'Comprueba el estado del libro
            If RTrim(TLibros(x).TEstado) = "Novedad" Then
                Novedad.Value = True
            Else
                Reedicion.Value = True
            End If

            'Contador de elementos llenos de la tabla
            If TLibros(x).TTitulo <> "" Then
                C = C + 1
            End If

            Exit For    'Salida del bucle
        End If
    Next

    Posicion = C    'Guarda la posición del elemento buscado
End Sub
```

16. Guarde de nuevo la aplicación.

Si prueba de nuevo la aplicación verá que ahora funciona correctamente, en caso de no ser así, repase las últimas modificaciones efectuadas.

8. Imágenes e iconos en los formularios

En este capítulo aprenderá a insertar imágenes en un formulario, animar y colocar un icono, crear imágenes compuestas y asignar un fondo a un formulario. Para ello deberá usar los siguientes controles:

Control de imagen (Image).

Cuadro de imagen (PictureBox).

El control Image

El control **Image** se utiliza para mostrar gráficos. Este tipo de control puede mostrar un gráfico de los siguientes formatos: mapa de bits, icono, metarchivo, metarchivo mejorado o archivos JPEG o GIF.

Además, los controles Image responden al evento *Click* y se pueden usar como sustitutos de los botones de comando, como elementos de una barra de herramientas o para crear animaciones sencillas.

IMPORTANTE !!

*El control **Image** utiliza menos recursos del sistema y se actualiza con más rapidez que un control **PictureBox**. Sin embargo sólo admite un subconjunto de las propiedades, eventos y métodos del control **PictureBox**. Ambos controles aceptan los mismos formatos de imagen. Sin embargo, en los controles **Image** puede ajustar las imágenes al tamaño del control para que quepan en él. No puede hacer lo mismo con los controles **PictureBox**.*

La imagen real que se muestra viene determinada por la propiedad **Picture**. Ésta contiene el nombre del archivo (y la ruta de acceso si es necesaria) para el archivo de imagen que debe mostrar.

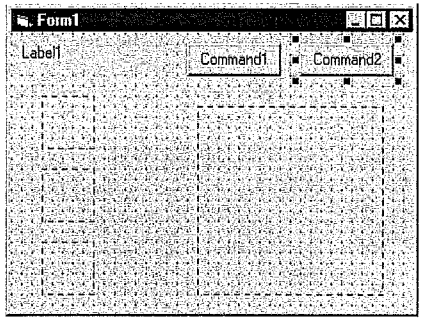
A continuación vamos a diseñar una aplicación para ver el funcionamiento de este control. Realice estos pasos:

1. *Abra el programa de **Visual Basic 6.0** si no lo tenía.*
2. *Abra un **Nuevo proyecto** del tipo **EXE estándar**.*
3. *Inserte los siguientes controles en la posición y con el tamaño que se indica a continuación:*

Objeto	Posición	Tamaño
Label1	120, 120	1575 x 255
Image1	360, 720	615 x 615
Image2	360, 1560	615 x 615
Image3	360, 2400	615 x 615
Image4	2160, 840	2175 x 2175
Command1	2040, 120	1095 x 375
Command2	3360, 120	1095 x 375

Si ha situado correctamente los controles, su formulario debe presentar el mismo aspecto que el de la figura de la derecha.

1. *Pulse el botón de la barra de herramientas.*
2. *Teclee **Imagen** como nombre de formulario y pulse el botón Guardar.*
3. *Teclee **Imagen** como nombre de proyecto y pulse el botón Guardar.*



IMPORTANTE!!

*Tanto a un control **Image** como **PictureBox**, se les puede asignar imágenes en tiempo de diseño o ejecución. En tiempo de diseño deberá utilizar la propiedad **Picture** y en tiempo de ejecución **LoadPicture**.*

Una vez que ha construido el diseño de formulario, establezca las propiedades del formulario y los controles.

a) Propiedades del formulario (**Form**).

Objeto	Propiedad	Valor
Form1	(Nombre)	Imagen
	Caption	Imágenes

b) Establecer las propiedades de los botones (**CommandButton**).

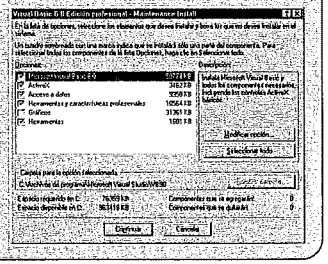
Objeto	Propiedad	Valor
Command1	(Nombre)	Zoom
	Caption	&Zoom
Command2	(Nombre)	Salir
	Caption	&Salir

c) Establecer las propiedades de la etiqueta (**Label**).

Objeto	Propiedad	Valor
Label1	(Nombre)	Titulo
	Caption	Pulse en una imagen

*Si cuando instaló **Visual Basic** lo hizo realizando una instalación **Típica**, observará que dentro de la carpeta **Archivos de programa\Microsoft Visual Studio\Common** no existe ninguna con el nombre de **Graphics** y por lo tanto no podrá efectuar esta aplicación. Si es este su caso, realice los pasos que a continuación se detallan:*

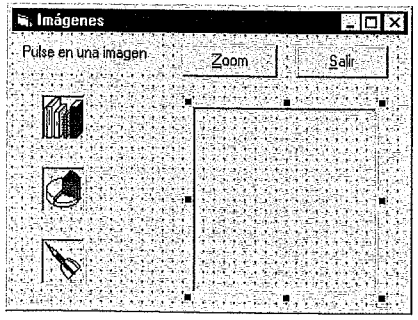
1. *Inserte el CD-Rom de instalación de **Visual Basic 6.0**.*
2. *Ejecute el programa **Setup.Exe** del CD-Rom.*
3. *Seleccione la opción **Agregar o quitar**.*
4. *Haga clic en la opción **Gráficos** para activarla.*
5. *Efectúe un clic sobre el botón Continuar.*



d) Establecer las propiedades de las imágenes (**Image**).

Objeto	Propiedad	Valor
Image1	(Nombre)	Imagen1
	BorderStyle	1 - Fixed Single
	Picture	Carpeta Writing
		Books04.ico
Image2	(Nombre)	Imagen2
	BorderStyle	1 - Fixed Single
	Picture	Carpeta Office
		Graph12.ico
Image3	(Nombre)	Imagen3
	BorderStyle	1 - Fixed Single
	Picture	Carpeta Arrows
		Point10.ico
Image4	(Nombre)	Imagen4
	BorderStyle	1 - Fixed Single

En estos momentos el aspecto del formulario debe ser como el de la figura de la izquierda.



El funcionamiento de la aplicación será el siguiente: una vez ejecutada la aplicación, deberá pulsar en una de las tres imágenes de la parte izquierda del formulario. Acto seguido esta imagen aparecerá en el centro del formulario, de modo que cuando pulse sobre el botón **Zoom** dicha imagen aumentará de tamaño.

A continuación pasaremos a escribir el código necesario para que funcione la aplicación.

1. *Haga clic sobre el control* .

2. *Seleccione de la lista desplegable de objetos el objeto* ***Imagen1***.

3. *Teclee el siguiente código:*

```
'Carga de una imagen e inicialización del tamaño
Private Sub Imagen1_Click()
        Alto = 540
        Ancho = 540
        Imagen4.Height = Alto        'Alto inicial de Imagen4
        Imagen4.Width = Ancho        'Ancho inicial de Imagen4
        Imagen4.Picture = LoadPicture("C:\Archivos de programa\" _
        & "Microsoft Visual Studio\Common\Graphics\Icons\" _
        & "Writing\Books04.ico")
        Zoom.Enabled = True          'Activación del botón Zoom
End Sub
```

*La propiedad **Height** establece la altura de un objeto. La propiedad **Width** establece la anchura de un objeto. La unidad de medida es el **twip** (unidad de medida de la pantalla igual a 1/20 de un punto de impresora).*

No se preocupe por las dos nuevas variables **Alto** y **Ancho** que aparecen en este procedimiento, ya se declararán más adelante.

*La función **LoadPicture** carga un gráfico en un objeto Picture, un control **PictureBox** o un control **Image**. Su sintaxis más sencilla es **LoadPicture ([nombreArchivo])**.*

*Dentro del parámetro **nombreArchivo** se pueden incluir carpetas y unidades. Si no se indica un nombre de archivo, **LoadPicture** borra el control **Image** o **PictureBox**.*

4. *Seleccione de la lista desplegable de objetos el objeto* ***Imagen2***.

5. *Teclee el siguiente código:*

```
'Carga de una imagen e inicialización del tamaño
Private Sub Imagen2_Click()
        Alto = 540
```

```
        Ancho = 540
        Imagen4.Height = Alto        'Alto inicial de Imagen4
        Imagen4.Width = Ancho        'Ancho inicial de Imagen4
        Imagen4.Picture = LoadPicture("C:\Archivos de programa\" _
            & "Microsoft Visual Studio\Common\Graphics\" _
            & "Icons\Office\Graph12.ico")
        Zoom.Enabled = True          'Activación del botón Zoom
End Sub
```

6. *Seleccione de la lista desplegable de objetos el objeto **Imagen3**.*

7. *Teclee el siguiente código:*

```
'Carga de una imagen e inicialización del tamaño
Private Sub Imagen3_Click()
        Alto = 540
        Ancho = 540
        Imagen4.Height = Alto        'Alto inicial de Imagen4
        Imagen4.Width = Ancho        'Ancho inicial de Imagen4
        Imagen4.Picture = LoadPicture("C:\Archivos de programa\" _
            & "Microsoft Visual Studio\Common\Graphics\" _
            & "Icons\Arrows\Point10.ico")
        Zoom.Enabled = True          'Activación del botón Zoom
End Sub
```

Con este código lo que conseguimos es que cuando se efectúe un clic sobre cualquiera de los dibujos de la parte izquierda del formulario, se carguen automáticamente en el centro del mismo para así poder ampliarlos.

8. *Active la ventana **Formulario** haciendo clic sobre el botón* .

9. *Seleccione el control **Imagen4**.*

10. *Seleccione la propiedad **Stretch** y cambie su valor a **True**.*

*La propiedad **Stretch** devuelve o establece un valor que indica si un gráfico cambia su tamaño para ajustarse al de un control Image. Si tiene el valor **True**, el gráfico cambia de tamaño para ajustarse al del control. En cambio si tiene el valor **False** (Predeterminado), el control cambia de tamaño para ajustarse al del gráfico.*

Antes de programar el botón **Zoom** deberemos declarar dos variables globales que controlarán el tamaño máximo que puede alcanzar el zoom de la imagen.

11. *Haga clic sobre el control* .

12. *Seleccione de la lista desplegable de objetos **(General)**.*

13. *Teclee el siguiente código:*

```
'Declaración de variables generales
Dim Alto As Integer, AltoMax As Integer
Dim Ancho As Integer
```

14. *Seleccione de la lista desplegable de objetos el objeto **Form**.*

15. *Teclee el siguiente código:*

```
'Tamaño máximo de la imagen
Private Sub Form_Load()
        AltoMax = 2175

        'Desactivación del botón Zoom
        Zoom.Enabled = False
End Sub
```

16. *Seleccione de la lista desplegable de objetos el objeto **Zoom**.*

17. *Teclee el siguiente código:*

```
Private Sub Zoom_Click()
        Alto = Alto + 15             'Incremento de la altura
        Ancho = Ancho + 15           'Incremento de la anchura
        Imagen4.Height = Alto        'Nueva altura
        Imagen4.Width = Ancho        'Nueva anchura
End Sub
```

18. *Seleccione de la lista desplegable de objetos el objeto **Salir**.*

19. *Teclee el siguiente código:*

```
Private Sub Salir_Click()
        End    'Finaliza el programa
End Sub
```

Si dejáramos el programa tal como está, llegaría un momento en que la imagen sobrepasaría los límites del tamaño del formulario, para solucionar esto:

20. *Seleccione de la lista desplegable de objetos el objeto **Zoom**.*

21. *Teclee el siguiente código:*

```
Private Sub Zoom_Click()
        If AltoMax >= Alto Then
                Alto = Alto + 15                'Incremento de la altura
                Ancho = Ancho + 15              'Incremento de la anchura
                Imagen4.Height = Alto           'Nueva altura
                Imagen4.Width = Ancho           'Nueva anchura
        End If
End Sub
```

22. *Almacene el proyecto y ejecútelo.*

El control PictureBox

Un control **PictureBox** puede mostrar un gráfico a partir de un mapa de bits, un icono o un metarchivo, así como un metarchivo mejorado, un archivo JPEG o un archivo GIF. Este control recorta el gráfico si él mismo no es lo bastante grande para mostrar la imagen completa.

El cuadro de imagen (**PictureBox**) puede considerarse un lienzo en blanco sobre el que se puede pintar, dibujar o imprimir.

La aplicación que desarrollaremos para ver cómo se trabaja con un control **PictureBox**, consistirá en una pequeña animación que moverá un objeto por la pantalla por un tiempo indefinido hasta que detengamos el programa.

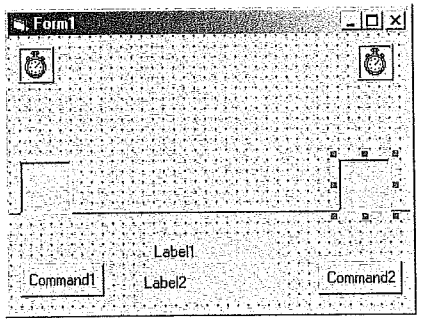
1. *Abra un **Nuevo proyecto** del tipo **EXE estándar**.*

2. *Inserte los siguientes controles en la posición y con el tamaño que se indica a continuación:*

Objeto	Posición	Tamaño
Label1	1680, 2400	1335 x 255
Label2	1560, 2760	1575 x 255
Command1	120, 2640	975 x 375
Command2	3600, 2640	975 x 375
Line	0, 2040	4695 x 15
Picture1	120, 1440	615 x 615
Picture2	3840, 440	615 x 615
Timer1	120, 120	420 x 420
Timer2	4080, 120	420 x 420

Si ha situado correctamente los controles, su formulario debe presentar el mismo aspecto que el de la figura de la derecha.

1. *Pulse el botón de la barra de herramientas.*
2. *Teclee **Animación** como nombre de formulario y pulse el botón Guardar.*
3. *Teclee **Animación** como nombre de proyecto y pulse el botón Guardar.*



Seguidamente estableceremos las propiedades del formulario y los controles.

a) Propiedades del formulario (**Form**).

Objeto	Propiedad	Valor
Form1	(Nombre)	Animacion
	Caption	Animaciones

b) Establecer las propiedades de los botones (**CommandButton**).

Objeto	Propiedad	Valor
Command1	(Nombre)	Iniciar
	Caption	&Iniciar
Command2	(Nombre)	Salir
	Caption	&Salir

c) Establecer las propiedades de las etiquetas (**Label**).

Objeto	Propiedad	Valor
Label1	(Nombre)	Titulo
	Alignment	2 - Center
	Caption	Eje X - Eje Y
Label2	(Nombre)	Coordenadas
	BorderStyle	1 - Fixed Single
	Caption	(Vacío)

d) Establecer las propiedades del temporizador (**Timer**).

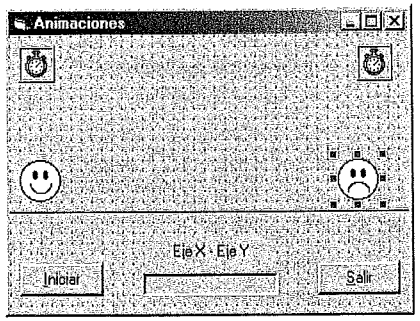
Objeto	Propiedad	Valor
Timer1	(Nombre)	Tiempo1
	Interval	25
Timer2	(Nombre)	Tiempo2
	Interval	25

Un control **Timer** puede ejecutar código a intervalos periódicos produciendo un evento Timer. Este control **Timer**, aunque es invisible para el usuario, resulta útil para el procesamiento de fondo.

*La propiedad **Interval** devuelve o establece el número de milisegundos entre las llamadas al evento Timer de un control **Timer**.*

e) Establecer las propiedades de los cuadros de imagen (**PictureBox**).

Objeto	Propiedad	Valor
Picture1	(Nombre)	Imagen1
	AutoSize	True
	BorderStyle	0 - None
	Picture	Carpeta Misc
		Face02
Picture2	(Nombre)	Imagen2
	AutoSize	True
	BorderStyle	0 - None
	Picture	Carpeta Misc
		Face04



*La propiedad **AutoSize** devuelve o establece un valor que determina si el tamaño de un control cambia automáticamente para presentar todo su contenido. Si tiene el valor **True** cambia automáticamente el tamaño del control para presentar todo el contenido del mismo. Con el valor **False** (valor por defecto) mantiene constante el tamaño del control. El contenido se recorta cuando excede el área del control.*

Una vez realizado el diseño, comenzaremos a escribir el código de la aplicación.

1. *Haga clic sobre el control* .
2. *Seleccione de la lista desplegable de objetos el objeto **Form**.*

3. *Teclee el siguiente código:*

