Introducción a Visual Basic

Comentario General

¿Qué es Visual Basic? 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 de la interfaz, simplemente puede arrastrar y colocar objetos prefabricados en su lugar dentro de la pantalla. Si ha utilizado alguna vez un programa de dibujo como *Paint*, ya tiene la mayor parte de las habilidades necesarias para crear una interfaz de usuario efectiva.

La palabra "Basic" hace referencia al lenguaje BASIC (Beginners All-Purpose Symbolic Instruction Code), un lenguaje utilizado por más programadores que ningún otro lenguaje en la historia de la informática o computación. 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. Los principiantes pueden crear aplicaciones útiles con sólo aprender unas pocas palabras clave, pero, al mismo tiempo, la eficacia del lenguaje permite a los profesionales acometer cualquier objetivo que pueda alcanzarse mediante cualquier otro lenguaje de programación de Windows.

El lenguaje de programación Visual Basic no es exclusivo de Visual Basic. La Edición para aplicaciones del sistema de programación de Visual Basic, incluida en Microsoft Excel, Microsoft Access y muchas otras aplicaciones Windows, utilizan el mismo lenguaje. El sistema de programación de Visual Basic, Scripting Edition (VBScript) para programar en Internet es un subconjunto del lenguaje Visual Basic. La inversión realizada en el aprendizaje de Visual Basic le ayudará a abarcar estas otras áreas.

Si su objetivo es crear un pequeño programa para su uso personal o para su grupo de trabajo, un sistema para una empresa o incluso aplicaciones distribuidas de alcance mundial a través de Internet, Visual Basic dispone de las herramientas que necesita.

- Las características de acceso a datos le permiten crear bases de datos y aplicaciones cliente 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 utilizar la funcionalidad proporcionada por otras aplicaciones, como el procesador de textos Microsoft Word, la hoja de cálculo Microsoft Excel y otras aplicaciones Windows. Puede incluso automatizar las aplicaciones y los objetos creados con la Edición profesional o la Edición empresarial de Visual Basic.
- Las capacidades de Internet facilitan el acceso a documentos y aplicaciones a través de Internet desde su propia aplicación.
- La aplicación terminada es un auténtico archivo .exe que utiliza una biblioteca de vínculos dinámicos (DLL) de tiempo de ejecución que puede distribuir con toda libertad.

Diferentes versiones

Versión	Sistema Operativo	Compilación	Acceso a base de datos Access (.mdb)
Visual Basic 3.0	Windows 3.x	16 bits	Access 1.0 y 2.0
Visual Basic 4.0	Windows 95 / 98 / 3. X / NT 4.0	16 y 32 bits (compilación condicional)	Access 95
Visual Basic 5.0	Windows 95 / 98 / NT 4.0	32 bits	Access 97
Visual Basic 6.0	Windows 95 / 98 / Me / 2000 / NT 4.0	32 bits	Access 97 y 2000

Diferentes ediciones

Visual Basic se encuentra disponible en tres versiones, cada una de las cuales está orientada a unos requisitos de programación específicos:

- La *Edición de aprendizaje* de Visual Basic permite a los programadores crear robustas aplicaciones para Microsoft Windows y Windows NT®. Incluye todos los controles intrínsecos, además de los controles de cuadrícula, de fichas y los controles enlazados a datos. La documentación que se proporciona con esta edición incluye "Aprenda Visual Basic ya" junto con el de la biblioteca de Microsoft Developer Network (MSDN™), que contienen documentación completa en pantalla.
- La *Edición profesional* proporciona a los profesionales 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. La documentación que se proporciona con la Edición profesional incluye el libro *Características empresariales de Visual Studio* más los CD de Microsoft Developer Network que contienen documentación completa en pantalla.
- La *Edición empresarial* permite a los profesionales crear sólidas aplicaciones distribuidas en un entorno de equipo. Incluye todas las características de la Edición profesional, así como herramientas de Back Office como SQL Server, Microsoft Transaction Server, Internet Information Server, Visual SourceSafe, SNA Server, etc. La documentación impresa que se proporciona con la Edición empresarial incluye el libro *Características empresariales de Visual Studio* más los CD de Microsoft Developer Network que contienen documentación completa en pantalla.

