

13

CAPÍTULO

Programación y desarrollo de software

CONTENIDO DEL CAPÍTULO ::

Este capítulo contiene las siguientes lecciones:

Lección 13A:

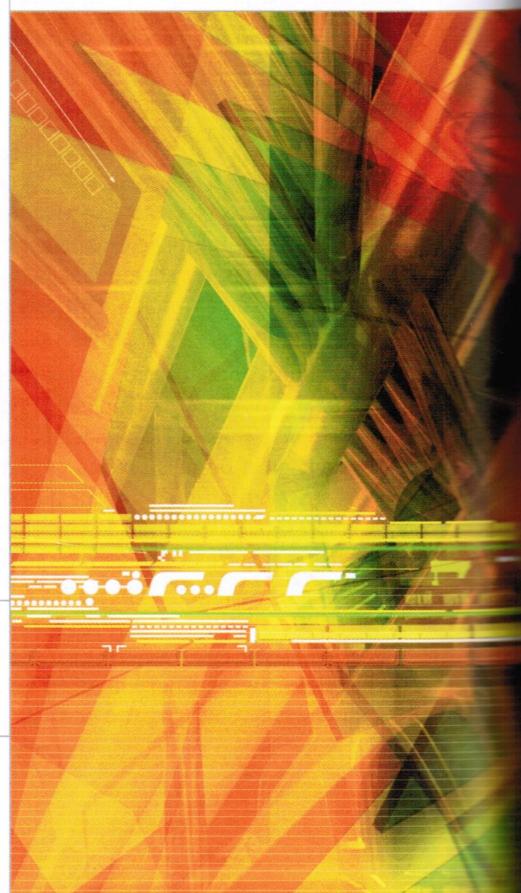
Creación de programas de computación

- » ¿Qué es un programa de cómputo?
- » Interacción hardware/software
- » Planeación de un programa de cómputo
- » La forma en que los programas resuelven problemas
- » Programación estructurada y orientada a objetos

Lección 13B:

Lenguajes de programación y el proceso de programación

- » La evolución de los lenguajes de programación
- » Lenguajes de desarrollo de la World Wide Web
- » El ciclo de vida del desarrollo de sistemas en la programación



Panorama general: ¿qué es un programa de cómputo?

El software, como aprendió en el capítulo 1, proporciona las instrucciones que hacen funcionar al hardware de la computadora; sin éste la computadora no puede funcionar, no es más que un montón de partes. Algunos comandos de software pueden estar integrados en piezas de hardware específicas (por ejemplo, los chips de CPU y ROM), pero incluso en esos casos, la programación es simplemente una versión de “codificación dura” del software. Un ejemplo de este tipo de codificación se puede encontrar en algo tan común como un reloj digital. Aunque posiblemente nunca lo haya considerado, cierta clase de software está integrada en los circuitos del reloj los cuales le permiten funcionar. Debido a que el software es una parte tan importante de cualquier sistema de computación, es importante entender qué tipo de software es y de dónde proviene.