```
Private Sub Form_Load()
        Imagen2.Visible = False      'Oculta la segunda imagen
        Tiempo1.Enabled = False      'Deshabilita el primer temporizador
        Tiempo2.Enabled = False      'Deshabilita el segundo temporizador
End Sub
```

*La propiedad **Enabled** establece un valor que determina si un formulario o un control puede responder a eventos generados por el usuario. Con el valor **True** (valor por defecto) permite que un objeto responda a eventos. En cambio con **False** impide que el objeto responda a eventos.*

Lo que realiza el procedimiento anterior es esconder la imagen de la derecha del formulario, al mismo tiempo deshabilita los dos temporizadores que controlan el movimiento de las dos imágenes.

4. *Seleccione de la lista desplegable de objetos el objeto **Iniciar**.*

5. *Teclee el siguiente código:*

```
Private Sub Iniciar_Click()
        Tiempo1.Enabled = True   'Activa el primer temporizador
End Sub
```

6. *Seleccione de la lista desplegable de objetos el objeto **Timer1**.*

7. *Teclee el siguiente código:*

```
Private Sub Tiempo1_Timer()
        If Imagen1.Left < 2000 Then
                Imagen1.Move Imagen1.Left + 40, Imagen1.Top - 30
        ElseIf Imagen1.Top < 1440 Then
                Imagen1.Move Imagen1.Left + 40, Imagen1.Top + 30
        End If

        If Imagen1.Top = 1440 Then
                Imagen1.Visible = False      'Oculta la primera imagen
                Imagen2.Visible = True       'Muestra la segunda imagen
```



```
                Tiempo1.Enabled = False      'Desactiva el primer temporizador
                Tiempo2.Enabled = True       'Activa el segundo temporizador
                Imagen2.Left = 3840          'Coordenada X de la segunda imagen
                Imagen2.Top = 1440           'Coordenada Y de la segunda imagen
        End If

        'Coordenadas X e Y del dibujo
        Coordenadas.Caption = Str(Imagen1.Left) + " - " + Str(Imagen1.Top)
End Sub
```

Las tareas que se realizan en este procedimiento son:

1. Controlar el desplazamiento horizontal y vertical de la imagen aumentando y disminuyendo el desplazamiento de ambos (**+40** eje X y **-30** eje Y).
2. Cuando llega más o menos al centro del formulario se incrementa el desplazamiento vertical (**+40** eje X y **+30** eje Y).
3. Una vez que la imagen ha realizado todo el recorrido se oculta y se muestra la otra (**Imagen2**) activándose el segundo temporizador (**Tiempo2**).
4. Se muestran constantemente las coordenadas donde se encuentra la imagen.

8. *Seleccione de la lista desplegable de objetos el objeto **Timer2**.*

9. *Teclee el siguiente código:*

```
Private Sub Tiempo2_Timer()
        If Imagen2.Left > 2000 Then
                Imagen2.Move Imagen2.Left - 40, Imagen2.Top - 30
        ElseIf Imagen2.Top < 1440 Then
                Imagen2.Move Imagen2.Left - 40, Imagen2.Top + 30
        End If

        If Imagen2.Top = 1440 Then
                Imagen1.Visible = True       'Muestra la primera imagen
                Imagen2.Visible = False      'Oculta la segunda imagen
                Tiempo1.Enabled = True       'Activa el primer temporizador
                Tiempo2.Enabled = False      'Desactiva el segundo temporizador
                Imagen1.Left = 120           'Coordenada X de la primera imagen
```

```
            Imagen1.Top = 1440          'Coordenada Y de la primera imagen
        End If

        'Coordenadas X e Y del dibujo
        Coordenadas.Caption = Str(Imagen2.Left) + " - " + Str(Imagen2.Top)
    End Sub
```

El funcionamiento de este procedimiento es el mismo que el del anterior pero a la inversa.

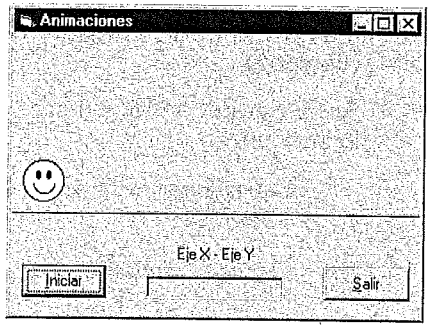
10. *Seleccione de la lista desplegable de objetos el objeto* ***Salir****.*

11. *Teclee el siguiente código:*

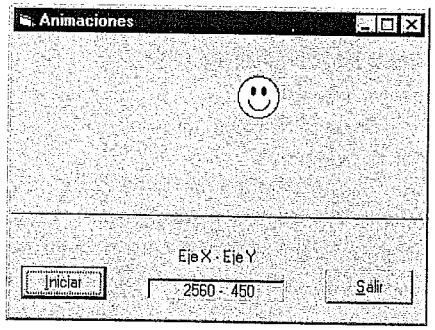
```
Private Sub Salir_Click()
    End    'Finaliza el programa
End Sub
```

12. *Almacene el proyecto y ejecútelo.*

13. *Haga clic en el botón* [Iniciar].



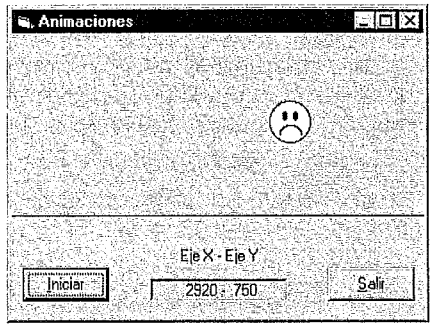
Observe como la imagen comienza a desplazarse hacia arriba y en diagonal, y cuando llega arriba comienza a desplazarse hacia abajo. También verá que aparecen en todo momento las coordenadas (X - Y) en las que se encuentra la imagen.



Fíjese como en el momento en que la figura llega al extremo inferior derecho cambia de aspecto y vuelve a realizar el mismo recorrido que hizo en un principio, pero en sentido contrario.

Este recorrido lo estará realizando indefinidamente hasta que se pulse el botón **Salir**.

14. *Efectúe un clic sobre el botón* [Salir].



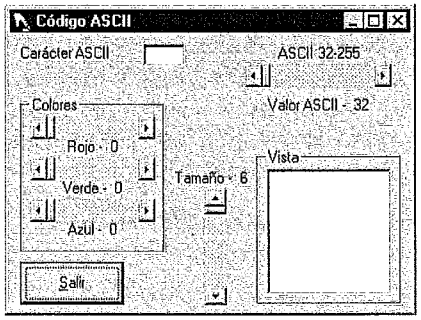
Con esta aplicación daremos por finalizado el apartado dedicado al control **PictureBox**.

Asignar un icono a una aplicación

Asignar un icono a una aplicación es muy sencillo, para ello utilizaremos una de las aplicaciones que realizamos con anterioridad a la cual le asignaremos un icono en movimiento.

Para poder hacer esto necesitaremos usar la propiedad **Icon** que funciona prácticamente como la propiedad **Picture** del control **Image**.

La propiedad ***Icon*** *devuelve el icono que aparece al minimizar un formulario en tiempo de ejecución.*



1. *Abra el proyecto* ***Código ASCII****.*

2. *Active en la ventana* ***Propiedades*** *el* ***Form Principal****.*

3. *Haga clic en la propiedad* ***Icon*** *y seleccione de la carpeta* ***Computer*** *el icono* ***Key04.ico****.*

Observe como en la barra de título aparece el icono que ha seleccionado. Si minimiza la aplicación, el aspecto que presentará en la barra de tareas será [Código ASCII].

4. *Almacene de nuevo la aplicación y ciérrela.*

Hasta aquí hemos visto cómo asignar un icono sencillo, sin embargo lo que haremos a continuación será asignar un icono en movimiento que irá cambiando automáticamente, al mismo tiempo que trabaja con la aplicación.

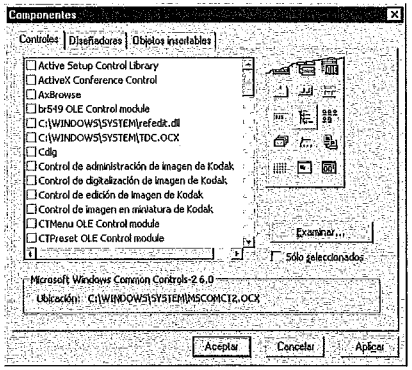
Para ello será necesario trabajar con un nuevo control, se llama **ImageList**.

*El control **ImageList** contiene una colección de objetos **ListImage**, a cada uno de los cuales se puede hacer referencia mediante su índice o su clave.*

1. *Recupere el proyecto **Biblioteca**.*

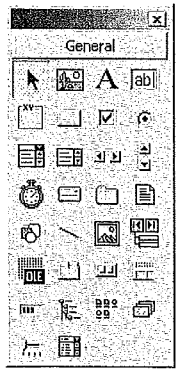
Realice los pasos que se detallan a continuación para insertar el control **ImageList**.

2. *Abra el menú **Proyecto**.*
3. *Seleccione la opción **Componentes**.*



Este cuadro de diálogo permite instalar controles que no están en la barra de herramientas.

Cada una de las opciones puede representar uno o varios controles, como sucede con el control que va a instalar.



A medida que seleccione cada una de la opciones, la parte inferior le informa del nombre del archivo que contiene los controles.

4. *Seleccione y active la casilla **Microsoft Windows Common Controls 6.0**.*
5. *Pulse el botón* Aceptar.

Ahora en la barra de herramientas aparecerán nuevos controles, es posible que no los vea en su totalidad debido al tamaño de la barra, si es este su caso, modifique el tamaño de la misma.

El aspecto que presenta el control **ImageList** es el siguiente .

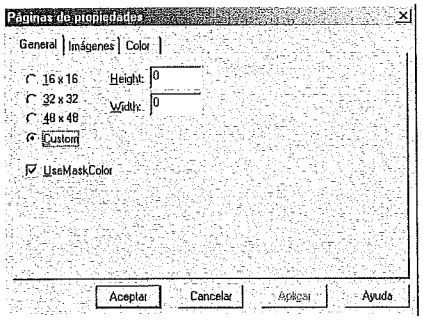
*El control **ImageList** no está concebido para emplearlo en solitario, sino como punto de almacenamiento central para proporcionar cómodamente imágenes a otros controles.*

1. *Inserte un control **ImageList** en un lugar vacío del formulario.*
2. *Efectúe un clic en la propiedad **(Nombre)** de este control y teclee* **Imágenes**.
3. *Efectúe un clic con el botón derecho sobre el control **ImageList**.*
4. *Seleccione la opción **Propiedades**.*

En el cuadro de diálogo que aparece se pueden cargar imágenes, cambiar su tamaño y el color de fondo.

Este control es tratado como si fuera una matriz (*array*).

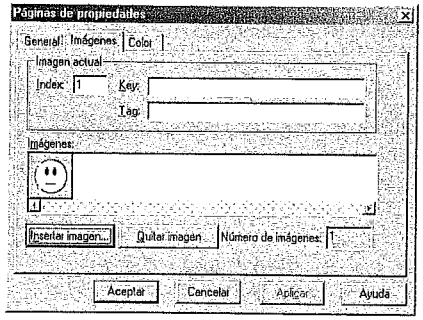
5. *Pulse obre la carpeta **Imágenes**.*
6. *Haga clic en* Insertar imagen...
7. *Vaya al directorio **C:\Archivos de programa\Microsoft Visual Studio\Common\Graphics\Icons\Misc**.*



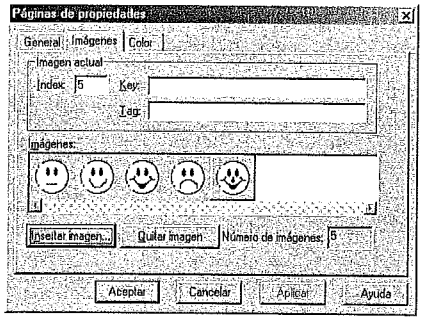
8. *Seleccione el icono **Face01**.*
9. *Haga clic en el botón* Abrir.

Observe que la imagen seleccionada se ha insertado en el cuadro **Imágenes**.

10. *Haga clic en* Insertar imagen...



11. *Seleccione el icono **Face02**.*
12. *Efectúe un clic en* [Abrir].
13. *Haga clic en* [Insertar imagen...].
14. *Seleccione el icono **Face03**.*
15. *Haga clic en el botón* [Abrir].
16. *Haga clic en* [Insertar imagen...].
17. *Seleccione el icono **Face04**.*
18. *Haga clic en el botón* [Abrir].
19. *Haga clic en* [Insertar imagen...].
20. *Seleccione el icono **Face05**.*
21. *Haga clic en el botón* [Abrir].
22. *Pulse el botón* [Aceptar].



A continuación deberá insertar un control **Timer** que se encargará de ejecutar un proceso cada *x* segundos.

1. *Inserte un control **Timer** en una zona vacía del formulario.*
2. *Establezca las propiedades que a continuación se detallan:*

Objeto	Propiedad	Valor
Timer1	(Nombre)	Tiempo
	Interval	300

Antes de teclear el código del control **Timer** deberá definir una variable general que se encargará de contar el número de imágenes utilizadas para realizar la animación. Para ello:

1. *Haga clic sobre el control* [ ].
2. *Seleccione de la lista de objetos **(General)**.*
3. *Teclee el siguiente código:*