Proyecto

Definición de Proyecto en Visual Basic

Para crear una aplicación con Visual Basic se trabaja con proyectos. Un *proyecto* es una colección de archivos que se usan para generar una aplicación.
Al crear una aplicación probablemente creará nuevos formularios; también puede volver a usar o modificar formularios creados en proyectos anteriores. Esto también se aplica a otros módulos o archivos que pueda incluir en su proyecto. Los controles ActiveX y los objetos de otras aplicaciones también se pueden compartir entre proyectos.
Después de ensamblar todos los componentes de un proyecto y escribir el código, puede compilar el proyecto para crear un archivo ejecutable.

Componentes de un Proyecto

Cuando desarrolla un aplicación, trabaja con un archivo de proyecto para administrar todos los diferentes archivos que crea. Un proyecto consta de lo siguiente:

- Un archivo de proyecto que realiza el seguimiento de todos los componentes (*.vbp*)
- Un archivo para cada formulario (*.frm*).
- Un archivo de datos binario para cada formulario que contenga datos sobre propiedades de controles del formulario (*.frx*). Estos archivos no se pueden modificar y los genera automáticamente cualquier archivo *.frm* que tenga propiedades en formato binario, como *Picture* o *Icon*.
- Opcionalmente, un archivo para cada módulo de clase (*.cls*).
- Opcionalmente, un archivo para cada módulo estándar (*.bas*).
- Opcionalmente, uno o más archivos con controles ActiveX (*.ocx*).
- Opcionalmente, un único archivo de recursos (.res).

El *archivo de proyecto* es simplemente una lista de todos los archivos y objetos asociados con el proyecto, así como información sobre las opciones de entorno establecidas. Esta información se actualiza cada vez que guarda el proyecto. Todos los archivos y objetos también se pueden compartir con otros proyectos.
Cuando ha completado todos los archivos del proyecto puede convertir el proyecto en un archivo ejecutable (.exe): en el menú *Archivo*, elija el comando *Generar proyecto.exe*.

Formularios

Un formulario es una ventana. La ventana Windows de cualquier aplicación.
Podemos abrir tantas ventanas como queramos en nuestro proyecto, pero el nombre de las ventanas debe ser distinto. Por defecto, la ventana que se abre en Visual Basic tiene el nombre de *Form1*.
Los módulos de formularios (extensión de nombre de archivo .frm) pueden contener descripciones en forma de texto del formulario y sus controles, incluyendo los valores de sus propiedades. También pueden contener declaraciones a nivel de formulario de constantes, variables y procedimientos externos, procedimientos de evento y procedimientos generales.

Módulos de clase

Los módulos de clase (extensión de nombre de archivo .cls) son similares a los módulos de formulario, excepto en que no tiene interfaz de usuario visible. Puede usar módulos de clase para crear sus propios objetos, incluyendo código para métodos y propiedades.

Módulos estándar

Un módulo es un archivo Visual Basic donde escribimos parte del código de nuestro programa, y digo parte, porque puede haber código en el formulario también.
Las rutinas incluidas dentro de los módulos pueden ser ejecutadas desde los formularios de la aplicación.
Los módulos estándar (extensión de nombre de archivo .bas) pueden contener declaraciones públicas o a nivel de módulo de tipos, constantes, variables, procedimientos externos y procedimientos públicos.