```
'Declaración de variables globales
Dim x As Integer
```

4. *Seleccione de la lista de objetos el objeto **Form**.*
5. *Teclee el siguiente código:*

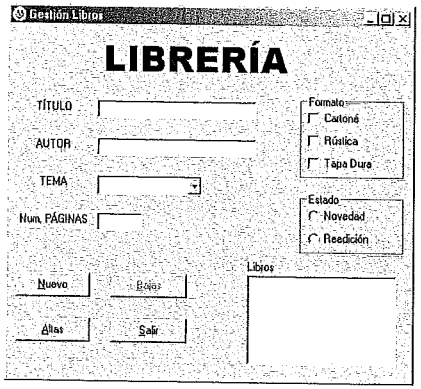
```
Private Sub Form_Load()
    x = 0
    Reg = 0
    TemaL.AddItem "Acción"
    TemaL.AddItem "Aventuras"
    TemaL.AddItem "Biografía"
    TemaL.AddItem "Ciencia"
    TemaL.AddItem "Ciencia Ficción"
    TemaL.AddItem "Cine"
    TemaL.AddItem "Economía"
    TemaL.AddItem "Gastronomía"
    TemaL.AddItem "Historia"
    TemaL.AddItem "Informática"
    TemaL.AddItem "Medicina"
    TemaL.AddItem "Misterio"
    TemaL.AddItem "Naturaleza"
    TemaL.AddItem "Policíaco"
    TemaL.AddItem "Política"
    TemaL.AddItem "Romántica"
    TemaL.AddItem "Teatro"
    TemaL.AddItem "Terror"

    'Desactivación del botón Bajas
    Bajas.Enabled = False
End Sub
```

En este procedimiento se inicializa la variable **x** con el valor **0**.

6. *Seleccione de la lista de objetos el objeto **Tiempo**.*

7. *Teclee el siguiente código:*

```
'Controlador de las imágenes que van apareciendo
'en la barra de título de la aplicación
Private Sub Tiempo_Timer()
        x = x + 1              'Número de imagen a cargar
        Biblioteca.Icon = Imagenes.ListImages(x).Picture
        If x = 5 Then x = 0    'Al llegar a la última imagen comienza de nuevo
End Sub
```



El código que acaba de teclear se ejecutará cada 300 milisegundos (interval 300). Su función es la de ir cambiando el icono de la aplicación usando la propiedad **Icon**. La imagen será una de las que tiene cargadas el control **ImageList** usando la variable **x** como índice.

Esta variable se aumenta cada vez en una unidad y en el momento que alcanza el valor **5** (número de imágenes) se vuelve a inicializar con el valor **0** para que empiece de nuevo.

8. *Almacene la aplicación y ejecútela.*

Observe como el icono de la barra de título va cambiando tanto si está ejecutando la aplicación como si la minimiza.

Creación de imágenes compuestas

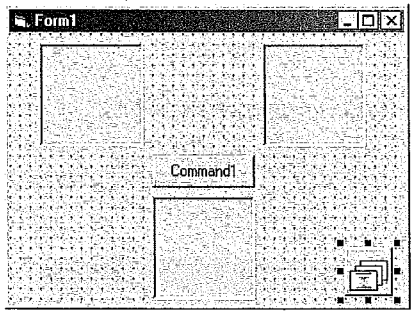
El control **ImageList** también se puede utilizar para crear imágenes compuestas, es decir, superponer dos imágenes. Para conseguir esto, deberemos utilizar el método **Overlay** junto con la propiedad **MaskColor**. Veamos un ejemplo:

Imagine que desea superponer las siguientes imágenes:

1. *Abra un **Nuevo proyecto** del tipo **EXE estándar**.*

2. *Inserte los siguientes controles en la posición y con el tamaño que se indica a continuación:*

Objeto	Posición	Tamaño
Picture1	360, 120	1215 x 1215
Picture2	3000, 120	1215 x 1215
Picture3	1680, 1920	1215 x 1215
Command1	1680, 1440	1215 x 375



Para insertar el control **ImageList** realice los mismos pasos que hizo en el apartado anterior (**Asignar un icono a una aplicación**).

Una vez insertado este control, el aspecto de su pantalla debe ser similar al de la figura de la izquierda. A continuación, establezca las siguientes propiedades:

a) Propiedades del formulario (**Form**).

Objeto	Propiedad	Valor
Form1	(Nombre)	Imagen
	Caption	Imágenes Superpuestas

b) Establecer las propiedades de los botones (**CommandButton**).

Objeto	Propiedad	Valor
Command1	(Nombre)	Union
	Caption	&Unir Imágenes

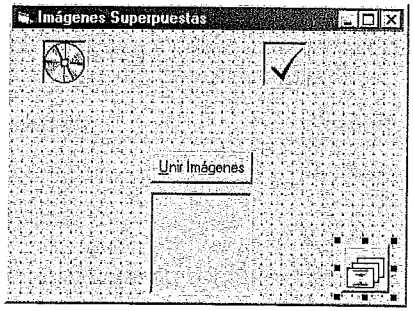
c) Establecer las propiedades de los cuadros de imagen (**PictureBox**).

Objeto	Propiedad	Valor
Picture1	(Nombre)	Imagen1
	AutoSize	True
	Picture	Carpeta Computer
		Cdrom01
Picture2	(Nombre)	Imagen2
	AutoSize	True
	Picture	Carpeta Misc
		Checkmrk
Picture3	(Nombre)	Imagen3
	AutoSize	True

d) Establecer las propiedades de la lista de imagen (**ImageList**).

Objeto	Propiedad	Valor
ImageList	(Nombre)	Imágenes

El aspecto actual de su formulario aparece en la figura de la derecha.



1. *Efectúe un clic con el botón derecho sobre el control* ***ImageList***.
2. *Seleccione la opción* ***Propiedades***.
3. *Pulse obre la carpeta* ***Imágenes***.
4. *Haga clic en* [Insertar imagen...].
5. *Vaya al directorio* ***C:\Archivos de programa\Microsoft Visual Studio\Common\Graphics\Icons\Computer***.
6. *Seleccione el icono* ***Cdrom01***.
7. *Haga clic en el botón* [Abrir].

8. *Haga clic en* [Insertar imagen...].
9. *Vaya al directorio* ***C:\Archivos de programa\Microsoft Visual Studio\Common\Graphics\Icons\Misc***.
10. *Seleccione el icono* ***Checkmrk***.
11. *Haga clic en el botón* [Abrir].
12. *Pulse el botón* [Aceptar].
13. *Efectúe un doble clic sobre el botón* ***Unir Imágenes*** *y teclee el siguiente código:*

```
Private Sub Union_Click()
        Set Imagen3.Picture = Imágenes.Overlay(1, 2)
End Sub
```

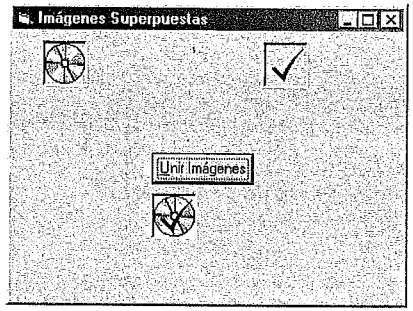
El método ***Overlay*** *permite superponer dos imágenes. Su sintaxis es* ***objeto.Overlay(índice1, índice2).***

objeto *es el objeto sobre el cual se realiza la superposición.*

índice1 *especifica la imagen que va a ser superpuesta.*

índice2 *especifica la imagen que se va a superponer.*

El color de la imagen que coincide con el valor de la propiedad ***MaskColor*** *aparecerá transparente. Si no hay ningún color coincidente, la imagen se dibujará opaca sobre la otra imagen.*



14. *Almacene la aplicación con el nombre* ***Fusión*** *tanto para el formulario como para el proyecto.*
15. *Ejecute la aplicación y pulse sobre el botón* ***Unir Imágenes***.

En el supuesto de que hubiese seleccionado otro tipo de imágenes, y al superponerlas no obtuviese el resultado esperado, deberá utilizar la propiedad **MaskColor**.

*La propiedad **MaskColor** establece el color que se usa para crear una máscara de un control **ImageList**. Su sintaxis es **objeto.MaskColor [= color]**.*

***objeto** es el control sobre el cual se aplica la máscara.*

***color** es un valor o constante que determina el color utilizado para crear máscaras.*

*Esta orden debe codificarse antes del método **Overlay**.*

Asignar un fondo al formulario

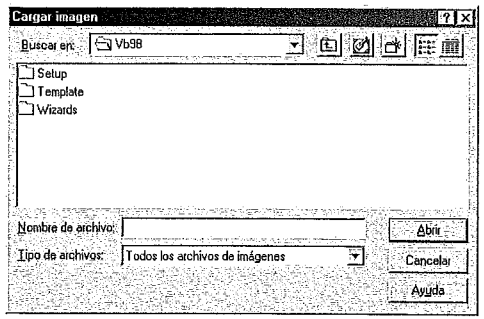
Al igual que algunos de los controles de Visual Basic, el formulario también puede tener en su interior un gráfico.

Asignar un gráfico al fondo de un formulario, puede provocar que la aplicación se enlentezca a la hora de presentar el formulario en pantalla una vez se ejecute la aplicación.

El ejemplo que realizaremos consistirá en asignar un fondo a una aplicación que ya tenemos hecha con anterioridad.

1. *Recupere el proyecto que lleva por nombre **Calculadora**.*
2. *Active el formulario **Calculadora**.*
3. *Sitúese en la ventana **Propiedades** y seleccione la propiedad **Picture**.*
4. *Pulse sobre el botón ...*

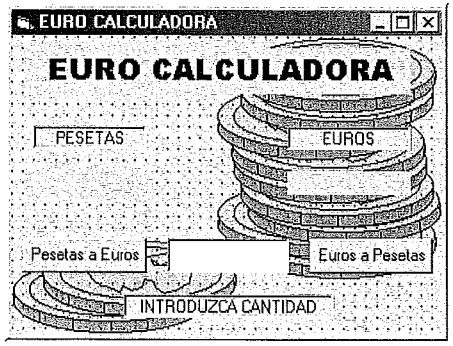
Aparecerá el siguiente cuadro de diálogo:



5. *Suba un nivel pulsando sobre el botón.*
6. *Haga doble clic sobre la carpeta **Common**.*

7. *Vuelva a efectuar un doble clic sobre la carpeta **Graphics**.*
8. *Abra la carpeta **Metafile** y después la carpeta **Business**.*
9. *Seleccione el archivo **Coins** y pulse el botón Abrir.*

*En la carpeta **Graphics** encontrará todas las imágenes, cursores e iconos que incorpora Visual Basic.*



El aspecto del formulario será como el de la figura de la izquierda.

Para finalizar haremos una pequeña modificación en una de las propiedades del formulario. Esta consistirá en desactivar el botón de maximizar, ya que si lo pulsase, el formulario y el gráfico ocuparían toda la pantalla, mientras que las etiquetas y botones de la calculadora quedarían desplazados con respecto al tamaño del formulario y esto a veces puede resultar poco atractivo.

1. *Seleccione la propiedad **MaxButton** de la ventana **Propiedades**.*
2. *Asígnele el valor **False**.*

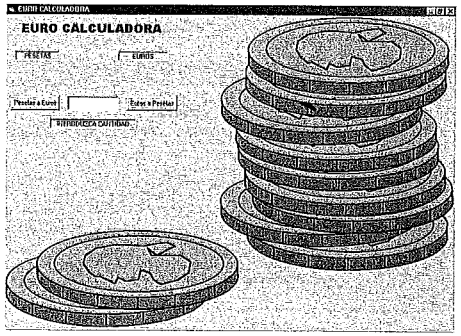
*La propiedad **MaxButton** devuelve un valor que indica si un formulario tiene un botón de maximizar. Si el valor es **True** incorpora dicho botón, en cambio si es **False** aunque lo incorpora permanece desactivado.*

*El botón **Maximizar** se convierte automáticamente en un botón **Restaurar** cuando la ventana está maximizada. Al minimizar o restaurar la ventana, el botón **Restaurar** pasará automáticamente de nuevo a ser un botón **Maximizar**.*

Si por el contrario ha optado por dejar el botón de maximizar activado, cuando ejecute la aplicación y pulse sobre el botón maximizar, el formulario presentará un aspecto como el de la figura de la siguiente página:

3. *Almacene de nuevo la aplicación.*

También puede asignar un fondo de color al formulario, usando la propiedad **BackColor** del formulario.



BackColor *devuelve o establece el color de fondo de un objeto. Su sintaxis es* **objeto.*BackColor* [= color]**.

Si establece la propiedad ***BackColor*** *de un objeto* ***Form*** *o de un control* ***PictureBox***, *se borra todo el texto y todos los gráficos, incluidos los gráficos persistentes.*

9. Arrays de controles

En este capítulo verá lo que es un array de controles y para qué sirve, así como su funcionamiento por medio de ejemplos.

Un ***array de controles*** *es un grupo de controles que comparten un nombre, tipo y procedimientos de evento comunes. Cada control de la matriz tiene un número de índice único que se puede usar para determinar qué control reconoce un evento.*

Imagínese una aplicación en la que simulemos una calculadora, pues bien será necesario tener diez botones numerados del 0 al 9. Si tuviera que escribir el código para cada uno de los botones, éste sería demasiado largo y repetitivo, sin embargo mediante el uso de los *arrays de controles*, sólo deberá teclear el código una vez e identificar cada botón con un índice.

Diseño de un array de controles

Hay tres maneras de crear una matriz de controles en tiempo de diseño:

1. Asignar el mismo nombre a más de un control.
2. Copiar un control existente y después pegarlo en el formulario.
3. Establecer la propiedad **Index** del control a un valor distinto de **Null**.

La aplicación que vamos a realizar consiste en un calendario perpetuo de forma que el usuario introducirá un mes y un año determinados y el ordenador mostrará en pantalla el calendario de ese mes.

1. *Ejecute el programa de* ***Visual Basic 6.0*** *si no lo estaba ejecutando.*
2. *Abra un* ***Nuevo proyecto*** *del tipo* ***EXE estándar***.
3. *Seleccione la propiedad* ***Height*** *y teclee el valor* ***7000***.
4. *Seleccione la propiedad* ***Width*** *y teclee el valor* ***6840***.

5. *Modifique el tamaño de la ventana* ***Proyecto1*** *para que sea visible todo el formulario.*

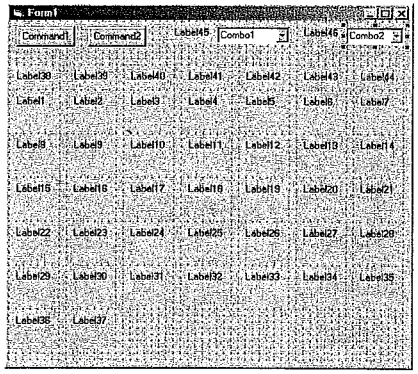
6. *Inserte los siguientes controles en la posición y con el tamaño que se indica a continuación:*

Objeto	Posición	Tamaño
Label1	124, 1440	637 x 615
Label2	1120, 1440	637 x 615
Label3	2115, 1440	637 x 615
Label4	3111, 1440	637 x 615
Label5	4106, 1440	637 x 615
Label6	5102, 1440	637 x 615
Label7	6097, 1440	637 x 615
Label8	124, 2280	637 x 615
Label9	1120, 2280	637 x 615
Label10	2115, 2280	637 x 615
Label11	3111, 2280	637 x 615
Label12	4106, 2280	637 x 615
Label13	5102, 2280	637 x 615
Label14	6097, 2280	637 x 615
Label15	124, 3120	637 x 615
Label16	1120, 3120	637 x 615
Label17	2115, 3120	637 x 615
Label18	3111, 3120	637 x 615
Label19	4106, 3120	637 x 615
Label20	5102, 3120	637 x 615
Label21	6097, 3120	637 x 615
Label22	124, 3960	637 x 615

Objeto	Posición	Tamaño
Label23	1120, 3960	637 x 615
Label24	2115, 3960	637 x 615
Label25	3111, 3960	637 x 615
Label26	4106, 3960	637 x 615
Label27	5102, 3960	637 x 615
Label28	6097, 3960	637 x 615
Label29	124, 4800	637 x 615
Label30	1120, 4800	637 x 615
Label31	2115, 4800	637 x 615
Label32	3111, 4800	637 x 615
Label33	4106, 4800	637 x 615
Label34	5102, 4800	637 x 615
Label35	6097, 4800	637 x 615
Label36	124, 5640	637 x 615
Label37	1120, 5640	637 x 615
Label38	124, 960	637 x 255
Label39	1120, 960	637 x 255
Label40	2115, 960	637 x 255
Label41	3111, 960	637 x 255
Label42	4106, 960	637 x 255
Label43	5102, 960	637 x 255
Label44	6097, 960	637 x 255
Label45	2862, 120	637 x 255
Label46	5102, 120	637 x 255
Command1	124, 120	1011 x 375
Command2	1368, 120	1011 x 375

Objeto	Posición	Tamaño
Combo1	3608, 120	1260 x 315
Combo2	5848, 120	1011 x 315

Si ha situado correctamente los controles, su formulario deberá presentar un aspecto similar al de la figura de la derecha.



1. *Pulse el botón* [icono] *de la barra de herramientas.*
2. *Teclee* ***Calendario*** *como nombre de formulario y pulse el botón* [Guardar].
3. *Teclee* ***Calendario*** *como nombre de proyecto y pulse el botón* [Guardar].

A continuación establezca las siguientes propiedades en el formulario y sus controles correspondientes:

a) Propiedades del formulario (**Form**).

Objeto	Propiedad	Valor
Form1	(Nombre)	Calendario
	Caption	Calendario

b) Establecer las propiedades de los botones (**CommandButton**).

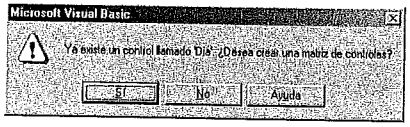
Objeto	Propiedad	Valor
Command1	(Nombre)	Calcular
	Caption	&Calcular
Command2	(Nombre)	Salir
	Caption	&Salir

c) Establecer las propiedades de los cuadros combinados (**ComboBox**).

Objeto	Propiedad	Valor
Combo1	(Nombre)	Meses
	Text	(Vacío)
Combo2	(Nombre)	Anys
	Text	(Vacío)

Hemos dejado para el final las etiquetas porque es aquí donde crearemos el *array de controles.*

1. *Seleccione el control* ***Label1*** *y teclee* ***Dia*** *en la propiedad* ***(Nombre)****.*
2. *Seleccione el control* ***Label1*** *y borre el texto que hay en la propiedad* ***Caption****.*
3. *Seleccione el control* ***Label2*** *y teclee* ***Dia*** *en la propiedad* ***(Nombre)****.*



Observe como aparece una ventana preguntando si desea crear una matriz de controles.

4. *Responda afirmativamente.*

Con esta sencilla operación ya tiene creada la matriz de controles.

Otra forma de crear arrays es creando el primer control, especificar las características y a continuación, utilizar los comandos ***Copiar*** *y* ***Pegar****.*

5. *Seleccione el control* ***Label2*** *y borre el texto que hay en la propiedad* ***Caption****.*
6. *Repita los pasos* ***1*** *y* ***2*** *para las casillas que van de* ***Label3*** *a* ***Label37****.*

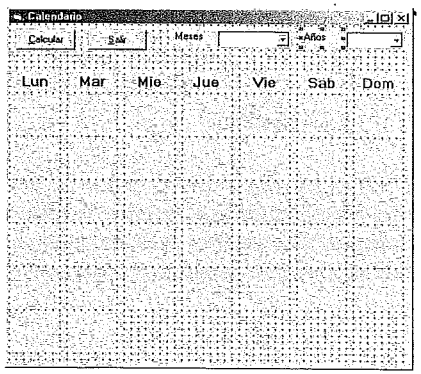
Continue estableciendo las siguientes propiedades para el resto de las etiquetas.

d) Establecer las propiedades de las etiquetas (**Label**).

Objeto	Propiedad	Valor
Label38	(Nombre)	Lunes
	Alignment	2 - Center
	Caption	Lun
	Font	Negrita - 12
Label39	(Nombre)	Martes
	Alignment	2 - Center
	Caption	Mar
	Font	Negrita - 12
Label40	(Nombre)	Miercoles
	Alignment	2 - Center
	Caption	Mie
	Font	Negrita - 12
Label41	(Nombre)	Jueves
	Alignment	2 - Center
	Caption	Jue
	Font	Negrita - 12
Label42	(Nombre)	Viernes
	Alignment	2 - Center
	Caption	Vie
	Font	Negrita - 12
Label43	(Nombre)	Sabado
	Alignment	2 - Center
	Caption	Sab
	Font	Negrita - 12
	ForeColor	&H00FF0000&
Label44	(Nombre)	Domingo

Objeto	Propiedad	Valor
	Alignment	2 - Center
	Caption	Dom
	Font	Negrita - 12
	ForeColor	&H000000FF&
Label45	(Nombre)	Mes
	Caption	Meses
Label46	(Nombre)	Any
	Caption	Años

En estos momentos su formulario debe tener un aspecto similar al de la figura de la derécha:



*Utilice siempre **arrays** cuando el formulario tenga controles que realizan la misma operación.*

Una vez realizado el diseño, comenzaremos a escribir el código de la aplicación.

1. *Haga clic sobre el control* .
2. *Seleccione de la lista desplegable de objetos el objeto **Form**.*
3. *Teclee el siguiente código:*

```
Private Sub Form_Load()
        Dim x As Integer    'Contador de años

        'Inicialización del cuadro combinado de los meses
        Meses.AddItem "Enero"
        Meses.AddItem "Febrero"
        Meses.AddItem "Marzo"
```

INSTITUTO DE EDUCACION SECUNDARIA "ARROYO HARINA" ALMENDRALEJO - (Badajoz)

```
        Meses.AddItem "Abril"
        Meses.AddItem "Mayo"
        Meses.AddItem "Junio"
        Meses.AddItem "Julio"
        Meses.AddItem "Agosto"
        Meses.AddItem "Septiembre"
        Meses.AddItem "Octubre"
        Meses.AddItem "Noviembre"
        Meses.AddItem "Diciembre"

        'Inicialización del cuadro combinado de los años
        For x = 1900 To 2100
            Anys.AddItem Str(x)
        Next

        'Desactiva el botón Calcular
        Calcular.Enabled = False
    End Sub
```

En el procedimiento anterior se inicializan los valores de los dos cuadros combinados (**Meses** y **Anys**), al mismo tiempo que se desactiva el botón **Calcular** para evitar que se produzca un error al intentar ver el mes de una fecha incorrecta (mes y/o año en blanco).

Tenga en cuenta que siempre que defina un array sin establecer el límite inferior, éste empezará por cero.

4. *Seleccione de la lista desplegable de objetos el objeto **Meses**.*
5. *Seleccione de la lista desplegable de procedimientos el procedimiento **Click**.*
6. *Teclee el siguiente código:*

```
'Comprobación de que se ha seleccionado un mes y un año
Private Sub Meses_Click()
    If Meses.Text <> "" And Anys.Text <> "" Then
        Calcular.Enabled = True     'Activa el botón Calcular
    End If
End Sub
```


7. *Seleccione de la lista desplegable de objetos el objeto **Anys**.*
8. *Seleccione de la lista desplegable de procedimientos el procedimiento **Click**.*
9. *Teclee el siguiente código:*

```
'Comprobación de que se ha seleccionado un mes y un año
Private Sub Anys_Click()
    If Meses.Text <> "" And Anys.Text <> "" Then
        Calcular.Enabled = True     'Activa el botón Calcular
    End If
End Sub
```

Con los dos procedimientos anteriores lo que se hace es comprobar que haya algún valor válido en ambos cuadros de diálogo, y si es así activa el botón **Calcular**.

10. *Seleccione de la lista desplegable de objetos el objeto **Calcular**.*
11. *Teclee el siguiente código:*

```
'Calcula el mes y el año seleccionado
Private Sub Calcular_Click()
    Dim NumDias As Integer, NumMes As Integer, _
    DiaSemana As Integer
    Dim  DiasMes As Integer, Pos As Integer, d As Integer
    Dim Fecha As Date

    'Borra el calendario anterior e inicializa sus valores
    For d = 0 To 36
        Dia(d).Appearance = 1
        Dia(d).Caption = ""
        Dia(d).Font = "Arial"
        Dia(d).FontBold = True
        Dia(d).FontSize = 26
        Dia(d).Alignment = 2
    Next

    DiasMes = 0

    'Calcula el número del mes y los días de cada mes
```

```
Select Case Meses.Text
    Case "Enero"
        NumDias = 31
        NumMes = 1
    Case "Febrero"
        If Val(Anys.Text) Mod 4 = 0 Then
            NumDias = 29
        Else
            NumDias = 28
        End If
        NumMes = 2
    Case "Marzo"
        NumDias = 31
        NumMes = 3
    Case "Abril"
        NumDias = 30
        NumMes = 4
    Case "Mayo"
        NumDias = 31
        NumMes = 5
    Case "Junio"
        NumDias = 30
        NumMes = 6
    Case "Julio"
        NumDias = 31
        NumMes = 7
    Case "Agosto"
        NumDias = 31
        NumMes = 8
    Case "Septiembre"
        NumDias = 30
        NumMes = 9
    Case "Octubre"
        NumDias = 31
        NumMes = 10
    Case "Noviembre"
        NumDias = 30
        NumMes = 11
    Case "Diciembre"
```

Año bisiesto


```
        NumDias = 31
        NumMes = 12
    End Select

    'Conversión de una cadena de caracteres en una fecha
    Fecha = CDate("1/" + Str(NumMes) + "/" + Anys.Text)

    'Cálculo del día de la semana del día 1 de cada mes
    DiaSemana = Weekday(Fecha, 2)

    Pos = DiaSemana     'Posición de los sábados y los domingos

    For d = DiaSemana To (NumDias + DiaSemana)
        If DiasMes < NumDias Then
            Dia(d - 1).Appearance = 0
            DiasMes = DiasMes + 1

            If Pos = 6 Then     'Si es Sábado
                Dia(d - 1).ForeColor = RGB(0, 0, 255)
            ElseIf Pos = 7 Then 'Si es Domingo
                Dia(d - 1).ForeColor = RGB(255, 0, 0)
                Pos = 0
            Else            'Si no es festivo
                Dia(d - 1).ForeColor = RGB(0, 0, 0)
            End If

            Dia(d - 1).Caption = DiasMes
            Pos = Pos + 1
        End If
    Next
End Sub
```

Pedir ayuda a [illegible]

Ha de tener cuidado cuando trabaje con matrices, ya que si Ud. hace referencia a un elemento de la matriz que no existe, éste provocará el error **'matriz fuera de rango'**.

En el procedimiento **Calcular** se realizan las siguientes tareas:

1. Se borra el calendario del mes anterior y se especifica el tipo de letra que se va a utilizar, así como su tamaño y alineación.
2. Dependiendo del mes seleccionado se calcula el número del mes (1-Enero, 2-Febrero...) y el número de días de cada uno de los meses (31-Enero, 28-Febrero...).
3. Mediante la unión de los valores introducidos en los cuadros combinados y con ayuda de la función **CDate** se convierten estos valores en una fecha.

*La función **CDate** devuelve una expresión que se convirtió en **Variant** del subtipo **Date**. Su sintaxis es **CDate(fecha)**. El argumento fecha es cualquier expresión de fecha válida.*

***CDate** reconoce literales de fecha y literales de hora, así como algunos números que se encuentran dentro del rango de fechas aceptable.*

4. Una vez obtenida la fecha se calcula qué día de la semana corresponde al día 1 (Enero, Febrero...) del mes y del año seleccionado con ayuda de la función **Weekday**.

*La función **Weekday** devuelve un valor de tipo **Variant (Integer)** que contiene un número entero que representa el día de la semana. Su sintaxis es **Weekday(fecha, [primerdíasemana])**.*

*La sintaxis de la función **Weekday** consta de los siguientes argumentos con nombre:*

***fecha** es una expresión numérica,expresión de tipo cadena o cualquier combinación que puede representar una fecha. Si fecha contiene Null, la función devolverá Null.*

***primerdíasemana** es una constante que especifica el primer día de la semana. Este argumento tiene estos valores: **1** - Domingo (predeterminado), **2** - Lunes, **3** - Martes, **4** - Miércoles, **5** - Jueves, **6** - Viernes y **7** - Sábado.*

*Los valores devueltos son: **1** - Domingo, **2** - Lunes, **3** - Martes, **4** - Miércoles, **5** - Jueves, **6** - Viernes y **7** - Sábado.*

5. Una vez calculado el día de la semana por el que comienza el mes, se visualizan todos sus días en diferentes colores (Negro - Laborables, Azul - Sábados, Rojo - Domingos) dependiendo del día de la semana.

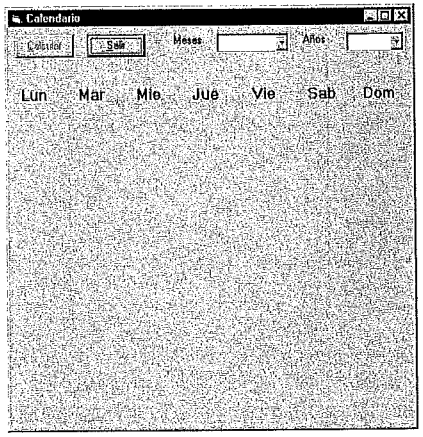
*Para trabajar con las propiedades de un array desde el código, debemos escribir el control seguido del índice entre paréntesis, un punto y el nombre de la propiedad. **NombreControl(Índice).Propiedad**.*

*En el trozo de código donde se escriben los días de cada mes, se utiliza como índice **d-1** debido a que el array comienza por **0** y nosotros asumimos que el Lunes se corresponde con el número **1**, cuando en realidad en la tabla es **0**.*

12. *Seleccione de la lista desplegable de objetos el objeto **Salir**.*
13. *Teclee el siguiente código:*