Archivos de Recursos

Los archivos de recursos (extensión de nombre de archivo .res) contienen mapas de bits, cadenas de texto y otros datos que puede modificar sin volver a modificar el código. Por ejemplo, si piensa traducir su aplicación a un idioma extranjero, puede guardar todas las cadenas de texto de la interfaz de usuario y los mapas de bits en un archivo de recursos, y simplemente traducir el archivo de recursos en vez de la aplicación completa. Un proyecto sólo puede contener un archivo de recursos.

Controles Active X

Los controles *ActiveX* (extensión de nombre de archivo .ocx) son controles opcionales que se pueden agregar al cuadro de herramientas y se pueden usar en formularios. Cuando instala Visual Basic, los archivos que contienen los controles incluidos en Visual Basic se copian a un directorio común (el subdirectorio \Windows\System). Existen controles *ActiveX* adicionales disponibles en diversas fuentes. También puede crear sus propios controles mediante las ediciones Profesional y Empresarial de Visual Basic.

Controles estándar

Los controles estándar los proporciona Visual Basic. Los controles estándar, como *CommandButton* (botón de comando) o *Frame* (marco), siempre están incluidos en el cuadro de herramientas, al contrario de lo que ocurre con los controles *ActiveX* y los objetos insertables, que se pueden agregar y quitar del cuadro de herramientas.

Programación orientada a objetos

La programación orientada a objetos es una forma de programación que utiliza objetos, ligados entre mensajes, para la resolución de problemas. Puede considerarse como una extensión de la programación estructurada en un intento de potenciar los conceptos de modularidad y reutilización del código.

Objetos

Los programas tradicionales se componen de procedimientos y de datos. Un programa orientado a objetos se compone solamente de objetos. Un objeto es una encapsulación genérica de datos y de los procedimientos para manipularlos. Dicho de otra manera, un objeto es una entidad que tiene asociado un conjunto de atributos particulares: métodos, eventos y propiedades. Ejemplo: Una caja de texto (*TextBox*) es un objeto, el ancho, el alto, etc. son propiedades. Las rutinas que permiten maximizar ó minimizar la caja de texto son métodos.

Propiedades

Son las características ó atributos que posee un objeto (ventana de Windows). Ejemplo: Color de fondo del formulario, Fuente de texto de un *TextBox*, ...

Métodos

Los métodos son funciones internas de un determinado objeto que permite realizar funciones sobre él o sobre otro objeto en respuesta a un determinado estímulo (evento - mensaje).
Ejemplo: Deseamos poner en la ventana Windows de nuestra aplicación “Hola Mundo”, por lo tanto pondremos el método: *Form1.Print "Hola mundo"*

Eventos

Un evento es un estímulo que recibe un objeto, por el cual, hace que se desate un procedimiento asociado. Un programa Visual Basic es un POE (Programa orientado a eventos).
Todo lo que hacemos en un programa Visual Basic está generado por medio de eventos.

Clases

Una clase es un tipo de objeto definido por el usuario, es una generalización de un tipo específico de objeto. Por ejemplo la cubetera es la clase, los cubitos son los objetos.
Para crear un objeto es necesario definir una variable que invoque a la clase del objeto a crear. Ejemplo:

```
Dim caja As TextBox
```

Características de la programación orientada a objetos

Abstracción

Por medio de la abstracción podremos observar un problema un en escenario en particular abstrayéndonos de las particularidades no esenciales.
Ejemplo: Guardaremos un registro en disco sin interesarnos en que pista se almacenará.

Encapsulamiento

Nos permite tomar a los objetos como cajas negras, o sea como unidades elementales.