```
Private Sub Salir_Click()
        End    'Finaliza el programa
End Sub
```

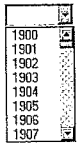
14. *Almacene y ejecute la aplicación.*



Observe como al ejecutar el programa aparece la hoja del calendario con los cuadros combinados en blanco y con el botón **Calcular** desactivado.

1. *Efectúe un clic en la lista desplegable del cuadro combinado correspondiente a los meses.*

2. *Seleccione el mes de **Enero**.*



3. *Realice otro clic sobre el cuadro combinado correspondiente a los años.*
4. *Seleccione el año **1999**.*

Automáticamente se ha activado el botón **Calcular**.

5. *Haga clic sobre el botón* Calcular.

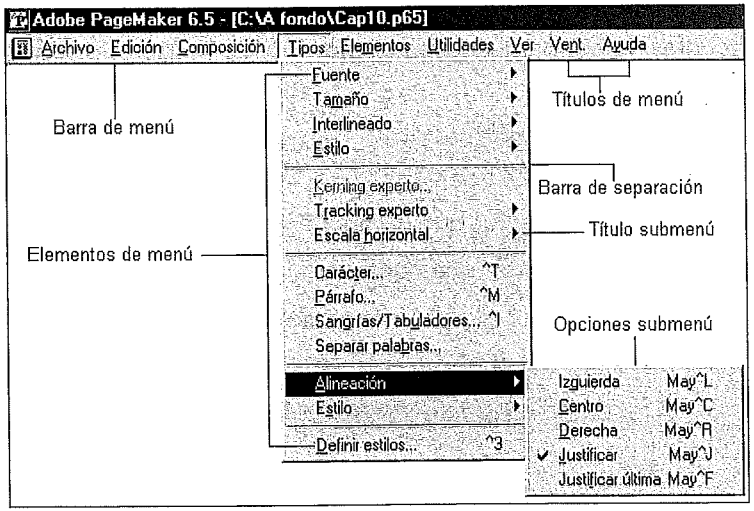
Inmediatamente aparecerá en el calendario el mes correspondiente al año que haya seleccionado.

Si quiere ver el calendario de otra fecha, bastará con cambiar el mes y/o el año y volver a hacer clic en el botón **Calcular**.

10. Trabajando con menús y cuadros de diálogo

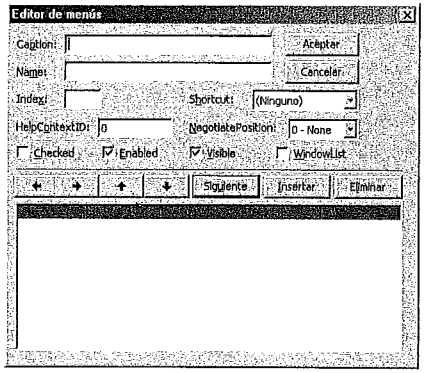
Hasta ahora las aplicaciones que hemos realizado tenían un formulario y varios controles con un aspecto bastante atractivo, sin embargo el aspecto de éstas se puede mejorar utilizando menús. En este capítulo aprenderá a crear y a utilizar menús en una aplicación

*Un **menú** es una lista de elementos, como comandos y nombres de archivos, que aparecen al elegir un título de menú en la barra de menús y que están agrupadas bajo un mismo título.*



Si desea que la aplicación proporcione un conjunto de comandos a los usuarios esta es la mejor manera de hacerlo. Un menú se puede desplegar tanto por medio del ratón como por medio del teclado. Cuando el usuario efectúe un clic sobre el título de un menú se desplegará una lista con todos los elementos que contiene el menú. En caso de que no contenga elementos, se ejecutará directamente una acción determinada.

Diseño de un menú



Para diseñar un menú se utilizará el editor de menús que presenta un aspecto como el de la figura de la derecha.

El acceso a este menú se puede realizar de dos modos distintos:

1. Haciendo clic en el botón .
2. Pulsando la combinación de teclas **Ctrl + E**.

La aplicación que realizaremos para practicar con el diseño de un menú consistirá en un simple editor de textos.

Pero antes de nada enunciaremos brevemente los pasos a seguir para crear un menú:

a) Abrir el editor de menús.

b) Introducir el título del menú.

c) Introducir los elementos que componen el menú.

d) Crear un submenú (si fuese necesario).

e) Añadir un separador (si fuese necesario).

f) Cerrar el editor de menús.

Una vez vistos los pasos necesarios para crear un menú comenzaremos a crearlo.

1. *Ejecute el programa de **Visual Basic 6.0** si no lo estaba ejecutando.*
2. *Abra un Nuevo proyecto del tipo **EXE estándar**.*
3. *Efectúe un clic sobre el botón .*

Observe como aparece el cuadro de diálogo mostrado en la parte superior. Mediante este cuadro podrá crear entre otras cosas las opciones y subopciones del menú de la aplicación. Dispone de una serie de botones

que le permitirán insertar y eliminar opciones, así como asignar una combinación de teclas a una opción del menú.

4. *Teclee en el cuadro **Caption** el texto **&Archivo**.*

*El cuadro **Caption** permite escribir el nombre del comando o del menú que desea que aparezca en la barra de menús o en un menú.*

*Si desea crear una barra separadora en el menú, escriba un único guión (-) en el cuadro **Caption**.*

5. *Teclee en el cuadro **Name** el texto **MenuArchivo**.*

*El cuadro **Name** permite escribir un nombre de control para el elemento del menú. El nombre de control es un identificador que sólo se utiliza para tener acceso al elemento del menú en el código, no aparece en ningún menú.*

6. *Pulse la tecla **Enter**.*

Con estos pasos acabamos de definir una de las opciones principales del menú, realice los pasos que se describen a continuación para crear las subopciones.

7. *Efectúe un clic sobre el botón para crear una subopción.*

***Flecha a la derecha**. Pasa el menú seleccionado a un nivel inferior cada vez que hace clic en el botón. Puede crear hasta cuatro niveles de submenús.*

***Flecha a la izquierda**. Pasa el menú seleccionado a un nivel superior cada vez que hace clic en el botón. Puede crear hasta cuatro niveles de submenús.*

***Flecha arriba**. Cada vez que se hace clic en este botón, el elemento seleccionado del menú se mueve hacia arriba una posición dentro del mismo nivel de menú.*

***Flecha abajo**. Cada vez que se hace clic en este botón, el elemento seleccionado del menú se mueve hacia abajo una posición dentro del mismo nivel de menú.*

8. *Establezca las siguientes propiedades:*

   Caption: ***&Nuevo***

   Name: ***MenuANuevo***

9. *Pulse la tecla* ***Enter***.

10. *Establezca las siguientes propiedades:*

    Caption: ***&Abrir***

    Name: ***MenuAAbrir***

11. *Pulse la tecla* ***Enter***.

12. *Establezca las siguientes propiedades:*

    Caption: ***&Guardar***

    Name: ***MenuAGuardar***

    Enabled: ***Desactivado***

El cuadro ***Enabled*** *permite seleccionar si el elemento del menú debe responder a eventos. Desactive esta opción si desea que el elemento del menú no esté disponible y aparezca atenuado.*

13. *Pulse la tecla* ***Enter***.

14. *Establezca las siguientes propiedades:*

    Caption: -

    Name: ***MenuASeparador***

15. *Pulse la tecla* ***Enter***.

16. *Establezca las siguientes propiedades:*

    Caption: ***&Salir***

    Name: ***MenuASalir***

    Shortcut: ***Ctrl + S***

El cuadro ***Shortcut*** *permite seleccionar una tecla de método abreviado para cada comando.*

Con esto ya tenemos definidas todas las opciones que compondrán el menú **Archivo**.

A continuación pasaremos a definir las opciones de los menús **Edición**, **Formato** y **Acerca de**.

17. *Pulse la tecla* ***Enter***.

18. *Efectúe un clic sobre el botón* ← *para subir un nivel.*

19. *Establezca las siguientes propiedades:*

    Caption: ***&Edición***

    Name: ***MenuEdicion***

20. *Pulse la tecla* ***Enter***.

21. *Efectúe un clic sobre el botón* → *para crear una subopción.*

22. *Establezca las siguientes propiedades:*

    Caption: ***&Cortar***

    Name: ***MenuECortar***

    Shortcut: ***Ctrl + X***

    Enabled: ***Desactivado***

23. *Pulse la tecla* ***Enter***.

24. *Establezca las siguientes propiedades:*

    Caption: ***C&opiar***

    Name: ***MenuECopiar***

    Shortcut: ***Ctrl + C***

    Enabled: ***Desactivado***

25. *Pulse la tecla* ***Enter***.

26. *Establezca las siguientes propiedades:*

    Caption: ***&Pegar***

    Name: ***MenuEPegar***

    Shortcut: ***Ctrl + V***

    Enabled: ***Desactivado***

27. *Pulse la tecla* ***Enter***.

28. *Efectúe un clic sobre el botón* [←] *para subir un nivel.*

29. *Establezca las siguientes propiedades:*

    Caption: ***&Formato***

    Name: ***MenuFormato***

30. *Pulse la tecla* ***Enter***.

31. *Efectúe un clic sobre el botón* [→] *para crear una subopción.*

32. *Establezca las siguientes propiedades:*

    Caption: ***&Negrita***

    Name: ***MenuFNegrita***

    Shortcut: ***Ctrl + N***

    Enabled: ***Desactivado***

33. *Pulse la tecla* ***Enter***.

34. *Establezca las siguientes propiedades:*

    Caption: ***&Cursiva***

    Name: ***MenuFCursiva***

    Shortcut: ***Ctrl + K***

    Enabled: ***Desactivado***

35. *Pulse la tecla* ***Enter***.

36. *Establezca las siguientes propiedades:*

    Caption: ***&Subrayado***

    Name: ***MenuFSubrayado***

    Shortcut: ***Ctrl + U***

    Enabled: ***Desactivado***

37. *Pulse la tecla* ***Enter***.

38. *Efectúe un clic sobre el botón* [←] *para subir un nivel.*

39. *Establezca las siguientes propiedades:*

    Caption: ***A&cerca de...***

    Name: ***MenuAcerca***

40. *Pulse el botón* [Aceptar].

Antes de ver el aspecto del menú que acaba de crear haremos una breve descripción del resto de las opciones que había en el editor de menús.

El cuadro ***Index*** *permite asignar un valor numérico que determina la posición del control en una matriz de controles.*

El cuadro ***HelpContextID*** *permite asignar un valor numérico único para el Id. de contexto. Este valor se utiliza para encontrar el tema de Ayuda apropiado en el archivo de Ayuda identificado mediante la propiedad* ***HelpFile***.

El cuadro ***NegotiatePosition*** *permite seleccionar la propiedad* ***NegotiatePosition*** *del menú. Esta propiedad determina si el menú aparece o no en un formulario contenedor y cómo aparece.*

El cuadro ***Checked*** *permite hacer que aparezca inicialmente una marca de verificación a la izquierda de un elemento del menú.*

El cuadro ***Visible*** *permite hacer que un elemento aparezca en el menú.*

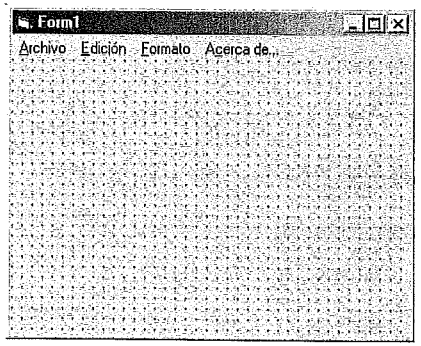
El cuadro ***WindowList*** *determina si el control del menú contiene una lista de formularios secundarios MDI abiertos en una aplicación MDI.*

La ***Lista Menús*** *es un cuadro de lista que muestra en orden jerárquico todos los elementos del menú. Los elementos de los submenús aparecen indentados para indicar su ubicación o su nivel en la jerarquía.*

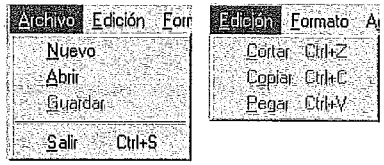
El botón [Siguiente] *selecciona la línea siguiente.*

El botón Insertar *inserta una línea en el cuadro de lista, inmediatamente encima de la línea actualmente seleccionada.*

El botón Eliminar *borra la línea actualmente seleccionada.*



Observe como se ha creado la barra de menús con las opciones que le ha especificado.



En el menú **Acerca de...** no se despliega ninguna ventana porque no se ha creado ninguna opción dentro de ella.

Creación de un Editor de textos

Nuestra aplicación va a tratar de la creación de un editor de textos, y aunque sus prestaciones serán muy limitadas, le servirá para practicar con los menús y para aprender cómo funciona el portapapeles en Visual Basic.

Puesto que ya tenemos diseñado el menú, continuaremos con el resto de las partes que compondrán el formulario.

1. *Inserte un control **TextBox** en el formulario en la posición **120, 120** y con un tamaño de **4335 x 3015**.*
2. *Establezca las siguientes propiedades en los controles de la tabla siguiente:*

a) Establecer las propiedades del formulario (**Form**).

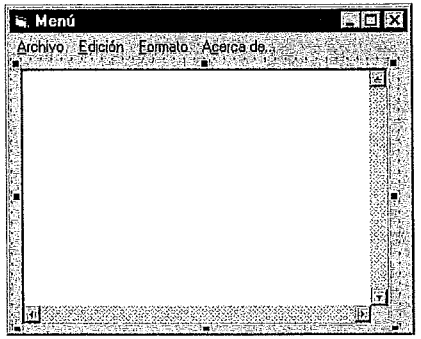
Objeto	Propiedad	Valor
Form1	(Nombre)	Menu
	Caption	Menú

b) Establecer las propiedades del cuadro de texto (**TextBox**).

Objeto	Propiedad	Valor
Text1	(Nombre)	Texto
	MultiLine	True
	ScrollBars	3 - Both
	Text	(Vacío)

*La propiedad **MultiLine** devuelve o establece un valor que indica si un control **TextBox** acepta y muestra múltiples líneas de texto. El valor **True** permite múltiples líneas de texto. El valor **False** (predeterminado) ignora los retornos de carro y restringe los datos a una única línea.*

*La propiedad **ScrollBars** devuelve o establece un valor que indica si un objeto tiene barras de desplazamiento horizontal o vertical. El valor **True** (predeterminado) permite que el formulario tenga una barra de desplazamiento horizontal (**1**), vertical (**2**) o ambas (**3**). Con el valor **False** el formulario no tiene barras de desplazamiento.*



Una vez establecidas las propiedades anteriores, el aspecto de su formulario debe ser como el de la figura de la izquierda.

A continuación comenzaremos a teclear el código de la aplicación.

1. *Haga doble clic sobre una zona vacía del formulario.*
2. *Teclee el siguiente código:*