Herencia

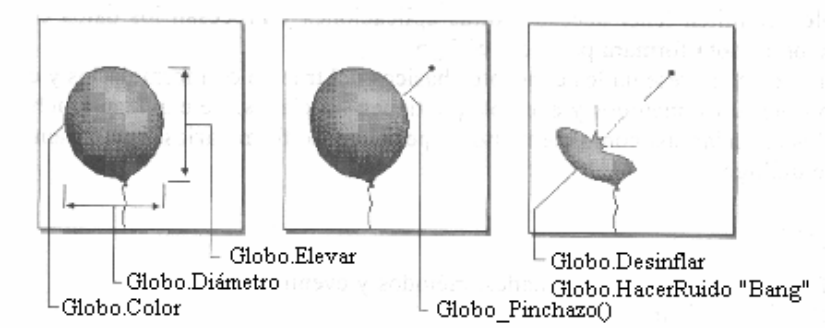
Es la encargada de compartir automáticamente métodos y datos entre clases y subclases. No disponible en Visual Basic.

Polimorfismo

Permite utilizar un mismo método de diferentes formas, dependiendo de la clase sobre la cual se aplique.

Explicación integrada y ejemplo de Objetos, 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.
Un objeto de uso diario como el globo de un niño tiene también propiedades, métodos y eventos. Entre las propiedades de un globo se incluyen atributos visibles como el peso, el diámetro y el color. Otras propiedades describen su estado (inflado o desinflado) o atributos que no son visibles, como su edad. Por definición, todos los globos tienen estas propiedades; lo que varía de un globo a otros son los valores de estas propiedades.
Un globo tiene también métodos o acciones inherentes que puede efectuar. Tiene un método inflar (la acción de llenarlo de helio) o un método desinflar (expeler su contenido) y un método elevarse (si se deja escapar). De nuevo, todos los globos pueden efectuar estos métodos.
Los globos tienen además respuestas predefinidas a ciertos eventos externos. Por ejemplo, un globo respondería al evento de pincharlo desinflándose o al evento de soltarlo elevándose en el aire.
Los objetos tienen propiedades, responden a eventos y ejecutan métodos:



Si se pudiera programar un globo, el código de Visual Basic podría ser como el siguiente. Para establecer las propiedades del globo:

```
Globo.Color = Rojo
Globo.Diámetro = 10
Globo.Inflado = True
```

Observe la sintaxis del código: el objeto (Globo) seguido de la propiedad (Color) seguida de la asignación del valor (Rojo). Podría modificar el color del globo desde el código si repitiera esta instrucción y sustituyera el valor por otro diferente. Las propiedades también se pueden establecer en la ventana Propiedades mientras se está diseñando la aplicación.
Los métodos de un globo se invocan de esta forma:

```
Globo.Inflar
Globo.Desinflar
Globo.Elevar 5
```

La sintaxis es similar a la sintaxis de las propiedades: el objeto (un nombre) seguido de un método (un verbo). En el tercer ejemplo hay un elemento adicional, llamado argumento, que indica la distancia que se eleva. Algunos métodos tendrán uno o más argumentos para describir más a fondo la acción que se va a ejecutar.
El globo puede responder a un evento como se muestra a continuación:

```
Sub Globo_Pinchazo()
   Globo.Desinflar
   Globo.HacerRuido "Bang"
   Globo.Inflado = False
   Globo.Diámetro = 1
End Sub
```

En este caso, el código describe el comportamiento del globo cuando se produce un evento Pinchazo: invoca el método Desinflar y luego invoca el método HacerRuido con un argumento “Bang” (el tipo de ruido que se va a hacer). Como el globo ya no está inflado, la propiedad Inflado tiene el valor False y la propiedad Diámetro adopta un nuevo valor.

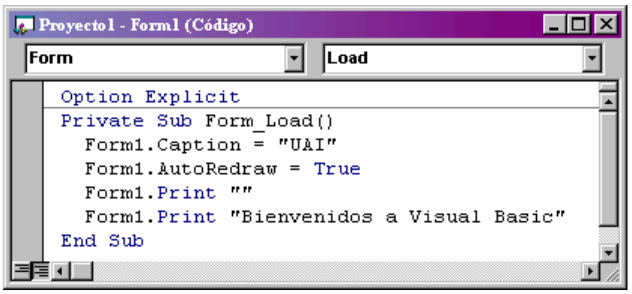
Si bien no puede programar un globo, sí puede programar un formulario o un control de Visual Basic. Como programador, tiene el control: decide qué propiedades se deben modificar, qué métodos se deben invocar o a qué eventos hay que responder para conseguir la apariencia y el comportamiento deseados.

Diferencias entre la programación procedural y la programación bajo Windows

Un estudio profundo del funcionamiento interno de Windows necesitaría un libro completo. No es necesario tener un profundo conocimiento de todos los detalles técnicos. Una versión reducida del funcionamiento de Windows incluye tres conceptos clave: ventanas, eventos y mensajes.
Una ventana es simplemente una región rectangular con sus propios límites. Probablemente ya sabe que hay varios tipos de ventanas: una ventana Explorador en Windows 98, una ventana de documento dentro de su programa de proceso de textos o un cuadro de diálogo que emerge para recordarle una cita. Aunque éstos son los ejemplos más comunes, realmente hay otros muchos tipos de ventanas. Un botón de comando es una ventana. Los iconos, cuadros de texto, botones de opción y barras de menús son todas ventanas.
El sistema operativo Microsoft Windows administra todas estas ventanas asignando a cada una un único número identificador (controlador de ventana o hWnd). El sistema controla continuamente cada una de estas ventanas para ver si existen signos de actividad o eventos. Los eventos pueden producirse mediante acciones del usuario, como hacer clic con el *mouse* (ratón) o presionar una tecla, mediante programación o incluso como resultado de acciones de otras ventanas.
Cada vez que se produce un evento se envía un mensaje al sistema operativo. El sistema procesa el mensaje y lo transmite a las demás ventanas. Entonces, cada ventana puede realizar la acción apropiada, basándose en sus propias instrucciones para tratar ese mensaje en particular (por ejemplo, volverse a dibujar cuando otra ventana la ha dejado al descubierto).
Como puede imaginar, tratar todas las combinaciones posibles de ventanas, eventos y mensajes podría ser interminable. Afortunadamente, Visual Basic le evita tener que tratar con todos los controladores de mensajes de bajo nivel. Muchos de los mensajes los controla automáticamente Visual Basic, mientras que otros se tratan como procedimientos de evento para su comodidad. Esto le permite crear rápidamente eficaces aplicaciones sin tener que tratar detalles innecesarios.
En las aplicaciones tradicionales o “por procedimientos”, la aplicación es la que controla qué partes de código y en qué secuencia se ejecutan. La ejecución comienza con la primera línea de código y continúa con una ruta predefinida a través de la aplicación, llamando a los procedimientos según se necesiten.
En una aplicación controlada por eventos, el código no sigue una ruta predeterminada; ejecuta distintas secciones de código como respuesta a los eventos. Los eventos pueden desencadenarse por acciones del usuario, por mensajes del sistema o de otras aplicaciones, o incluso por la propia aplicación. La secuencia de estos eventos determina la secuencia en la que se ejecuta el código, por lo que la ruta a través del código de la aplicación es diferente cada vez que se ejecuta el programa.
Puesto que no puede predecir la secuencia de los eventos, el código debe establecer ciertos supuestos acerca del “estado del mundo” cuando se ejecute. Cuando haga suposiciones (por ejemplo, que un campo de entrada debe contener un valor antes de ejecutar un procedimiento para procesar ese valor), debe estructurar la aplicación de forma que asegure que esa suposición siempre será válida (por ejemplo, deshabilitando el botón de comando que inicia el procedimiento hasta que el campo de entrada contenga un valor).
El código también puede desencadenar eventos durante la ejecución. Por ejemplo, cambiar mediante programación el texto de un cuadro de texto hace que se produzca el evento *Change* del cuadro de texto. Esto causaría la ejecución del código (si lo hay) contenido en el evento *Change*. Si supone que este evento sólo se desencadenará mediante la interacción del usuario, podría ver resultados inesperados. Por esta razón es importante comprender el modelo controlado por eventos y tenerlo en cuenta cuando diseñe su aplicación.
Con el fin de observar las diferencias entre *programación secuencial* (estilo DOS) y *programación orientada a objetos* (estilo Windows), vamos a construir una aplicación de ejemplo desde los dos estilos comentados.