```
Private Sub Form_Load()
    Texto.Height = ScaleHeight
    Texto.Width = ScaleWidth
    Texto.Left = 0
    Texto.Top = 0
End Sub
```

*Las propiedades **Height** (alto) y **Width** (ancho) devuelven o establecen las dimensiones de un objeto.*

*La propiedad **Left** devuelve o establece la distancia entre el borde interno izquierdo de un objeto y el borde izquierdo de su contenedor.*

*La propiedad **Top** devuelve o establece la distancia entre el borde interno superior de un objeto y el borde superior de su contenedor.*

*La propiedad **ScaleHeight** devuelve la medida vertical del interior de un objeto al usar métodos gráficos o al colocar controles.*

*La propiedad **ScaleWidth** devuelve la medida horizontal del interior de un objeto al usar métodos gráficos o al colocar controles.*

Este código se ejecutará al iniciar la aplicación, pero *¿qué ocurriría si el usuario modifica el tamaño del formulario mientras se ejecuta la aplicación?* Pues sucede que el tamaño de la caja de texto no adoptará las nuevas dimensiones.

Para solucionar esta pequeño inconveniente, hay que programar el evento **Resize**.

3. *Haga clic en la lista de eventos situada en la parte superior derecha de la ventana de Código.*

4. *Seleccione el evento **Resize** y teclee el siguiente código:*

```
Private Sub Form_Resize()
        Texto.Width = ScaleWidth
        Texto.Height = ScaleHeight
End Sub
```

*El evento **Resize** sucede cuando un objeto se muestra primero o cuando cambia el estado de una ventana. (Por ejemplo, cuando se maximiza, minimiza o restaura un formulario).*

Con este procedimiento lo que hacemos es redimensionar de nuevo el tamaño del caja de texto para que se iguale al de la ventana del formulario.

5. *Almacene la aplicación con el nombre **Menú** tanto para el formulario como para el proyecto.*

Creación del menú Archivo

Ahora ha llegado el momento de iniciar la programación de las opciones de cada una de los menús. Comenzaremos con el menú **Archivo**.

1. *Pulse sobre el menú **Archivo** en la ventana del formulario y seleccione la opción **Nuevo**.*

2. *Teclee el siguiente código:*

```
Private Sub MenuANuevo_Click()
        Texto.Text = ""                    'Borra el contenido de la caja de texto
        MenuAGuardar.Enabled = False       'Desactiva la opción Guardar
End Sub
```

Cuadros de diálogo

El control de cuadro de diálogo **CommonDialog** proporciona un conjunto de cuadros de diálogo estándar para operaciones como abrir y guardar archivos, establecer opciones de impresión y seleccionar colores y fuentes. Este control también ofrece la posibilidad de presentar Ayuda mediante el sistema de Ayuda de Windows.

Para usar el control **CommonDialog** si todavía no la ha utilizado debe:

1. *Seleccionar en el menú **Proyecto** la opción **Componentes**.*

2. *Buscar en la pestaña **Controles** la opción **Microsoft Common Dialog Control 6.0** y activarla.*

3. *Hacer clic en el botón* Aceptar.

Observe como en la barra de herramientas aparece un nuevo control cuyo aspecto es el siguiente: .

4. *Haga clic en el control y colóquelo en el formulario.*

5. *En la propiedad **(Nombre)** teclee* **Cuadro**.

6. *Pulse sobre el menú **Archivo** en la ventana del formulario y seleccione la opción **Abrir**.*

7. *Teclee el siguiente código:*

```
Private Sub MenuAAbrir_Click()
        Dim Longitud As Long        'Longitud del fichero abierto
        Dim Textos As String        'Fichero leído

        'Tipos de archivo que se pueden abrir
        Cuadro.Filter = "Todos los archivos (*.*) | *.* | " _
        + "Archivos de texto (*.txt) | *.txt"

        'Visualización del cuadro de diálogo
        Cuadro.ShowOpen

        'Si se ha seleccionado algún fichero
        If Cuadro.FileName <> "" Then
             MenuAGuardar.Enabled = True   'Activa la opción Guardar
             'Apertura y lectura del fichero seleccionado
             Open Cuadro.FileName For Input As #1
             Longitud = FileLen(Cuadro.FileName)
             Textos = Input(Longitud, #1)
             Close #1                      'Cierre del fichero leído

             'Asignación del contenido del fichero al cuadro de texto
             Texto = Textos
        End If
End Sub
```

Ahora explicaremos detalladamente el funcionamiento del procedimiento que acaba de teclear.

*La propiedad **Filter** devuelve o establece los filtros presentados en el cuadro de lista **Tipo de archivos** de un cuadro de diálogo. Su sintaxis es **objeto.Filter [= descripción1 |filtro1 |descripción2 |filtro2...].***

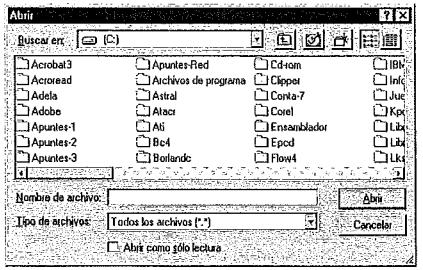
***descripción** es una expresión de cadena que describe el tipo de archivo.*

***filtro** es una expresión de cadena que especifica la extensión del nombre de archivo.*

El filtro especifica el tipo de archivos que se presentan en el cuadro de lista de archivos del cuadro de diálogo. Por ejemplo, si selecciona el filtro *.txt se presentarán todos los archivos de texto.

Utilice esta propiedad siempre que quiera proporcionar al usuario una lista de filtros que se pueden seleccionar cuando se presente el cuadro de diálogo.

La barra vertical (|) (Alt +124) se utiliza para separar los valores descripción y filtro. No incluya espacios en blanco delante o a continuación de la barra vertical, ya que dichos espacios aparecerán en los valores descripción y filtro.



Seguidamente mediante el método **ShowOpen** se abre el cuadro de diálogo.

*El método **ShowOpen** presenta el cuadro de diálogo **Abrir** del control **CommonDialog**.*

Una vez que el usuario ha elegido el archivo desde el cuadro de diálogo, abrimos el fichero con la función **Open**.

*La instrucción **Open** activa operaciones de entrada/salida (E/S) con un archivo.*

*Su sintaxis es **Open rutaacceso [For modo] [Access acceso] [bloquear] As [#]númeroarchivo [Len=longitudregistro].***

***rutaacceso** es una expresión de cadena que especifica un nombre de archivo; puede incluir el directorio o carpeta y la unidad de disco.*

***modo** especifica el modo de archivo: **Append** (Añadir), **Binary** (Binario), **Input** (Lectura), **Output** (Escritura) o **Random** (Aleatorio). Si no se especifica, el archivo se abre con acceso **Random**.*

***acceso** especifica las operaciones permitidas en el archivo abierto: **Read**, **Write** o **Read Write**.*

***bloquear** especifica las operaciones permitidas en el archivo abierto por otros procesos: **Shared**, **Lock Read**, **Lock Write** y **Lock Read Write**.*

***númeroarchivo** es un número de archivo válido entre **1** y **511** inclusive.*

***longitudregistro** es un número igual o inferior a **32.767 bytes**. En archivos abiertos para acceso aleatorio, este valor es la longitud del registro. Para archivos secuenciales, este valor es el número de caracteres almacenados en búfer.*

También se utiliza la propiedad **FileName** del cuadro de diálogo junto a la función **Open** cuando se procede a abrir el fichero.

*La propiedad **FileName** devuelve o establece la ruta y el nombre de un archivo seleccionado.*

Una vez abierto el fichero en el modo de sólo lectura se calcula la longitud del fichero con la función **FileLen**.

*La función **FileLen** devuelve un tipo **Long** que especifica la longitud de un archivo en bytes. Su sintaxis es **FileLen(nombre_ruta)**.*

***nombre_ruta** es una expresión de cadena que especifica un nombre de archivo. Puede incluir el directorio o carpeta y la unidad de disco.*

Para finalizar se abre el fichero abierto anteriormente con la instrucción **Open** y se asigna al control **Texto** el valor de la variable **Textos** (que almacena el contenido del fichero abierto y leído) **Texto = Textos**. Después se cierra el fichero (**Close #1**).

*La instrucción condicional **If Cuadro.Filename <> "" Then ... End** se ha puesto para evitar que se produzca un error en caso de que al abrir el cuadro de diálogo no se seleccione ningún archivo y se pulse el botón* Cancelar.

Seguidamente realizaremos el código de la opción **Guardar**.

8. *Seleccione el objeto **MenuAGuardar** de la lista desplegable de objetos.*

9. *Teclee el siguiente código:*

```
Private Sub MenuAGuardar_Click()
        'Tipos de archivo que se pueden guardar
        Cuadro.Filter = "Archivos de texto (*.txt) | *.txt"

        'Visualización del cuadro de diálogo
        Cuadro.ShowSave
```



```
        'Apertura y lectura del fichero seleccionado
        Open Cuadro.FileName For Output As #1
        Print #1, Texto.Text  'Almacenamiento del texto escrito
        Close #1              'Cierre del fichero leído
End Sub
```

*El método **ShowSave** presenta el cuadro de diálogo **Guardar como** del control **CommonDialog**.*

El funcionamiento de este procedimiento es muy parecido al anterior, sólo que en lugar de abrir el fichero como lectura (**Input**) se hace como escritura (**Output**).

*La instrucción **Print** escribe en un archivo secuencial los datos con formato de presentación. Su sintaxis es **Print #númeroarchivo, [listasalida]**.*

***númeroarchivo** es cualquier número de archivo válido.*

***listasalida** es una expresión o lista de expresiones para imprimir.*

*La instrucción **Input** lee datos de un archivo secuencial abierto y asigna esos datos a variables. Su sintaxis es **Input #númeroarchivo, listavariables**.*

***númeroarchivo** es cualquier número de archivo válido.*

***listavariables** es una lista de variables delimitada por comas que se asignan a los valores que se leen de un archivo; no puede ser una matriz o una variable de objeto.*

10. *Pulse sobre el menú **Archivo** en la ventana del formulario y seleccione la opción **Salir**.*

11. *Teclee el siguiente código:*

```
Private Sub MenuASalir_Click()
        End    'Finaliza el programa
End Sub
```

Para finalizar este apartado dedicado a la creación del menú **Archivo** falta indicar que la opción **Guardar** se active en el momento que haya algún texto escrito.

12. *Seleccione el objeto **Texto** de la lista desplegable de objetos.*

13. *Teclee el siguiente código:*

```
Private Sub Texto_Change()
        If Texto.Text <> "" Then MenuAGuardar.Enabled = True
End Sub
```

Creación del menú Edición

Para crear este menú deberemos utilizar el portapapeles que es un objeto que no posee propiedades ni eventos, pero en cambio si que tiene métodos que permiten transferir datos a y desde su entorno.

*El objeto **Clipboard** proporciona acceso al portapapeles del sistema. Su sintaxis es **Clipboard**.*

Para realizar esta tarea se dispone de los métodos **SetText** y **GetText**.

*El método **SetText** coloca una cadena de texto en el objeto **Clipboard** usando el formato del objeto **Clipboard** especificado. No acepta argumentos con nombre. Su sintaxis es **objeto.SetText datos, formato**.*

***objeto** es una expresión de objeto que da como resultado un objeto de la lista.*

***datos** es una cadena de datos que se va a colocar en el portapapeles.*

***formato** es una constante o un valor que especifica uno de los formatos del portapapeles reconocidos por Visual Basic. Los valores pueden ser:*

***&HBF00** Información de conversación DDE.*

***&HBF01** Formato de texto enriquecido.*

***1** (Predeterminado) Texto.*

*El método **GetText** devuelve una cadena de texto del objeto **Clipboard**. No acepta argumentos con nombre. Su sintaxis es **objeto.GetText (formato)**.*

*Los parámetros tienen el mismo significado que en el método **SetText**.*

***formato** es un valor o una constante que especifica el formato del objeto **Clipboard**. La constante o el valor debe ir entre paréntesis y pueden ser:*

***&HBF00** Información de conversación DDE.*

***1** (Predeterminado) Texto.*

***&HBF01** Formato de texto enriquecido (archivo .rtf).*

El menú **Edición** consta de tres órdenes *Cortar, Copiar* y *Pegar* que inicialmente no pueden estar activas. Las órdenes *Cortar* y *Copiar* estarán activas cuando se haya seleccionado un bloque de texto, y la orden *Pegar* cuando haya texto en el portapapeles.

Para saber si el usuario ha seleccionado por medio del ratón o del teclado un bloque de texto se hará uso de la propiedad **SelLength**.

*La propiedad **SelLength** devuelve o establece el número de caracteres seleccionados.*

1. *Active el formulario **Menu** seleccionándolo con un doble clic desde la ventana **Proyecto**.*

2. *Haga clic sobre el control* .

3. *Seleccione de la lista desplegable de objetos el objeto **MenuEdicion**.*

4. *Teclee el siguiente código:*

```
Private Sub MenuEdicion_Click()
        If Texto.SelLength > 0 Then
                MenuECortar.Enabled = True      'Activa la opción Cortar
                MenuECopiar.Enabled = True      'Activa la opción Copiar
        End If
        If Len(Clipboard.GetText()) > 0 Then
                MenuEPegar.Enabled = True       'Activa la opción Pegar
        End If
End Sub
```

*La función **Len** devuelve un tipo **Long** que contiene el número de caracteres en una cadena o el número de bytes necesarios para almacenar una variable. Su sintaxis es **Len(cadena I nombrevar)**.*

***cadena** es cualquier expresión de cadena válida. Si cadena contiene **Null**, se devuelve **Null**.*

***nombrevar** es cualquier nombre devariable válido. Si **nombrevar** contiene **Null**, se devuelve **Null**. Si **nombrevar** es un tipo **Variant**, **Len** actúa igual que con un tipo **String** y siempre devuelve el número de caracteres que contiene.*

En el procedimiento anterior se comprueba si hay texto seleccionado y si es así se activan las opciones *Cortar* y *Copiar*. Lo mismo sucede con la opción *Pegar* en caso de que haya algo en el portapapeles.

5. *Seleccione de la lista desplegable de objetos el objeto **MenuECortar**.*

6. *Teclee el siguiente código:*

```
Private Sub MenuECortar_Click()
        Clipboard.SetText Texto.SelText
        Texto.SelText = ""
End Sub
```

En este procedimiento se copia el texto seleccionado en el portapapeles y se borra de la caja de texto.

*La propiedad **SelText** devuelve o establece una cadena con el texto seleccionado actualmente o es una cadena de longitud cero ("") si no hay caracteres seleccionados.*

7. *Seleccione de la lista desplegable de objetos el objeto **MenuECopiar**.*

8. *Teclee el siguiente código:*

```
Private Sub MenuECopiar_Click()
        Clipboard.SetText Texto.SelText
End Sub
```

El procedimiento anterior se encarga de copiar el texto seleccionado en el portapapeles.

9. *Seleccione de la lista desplegable de objetos el objeto **MenuEPegar**.*

10. *Teclee el siguiente código:*

```
Private Sub MenuEPegar_Click()
        Texto.SelText = Clipboard.GetText()
End Sub
```

Este procedimiento se encarga de copiar en la caja de texto el contenido del portapapeles.

Ahora ya sólo falta desactivar las opciones de *Cortar* y *Copiar* cada vez que se abra un nuevo documento.

11. *Seleccione el objeto **MenuANuevo** de la lista desplegable de objetos.*

12. *Teclee el siguiente código:*

```
Private Sub MenuANuevo_Click()
        Texto.Text = ""                    'Borra el contenido de la caja de texto
        MenuAGuardar.Enabled = False       'Desactiva la opción Guardar
        MenuECortar.Enabled = False        'Desactiva la opción Cortar
        MenuECopiar.Enabled = False        'Desactiva la opción Copiar
End Sub
```

Creación del menú Formato

En este menú nos encargaremos de activar o desactivar según corresponda las opciones (Negrita, Cursiva y Subrayado).

1. *Seleccione el objeto **MenuFNegrita** de la lista desplegable de objetos.*

2. *Teclee el siguiente código:*

```
Private Sub MenuFNegrita_Click()
        If Texto.FontBold = True Then
                Texto.FontBold = False             'Desactiva Negrita
                MenuFNegrita.Checked = False       'Desactiva marca Negrita
        Else
                Texto.FontBold = True              'Activa Negrita
```

```
            MenuFNegrita.Checked = True     'Activa marca Negrita
        End If
    End Sub
```

*La propiedad **FontBold** permite establecer el tipo de letra negrita. Si tiene el valor **True** activa este estilo de letra. El valor **False** (predeterminado) lo deshabilita.*

En el procedimiento anterior se comprueba si el texto está en negrita, si es así se desactiva tanto el estilo como la marca, en caso contrario se activan el estilo y la marca.

3. *Seleccione el objeto **MenuFCursiva** de la lista desplegable de objetos.*

4. *Teclee el siguiente código:*

```
Private Sub MenuFCursiva_Click()
    If Texto.FontItalic = True Then
        Texto.FontItalic = False            'Desactiva Cursiva
        MenuFCursiva.Checked = False        'Desactiva marca Cursiva
    Else
        Texto.FontItalic = True             'Activa Cursiva
        MenuFCursiva.Checked = True         'Activa marca Cursiva
    End If
End Sub
```

*La propiedad **FontItalic** permite establecer el tipo de letra cursiva. Si tiene el valor **True** activa este estilo de letra. El valor **False** (predeterminado) lo deshabilita.*

5. *Seleccione el objeto **MenuFSubrayado** de la lista desplegable de objetos.*

6. *Teclee el siguiente código:*

```
Private Sub MenuFSubrayado_Click()
    If Texto.FontUnderline = True Then
        Texto.FontUnderline = False         'Desactiva Subrayado
```



```
        MenuFSubrayado.Checked = False  'Desactiva marca Subrayado
    Else
        Texto.FontUnderline = True          'Activa Subrayado
        MenuFSubrayado.Checked = True   'Activa marca Subrayado
    End If
End Sub
```

*La propiedad **Underline** permite establecer el tipo de letra subrayado. Si tiene el valor **True** activa este estilo de letra. El valor **False** (predeterminado) lo deshabilita.*

Llegados a este punto nos volvemos a encontrar con el problema de que se activen las opciones del menú **Formato** en el momento que haya algún texto escrito y se desactiven cuando no haya ninguno. Para solucionar este problema efectúe los pasos que se enumeran a continuación.

7. *Seleccione el objeto **Texto** de la lista desplegable de objetos.*

8. *Teclee el siguiente código:*

```
Private Sub Texto_Change()
    If Texto.Text <> "" Then
        MenuAGuardar.Enabled = True
        MenuFNegrita.Enabled = True     'Activa la opción Negrita
        MenuFCursiva.Enabled = True     'Activa la opción Cursiva
        MenuFSubrayado.Enabled = True   'Activa la opción Subrayado
    End If
End Sub
```

9. *Seleccione el objeto **MenuANuevo** de la lista desplegable de objetos.*

10. *Teclee el siguiente código:*

```
Private Sub MenuANuevo_Click()
    Texto.Text = ""                     'Borra el contenido de la caja de texto
    MenuAGuardar.Enabled = False        'Desactiva la opción Guardar
    MenuECortar.Enabled = False         'Desactiva la opción Cortar
    MenuECopiar.Enabled = False         'Desactiva la opción Copiar
    MenuFNegrita.Enabled = False        'Desactiva la opción Negrita
```

```
        MenuFCursiva.Enabled = False     'Desactiva la opción Cursiva
        MenuFSubrayado.Enabled = False 'Desactiva la opción Subrayado
End Sub
```

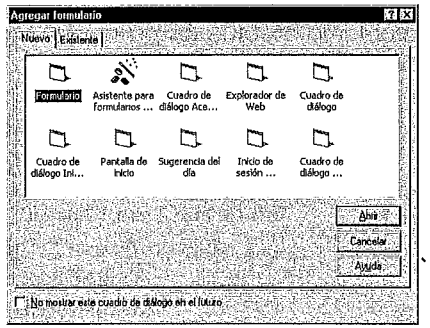
Creación del menú Acerca de...

Este menú casi siempre presente en las aplicaciones de Windows, suele mostrar la versión de la aplicación, el autor y hasta el número de serie del producto.

Para realizar este menú hemos de añadir otro formulario al proyecto, es decir que trabajaremos con dos formularios.

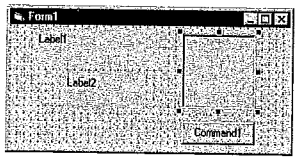
Los pasos a seguir son:

1. *Abra el menú **Proyecto** y seleccione la opción **Agregar formulario**.*
2. *Seleccione **Formulario** y pulse el botón* Abrir.
3. *Seleccione la propiedad **Height** y teclee el valor **2445**.*
4. *Seleccione la propiedad **Width** y teclee el valor **4800**.*
5. *Inserte los siguientes controles en la posición y con el tamaño que se indica a continuación:*



Objeto	Posición	Tamaño
Label1	480, 120	2055 x 495
Label2	960, 840	1215 x 495
Command1	2880, 1560	1215 x 375
Picture1	2880, 120	1215 x 1215

A continuación establezca las siguientes propiedades en el formulario y sus controles correspondientes:



a) Propiedades del formulario (**Form**).

Objeto	Propiedad	Valor
Form1	(Nombre)	Acercade
	Caption	Acerca de...

b) Establecer las propiedades del botón (**CommandButton**).

Objeto	Propiedad	Valor
Command1	(Nombre)	Aceptar
	Caption	&Aceptar

c) Establecer las propiedades de las etiquetas (**Label**).

Objeto	Propiedad	Valor
Label1	(Nombre)	Nombre
	Alignment	2 - Center
	Caption	Editor de Textos
	Font	Tamaño - 12
	Estilo de fuente	Negrita
Label2	(Nombre)	Version
	Alignment	2 - Center
	Caption	Versión 1.0
	Font	Tamaño - 10
	Estilo de fuente	Negrita

d) Establecer las propiedades del cuadro de imagen (**PictureBox**).

Objeto	Propiedad	Valor
Picture1	(Nombre)	Imagen
	Picture	Carpeta Metafile
		Bussines
		Pccomputr.wmf

6. *Active el formulario **Menu** seleccionándolo desde la ventana **Proyecto**.*
7. *Haga clic en la opción **Acerca de...***
8. *Teclee el siguiente código:*

```
Private Sub MenuAcerca_Click()
        Acercade.Show   'Muestra el formulario
End Sub
```

*El método **Show** provoca que el formulario se presente en pantalla por encima del formulario actual. Si el formulario especificado no está cargado cuando se invoca el método **Show**, Visual Basic lo carga automáticamente.*

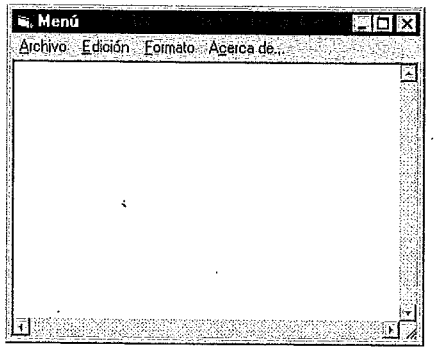
*Cuando **Show** presenta un formulario no modal (valor **0**), continúa con la ejecución del código que haya a continuación. Cuando **Show** presenta un formulario modal (valor **1**), el código que hay a continuación no se ejecuta hasta que el formulario se oculta o se descarga.*

9. *Active el formulario **Acercade** seleccionándolo con un doble clic desde la ventana **Proyecto**.*
10. *Haga doble clic en el botón **Aceptar**.*
11. *Teclee el siguiente código:*

```
Private Sub Aceptar_Click()
        Unload Acercade
End Sub
```

*La instrucción **Unload** descarga de memoria un formulario o un control. Su sintaxis es **Unload objeto**.*

***objeto** es el nombre de un objeto **Form** o de un elemento de una matriz de controles para descargar.*



12. *Archive el proyecto dejando el nombre que sugiere por defecto **Visual Basic** para el formulario que hemos añadido.*
13. *Guarde y ejecute la aplicación.*

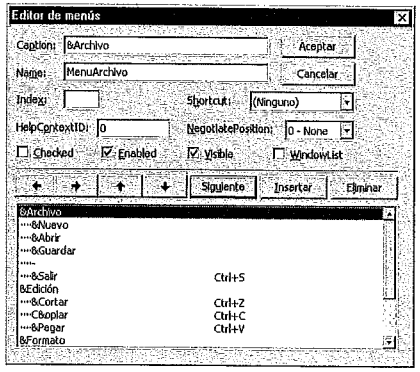
INSTITUTO DE EDUCACION SECUNDARIA "ARROYO HARINAS" ALMENDRALEJO (Badajoz)

Mejorando el Editor de textos

Sin duda alguna se habrá dado cuenta que el editor de textos que acabamos de crear podría ser mejorado, incorporando alguna nueva opción como, **Imprimir**, **Fuentes**, **Tamaño**, etc.

Mejorando el menú Archivo

Una de las principales opciones que debe incorporar cualquier editor de textos, es la de imprimir puesto que de no ser así, su utilidad descendería notablemente.



1. *Active el formulario **Menu**.*
2. *Haga clic sobre el botón* .
3. *Efectúe un clic sobre el separador que hay en el **Editor de menús**.*
4. *Pulse el botón* Insertar.
5. *En la nueva opción que aparece establezca las siguientes propiedades:*

Caption: ***&Imprimir***

Name: ***MenuAImprimir***

Shortcut: ***Ctrl + P***

Enabled: ***Desactivado***

6. *Pulse el botón* [Aceptar].

El aspecto del menú **Archivo** debe ser el siguiente:

Ahora sólo falta modificar el código del programa para activar la opción de imprimir cuando haya algún texto en pantalla.

1. *Seleccione el objeto* ***Texto*** *de la lista desplegable de objetos.*

2. *Teclee el siguiente código:*

```
Private Sub Texto_Change()
    If Texto.Text <> "" Then
        MenuAGuardar.Enabled = True
        MenuAImprimir.Enabled = True   'Activa la opción Imprimir
        MenuFNegrita.Enabled = True    'Activa la opción Negrita
        MenuFCursiva.Enabled = True    'Activa la opción Cursiva
        MenuFSubrayado.Enabled = True  'Activa la opción Subrayado
    End If
End Sub
```

3. *Selecccione el objeto* ***MenuANuevo*** *de la lista desplegable de objetos.*

4. *Teclee el siguiente código:*

```
Private Sub MenuANuevo_Click()
    Texto.Text = ""                    'Borra el contenido de la caja de texto
    MenuAGuardar.Enabled = False       'Desactiva la opción Guardar
    MenuAImprimir.Enabled = False      'Desactiva la opción Imprimir
    MenuECortar.Enabled = False        'Desactiva la opción Cortar
    MenuECopiar.Enabled = False        'Desactiva la opción Copiar
    MenuFNegrita.Enabled = False       'Desactiva la opción Negrita
    MenuFCursiva.Enabled = False       'Desactiva la opción Cursiva
```



```
    MenuFSubrayado.Enabled = False 'Desactiva la opción Subrayado
End Sub
```

Por último sólo falta activar el método **ShowPrinter** para poder imprimir el texto que tengamos en pantalla.

1. *Seleccione el objeto* ***MenuAImprimir*** *de la lista de objetos.*

2. *Teclee el siguiente código:*

```
Private Sub MenuAImprimir_Click()
    Cuadro.ShowPrinter
    Printer.Print Texto
    Printer.EndDoc
End Sub
```

El método ***ShowPrinter*** *presenta el cuadro de diálogo* ***Imprimir*** *del control* ***CommonDialog****. Su sintaxis es* ***objeto.ShowPrinter****.*

Mejorando el menú Formato

El menú **Formato** lo mejoraremos incorporando tres nuevas opciones: **Fuentes**, **Tamaño** y **Color**.

1. *Active el formulario* ***Menu****.*

2. *Haga clic sobre el botón* [ ].

3. *Efectúe un clic sobre la opción* ***A&cerca de...*** *que hay en el* ***Editor de menús****.*

4. *Pulse el botón* [Insertar].

5. *Efectúe un clic sobre el botón* [→] *para crear una subopción.*

6. *En la nueva opción que aparece establezca las siguientes propiedades:*

Caption:

Name: ***MenuFSeparador***

7. *Pulse la tecla* ***Enter***.

8. *Pulse el botón* [Insertar].

9. *Efectúe un clic sobre el botón* [→] *para crear una subopción.*

10. *Establezca las siguientes propiedades:*

    Caption: ***&Fuentes***

    Name: ***MenuFFuentes***

11. *Pulse la tecla* ***Enter***.

12. *Pulse el botón* [Insertar].

13. *Efectúe un clic sobre el botón* [→] *para crear una subopción.*

14. *Establezca las siguientes propiedades:*

    Caption: ***&Tamaño***

    Name: ***MenuFTamaño***

15. *Pulse la tecla* ***Enter***.

16. *Pulse el botón* [Insertar].

17. *Efectúe un clic sobre el botón* [→] *para crear una subopción.*

18. *Establezca las siguientes propiedades:*

    Caption: ***C&olor***

    Name: ***MenuFColor***

19. *Pulse el botón* [Aceptar].

El aspecto del menú **Formato** deberá ser como el de la figura de la derecha.

Ahora sólo falta activar el método **ShowFont** para poder cambiar la fuente del texto que tengamos en pantalla.

1. *Seleccione el objeto* ***MenuFFuentes*** *de la lista de objetos.*

2. *Teclee el siguiente código:*