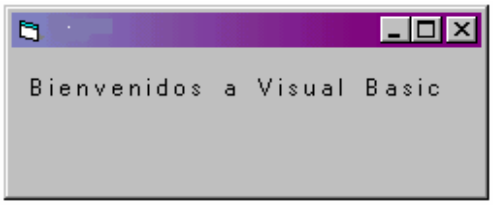
Programación secuencial

Suponiendo que ya ha arrancado Visual Basic, haga doble clic sobre el formulario *Form1* visualizado por defecto y en la ventana que se presenta escriba e1 código que se muestra en la figura siguiente:



Esta ventana presenta un procedimiento privado llamado *Form_Load* que contiene cuatro sentencias. La primera, *Form1.Caption = "UAI"*, pone el título al formulario *Form1*, la segunda *Form1.AutoRedraw = True*, activa el redibujado automático del formulario, la tercera y la cuarta, *Form1.Print ""* y *Form1.Print "Bienvenidos a Visual Basic"*, dibuja el mensaje especificado sobre el formulario.

Para guardar la aplicación, ejecute la opción *Guardar proyecto* del menú *Archivo* y ponga nombre a los archivos cuando le sean solicitados. Por ejemplo, guarde el formulario con el nombre *saludol.frm* y el proyecto con el nombre *saludo1. vbp*.
Para ver el resultado, ejecute la aplicación. Para ello, haga clic en el botón correspondiente de la barra de herramientas o ejecute la opción *Iniciar* del menú *Ejecutar*. El resultado puede verlo en la figura que se presenta a continuación:



Observe que el resultado es una ventana Windows titulada *UAI* que visualiza el mensaje *Bienvenidos a Visual Basic*.
Así mismo, observe que la ventana Windows tiene su menú de control a la izquierda (el icono de la aplicación); sus botones para minimizarla, maximizarla o cerrarla, a la derecha; y que se puede redimensionar, actuando sobre el marco.

Para dotar a la ventana con esta funcionalidad, no hemos tenido que escribir nada de código; es un trabajo que Visual Basic ha hecho por nosotros.
Para finalizar la aplicación, haga clic en el botón correspondiente de la barra de herramientas o ejecute la opción *Terminar* del menú *Ejecutar*. Otra forma de realizar la misma operación es hacer clic en botón, cerrar la ventana ( ).

Programación orientada a objetos

Como hemos dicho anteriormente, una aplicación en Windows presenta todas las opciones posibles en uno o más formularios (ventanas o cajas de diálogo), para que el usuario elija una de ellas. Esto da lugar a una nueva forma de pensar y de programar. Por ejemplo, vamos a realizar una aplicación Windows, *saludo2*, que visualice una ventana como la de la figura siguiente, de forma que cuando el usuario haga clic en el botón *Mensaje*, en la caja de texto aparezca el mensaje *Bienvenidos a Visual Basic* y cuando haga clic en *Borrar* desaparezca dicho mensaje.

Suponiendo que ya tenemos arrancado Visual Basic, ¿cuál es el siguiente paso para desarrollar una aplicación Windows? En general, para construir una aplicación siga los pasos indicados a continuación:

1. Cree una nueva aplicación (nuevo proyecto).
2. Ajuste el tamaño por defecto del formulario.
3. Dibuje los controles.
4. Defina las propiedades del formulario y de los controles.
5. Escriba el código para cada uno de los objetos.
6. Guarde la aplicación.
7. Verifique la aplicación.
8. Cree un fichero ejecutable.