```
Private Sub MenuFFuentes_Click()
        Cuadro.Flags = cdlCFBoth
        Cuadro.ShowFont
        Texto.Font = Cuadro.FontName
End Sub
```

La método ***ShowFont*** *presenta el cuadro de diálogo* ***Fuente*** *del control* ***CommonDialog****. Su sintaxis es* ***objeto.ShowFont****.*

Antes de usar el método ***ShowFont*** *tiene que asignar a la propiedad* ***Flags*** *del control* ***CommonDialog*** *una de estas tres constantes o valores:* ***cdlCFBoth*** *o* ***&H3****,* ***cdlCFPrinterFonts*** *o* ***&H2****, o* ***cdlCFScreenFonts*** *o* ***&H1****. Si no establece* ***Flags****, aparecerá un cuadro de mensajes avisando que "No hay ninguna fuente instalada" y se producirá un error en tiempo de ejecución.*

A continuación escribiremos el código correspondiente a la opción que permite cambiar el tamaño de la fuente de letra. Para ello utilizaremos el método **ShowSize**.

1. *Seleccione el objeto* ***MenuFTamaño*** *de la lista de objetos.*

2. *Teclee el siguiente código:*

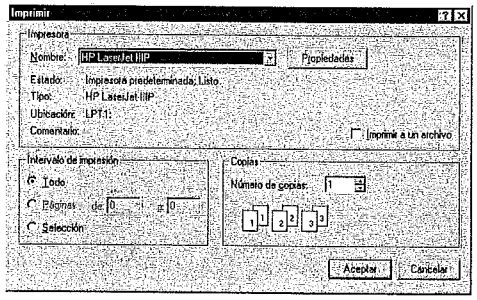
```
Private Sub MenuFTamaño_Click()
        Cuadro.Flags = cdlCFBoth
        Cuadro.ShowFont
        Texto.FontSize = Cuadro.FontSize
End Sub
```

*La propiedad **FontSize** devuelve o establece el tamaño de la fuente que se va a usar para el texto mostrado en un control o en una operación de dibujo o impresión en tiempo de ejecución.*

1. *Seleccione el objeto **MenuFColor** de la lista de objetos.*
2. *Teclee el siguiente código:*

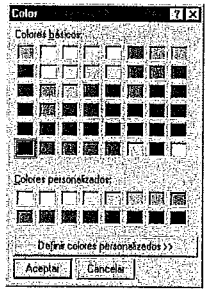
```
Private Sub MenuFColor_Click()
    Cuadro.ShowColor
    Texto.ForeColor = Cuadro.Color
End Sub
```

*El método **ShowColor** presenta el cuadro de diálogo **Color** del control **CommonDialog**. Su sintaxis es **objeto.ShowColor**.*



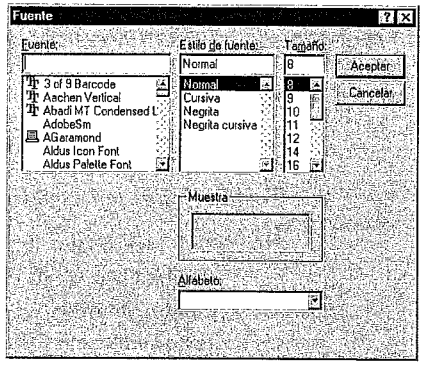
Si ejecuta el programa y efectúa un clic sobre la opción **Imprimir** del menú **Archivo** verá como aparece el típico cuadro de diálogo de imprimir.

Si selecciona la opción **Fuentes** del menú **Formato** observará que aparece el cuadro de diálogo de **Fuente**.



Si opta por elegir la opción **Tamaño** del menú **Formato** también aparecerá el cuadro de diálogo de **Fuente**.

El cuadro de diálogo **Color** aparece si selecciona la opción **Color** dentro del menú **Formato**.



3. *Almacene de nuevo la aplicación.*

Un nuevo control para mejorar la aplicación

Sin duda alguna se habrá dado cuenta que en la aplicación que acabamos de realizar, siempre que se asigna un formato al texto, éste afecta a todo el texto que hay escrito. Pues bien, esto no es del todo correcto si queremos que se comporte como si de un verdadero editor de textos se tratara.

Para solucionar este problema se puede utilizar el control **RichTextBox** (cuadro de texto enriquecido).

*El control **RichTextBox** permite al usuario escribir y modificar texto al tiempo que proporciona características de formato más avanzadas que el control **TextBox** convencional. Su sintaxis es **RichTextBox**.*

El control **RichTextBox** proporciona una serie de propiedades que puede usar para aplicar formato a cualquier parte del texto de un control. Para cambiar el formato del texto, debe seleccionarlo previamente. Sólo puede asignar formato de carácter y de párrafo al texto seleccionado. Mediante estas propiedades puede poner el texto en negrita o en cursiva, cambiar el color, y crear superíndices y subíndices. También puede ajustar el formato de párrafo si establece sangrías izquierda y derecha, así como sangrías francesas.

Una de las ventajas que presenta este control es la de no poseer un límite de caracteres a la hora de escribir un texto.

A continuación realizaremos los pasos necesarios para modificar el código del editor de textos utilizando el control **RichTextBox**. Lo primero que deberá hacer es insertar dicho control en el cuadro de herramientas. Para ello:

1. *Seleccione en el menú **Proyecto** la opción **Componentes**.*
2. *Busque en la pestaña **Controles** la opción **Microsoft RichTextBox Control 6.0** y actívela.*

3. *Haga clic en el botón* [Aceptar].

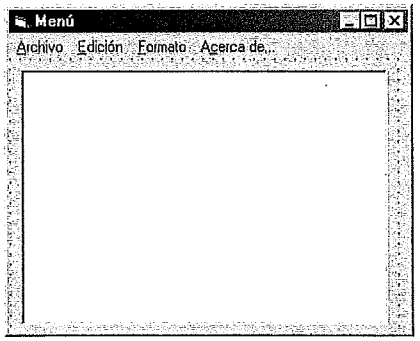
Observe como en la barra de herramientas aparece un nuevo control cuyo aspecto es el siguiente: [icono].

Ahora sustituiremos el control de cuadro de texto **TextBox** que hay en el formulario por el control **RichTextBox**.

1. *Active el formulario **Menú** y elimine el cuadro de texto.*
2. *Inserte el control **RichTextBox** en el formulario en la posición **120, 120** con un tamaño de **4335 x 3015**.*
3. *Establezca las siguientes propiedades en el nuevo control:*

Objeto	Propiedad	Valor
RichTextBox1	(Nombre)	Texto
	ScrollBars	3 - Both
	Text	(Vacío)

El nuevo aspecto del formulario será el siguiente:



4. *Haga clic sobre el botón* [icono].
5. *Active la opción **Enabled** de las opciones **Negrita**, **Cursiva** y **Subrayado**.*

Seguidamente pasaremos a modificar el código de los objetos, necesario para el nuevo funcionamiento de la aplicación.

1. *Seleccione el objeto **MenuANuevo** y deje el código tal y como aparece a continuación:*

```
Private Sub MenuANuevo_Click()
        Texto.Text = ""              'Borra del contenido de la caja de texto
        MenuAGuardar.Enabled = False    'Desactiva la opción Guardar
        MenuAImprimir.Enabled = False   'Desactiva la opción Imprimir
        MenuECortar.Enabled = False     'Desactiva la opción Cortar
```



```
        MenuECopiar.Enabled = False     'Desactiva la opción Copiar
End Sub
```

2. *Seleccione el objeto **MenuFNegrita** y deje el código tal y como aparece a continuación:*

```
Private Sub MenuFNegrita_Click()
        If Texto.SelBold = False Then
                Texto.SelBold = True   'Activa Negrita
        Else
                Texto.SelBold = False  'Desactiva Negrita
        End If
End Sub
```

*Las propiedades **SelBold, SelItalic, SelStrikethru** y **SelUnderline** devuelven o establecen estilos de fuente del texto seleccionado actualmente en un control **RichTextBox**. Los estilos de fuente incluyen los formatos siguientes: Negrita, Cursiva, Tachado y Subrayado. No está disponible en tiempo de diseño. Su sintaxis es:*

objeto.*SelBold* [= valor]

objeto.*SelItalic* [= valor]

objeto.*SelStrikethru* [= valor]

objeto.*SelUnderline* [= valor]

***valor** puede tomar el valor **Null, True** o **False**.*

3. *Seleccione el objeto **MenuFCursiva** y deje el código tal y como aparece a continuación:*

```
Private Sub MenuFCursiva_Click()
        If Texto.SelItalic = False Then
                Texto.SelItalic = True   'Activa Cursiva
        Else
                Texto.SelItalic = False  'Desactiva Cursiva
        End If
End Sub
```

4. *Seleccione el objeto **MenuFSubrayado** y deje el código tal y como aparece a continuación:*

```
Private Sub MenuFSubrayado_Click()
        If Texto.SelUnderline = False Then
                Texto.SelUnderline = True   'Activa Subrayado
        Else
                Texto.SelUnderline = False  'Desactiva Subrayado
        End If
End Sub
```

5. *Seleccione el objeto **MenuFFuentes** y deje el código tal y como aparece a continuación:*

```
Private Sub MenuFFuentes_Click()
        Cuadro.Flags = cdlCFBoth
        Cuadro.ShowFont
        Texto.SelFontName = Cuadro.FontName
End Sub
```

*La propiedad **SelFontName** devuelve o establece la fuente usada para mostrar el texto seleccionado actualmente o los caracteres que siguen inmediatamente al punto de inserción en el control **RichTextBox**. No está disponible en tiempo de diseño. Su sintaxis es* **objeto.*SelFontName* [= cadena]**.

6. *Seleccione el objeto **MenuFTamaño** y deje el código tal y como aparece a continuación:*

```
Private Sub MenuFTamaño_Click()
        Cuadro.Flags = cdlCFBoth
        Cuadro.ShowFont
        Texto.SelFontSize = Cuadro.FontSize
End Sub
```

*La propiedad **SelFontSize** devuelve o establece un valor que especifica el tamaño de la fuente usada para mostrar texto en un control **RichTextBox**. No está disponible en tiempo de diseño. Su sintaxis es* **objeto.*SelFontSize* [= puntos]**.

***puntos** especifica el tamaño en puntos del texto.*

7. *Seleccione el objeto **MenuFColor** y deje el código tal y como aparece a continuación:*

```
Private Sub MenuFColor_Click()
        Cuadro.ShowColor
        Texto.SelColor = Cuadro.Color
End Sub
```

*Devuelve o establece un valor que determina el color del texto del control **RichTextBox**. No está disponible en tiempo de diseño. Su sintaxis es* **objeto.*SelColor* [= color]**.

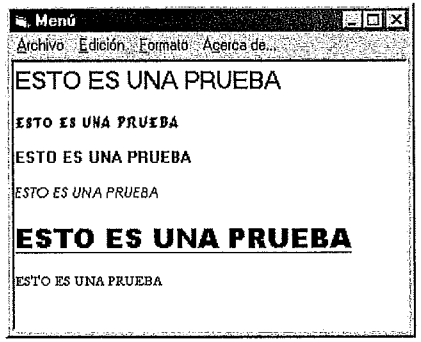
***color** es un valor que especifica un color, este puede ser **Null**, un color RGB o un color del sistema.*

8. *Seleccione el objeto **Texto** y deje el código tal y como aparece a continuación:*

```
Private Sub Texto_Change()
        If Texto.Text <> "" Then
                MenuAGuardar.Enabled = True     'Activa la opción Guardar
                MenuAImprimir.Enabled = True    'Activa la opción Imprimir
        End If
End Sub
```

9. *Almacene la aplicación con el nombre **MenuMejorado** tanto para el formulario como para el proyecto.*

10. *Ejecute la aplicación.*



Observe que si prueba a escribir texto, puede alternar diferentes tipos de letra, así como tamaños, colores y aspectos (cursiva, negrita y subrayado).

También puede utilizar los métodos **LoadFile** y **SaveFile** para cargar y almacenar respectivamente un archivo.

*El método **LoadFile** carga un archivo .rtf o un archivo de texto en un control **RichTextBox**. Su sintaxis es* **objeto.*LoadFile* nombreRuta, tipoArchivo.**

*El método **SaveFile** guarda en un archivo el contenido de un control **RichTextBox**. Su sintaxis es* **objeto.*SaveFile* (nombreRuta, tipoArchivo).**

11. El ratón

Los formularios y casi todos los controles reconocen tres eventos producidos por el ratón: **MouseDown**, **MouseUp** y **MouseMove**.

Los eventos del ratón son reconocidos por el formulario cuando el puntero del ratón está en una zona en la que no hay ningún control, y un control reconoce un evento del ratón cuando el puntero del mismo está sobre el propio control.

*El evento **MouseDown** es reconocido cuando se pulsa cualquier botón del ratón.*

*El evento **MouseUp** es reconocido cuando el usuario suelta cualquier botón del ratón.*

*Utilice un procedimiento de evento **MouseDown** o **MouseUp** para especificar acciones que ocurrirán cuando se presione o se suelte un determinado botón del* mouse. *A diferencia de los eventos **Click** y **DblClick**, los eventos **MouseDown** y **MouseUp** le permiten distinguir entre los botones primario, secundario y central. También puede escribir código para combinaciones de teclado y* mouse *que utilicen los modificadores de teclado **MAYÚS**, **CTRL** y **ALT**.*

Si necesita comprobar los argumentos botón *o* mayús *puede usar las constantes que se enumeran:*

Constante (Botón)	Valor	Descripción
vbLeftButton	1	Está pulsado el botón primario.
vbRightButton	2	Está pulsado el botón secundario.
vbMiddleButton	4	Está pulsado el botón central.

Constante (Mayús)	Valor	Descripción
vbShiftMask	1	Está pulsada la tecla MAYÚS.
vbCtrlMask	2	Está pulsada la tecla CTRL.
vbAltMask	4	Está pulsada la tecla ALT.

*El evento **MouseMove** es reconocido cuando el usuario mueve el ratón de una posición a otra.*