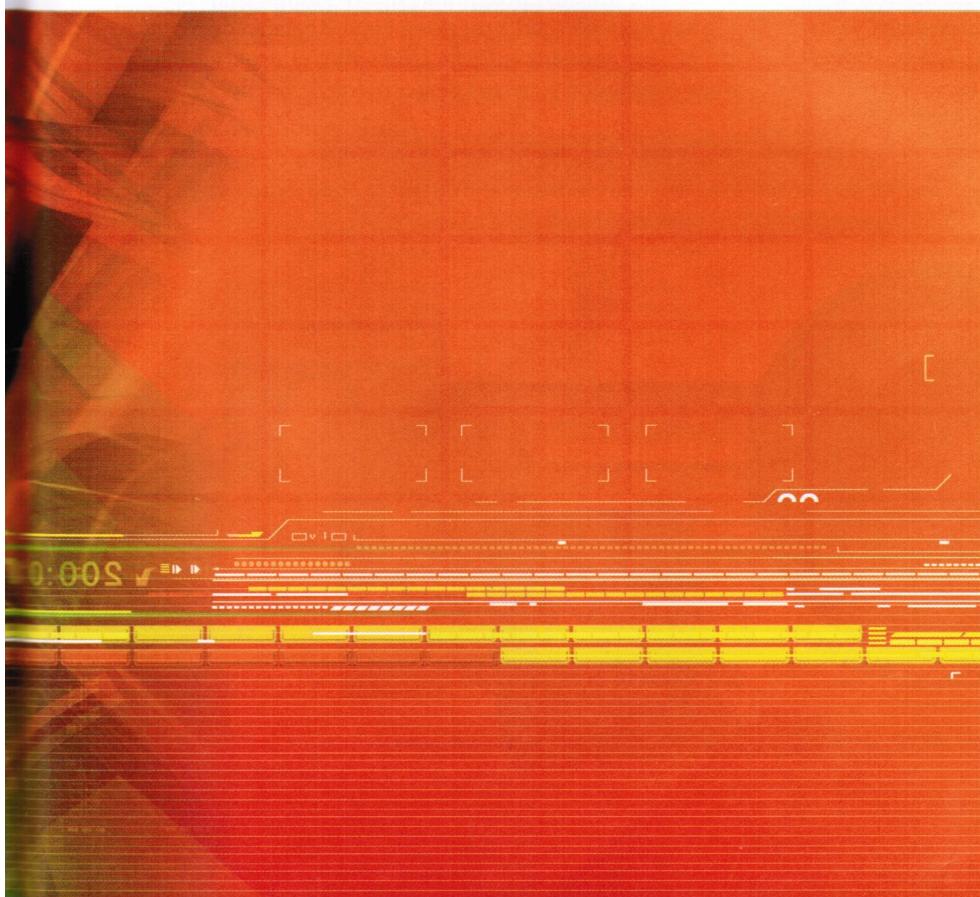
El término *software* puede utilizarse en forma genérica, por ejemplo en la frase “El software le dice a la computadora lo que debe hacer”. El término también puede utilizarse de una manera más específica para describir un sistema operativo o aplicación. Por ejemplo, puede decir, “Windows XP es un producto de software”, o “Photoshop es un producto de software”. En cualquier computadora, el software normalmente consiste en el sistema operativo, distintas herramientas (algunas veces se consideran parte del sistema operativo) y aplicaciones.

Las secciones siguientes explican qué son los programas y cómo funcionan. Describen algunos de los procesos y herramientas que utilizan los desarrolladores de software cuando crean programas de cómputo.

Creación de programas de computación

OBJETIVOS ::

- » Definir el término *programa de computadora*.
- » Describir el uso de los diagramas de flujo y el pseudocódigo en la programación.
- » Identificar dos maneras en las cuales un programa puede trabajar alrededor de una solución.
- » Diferenciar los dos principales enfoques hacia la programación de computadoras.
- » Listar y describir tres elementos de la programación orientada a objetos.



¿Qué es un programa de cómputo?

Un programa de cómputo es un conjunto de instrucciones o declaraciones (también conocidas como *código*) que debe realizar el CPU de una computadora. Los programas, o software, tienen distintas formas. Éstas se pueden dividir en tres categorías principales: sistemas operativos, herramientas y aplicaciones (véase la figura 13A.1).

Un programa normalmente está compuesto de un módulo principal y submódulos. Estos módulos están almacenados como un conjunto de archivos; los programas grandes pueden contener miles de archivos individuales, cada uno para un propósito específico. Algunos de los archivos contienen instrucciones para la computadora, mientras que otros archivos contienen datos. Para las PC basadas en Windows, algunas extensiones comunes para los archivos de programas son las siguientes:

- » **Archivos ejecutables.** Un **archivo ejecutable (.exe, .com)** es parte de un programa que en realidad envía comandos al procesador. De hecho, cuando usted ejecuta un programa, está ejecutando el archivo ejecutable. El procesador ejecuta dos comandos del archivo, de ahí proviene el nombre *archivo ejecutable*. Los archivos ejecutables normalmente (pero no siempre) tienen la extensión de nombre de archivo .exe.
- » **Archivos de biblioteca dinámica de vínculos.** Un **archivo de biblioteca dinámica de vínculos (.dll)** es un archivo .exe parcial. Un archivo .dll no puede ejecutarse por sí mismo; en lugar de esto, sus comandos son accedidos por otro programa que está ejecutándose. Debido a que los archivos .dll pueden contener partes de un programa ejecutable, ofrecen a los programadores una manera efectiva de dividir programas grandes en componentes pequeños que son reemplazables. Esta característica hace que el programa completo sea más fácil de actualizar. Además, los archivos .dll también pueden ser compartidos por distintos programas al mismo tiempo.
- » **Archivos de inicialización.** Un **archivo de inicialización (.ini)** contiene información sobre configuraciones, por ejemplo, el tamaño y punto de inicio de una ventana, el color del fondo, el nombre del usuario y otros aspectos. Los archivos de inicialización ayudan a los programas a ejecutarse o contienen información que los programas pueden utilizar cuando se ejecutan. Aunque los archivos de inicialización se siguen utilizando, muchos programas nuevos almacenan las preferencias del usuario y otras variables del programa en el Registro de Windows, una base de datos especial que contiene información sobre el usuario de la computadora, los programas instalados y ciertos dispositivos de hardware.
- » **Archivos de ayuda.** Un **archivo de ayuda (.hlp, .chm)** contiene información en un formato indexado y con vínculos cruzados. Al incluir un archivo de ayuda, los programadores pueden proporcionar al usuario información de ayuda en línea.



Norton EN LÍNEA

Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre archivos ejecutables, de biblioteca dinámica de vínculos, de inicialización y de ayuda.

FIGURA 13A.1

Las principales categorías de software incluyen sistemas operativos, utilerías y aplicaciones.



- » **Archivos de secuencia de comandos.** Un **archivo de secuencia de comandos** (**.bat**) automatiza tareas comunes o repetitivas. Un archivo de secuencias de comandos es un programa simple que consiste en un archivo de texto sin formato que contiene uno o más comandos del sistema operativo. Si escribe el nombre de un archivo de secuencia de comandos en un símbolo del sistema, su sistema operativo ejecutará los comandos del sistema operativo.

Por omisión, la mayor parte de los archivos de programas se almacenan en una carpeta nombrada como la aplicación o una abreviación de ella. Sin embargo, algunos archivos de programa se pueden encontrar en otras carpetas. Por ejemplo, los archivos .dll en Windows XP normalmente están almacenados en la carpeta c:\windows\system 32. Para ver una lista de la mayoría de los archivos necesarios para ejecutar una aplicación, puede abrir la carpeta de esa aplicación. La figura 13A.2 muestra un directorio común de software, en este caso, para un programa hipotético llamado CoolToys versión 1.3.

Interacción hardware/software

El software es la razón por la cual las personas compran computadoras. Las instrucciones de un programa se ejecutan en el nivel del hardware principalmente en el CPU. Por ejemplo, el programa puede decirle al CPU que recupere una parte específica de información de la memoria. Si el programa le dice al CPU que reproduzca un archivo de sonido, entonces el programa genera una **interrupción**. Una interrupción es una señal hacia el CPU con el fin de que ejecute una serie de pasos previamente programados. En este caso el hardware enviaría el archivo de sonido al dispositivo de salida de audio.

Un individuo que tiene una capacitación muy exhaustiva, conocido como programador de computadoras, crea esta lista de instrucciones. Esta lista con frecuencia se conoce como código y el proceso de escritura de la lista normalmente se conoce como codificación.

Código

El término **código** se refiere a las declaraciones escritas en cualquier lenguaje de programación, por ejemplo en el código máquina o en el de alto nivel. Como observó en el capítulo 5 las computadoras piensan y hablan con el sistema binario de numeración. Desde luego, el sistema binario es demasiado críptico para que los humanos se puedan relacionar con él durante mucho tiempo. Ésta es la razón por la cual se crearon los lenguajes de programación de computadoras: para simplificar el proceso de escritura de instrucciones que las computadoras puedan utilizar.

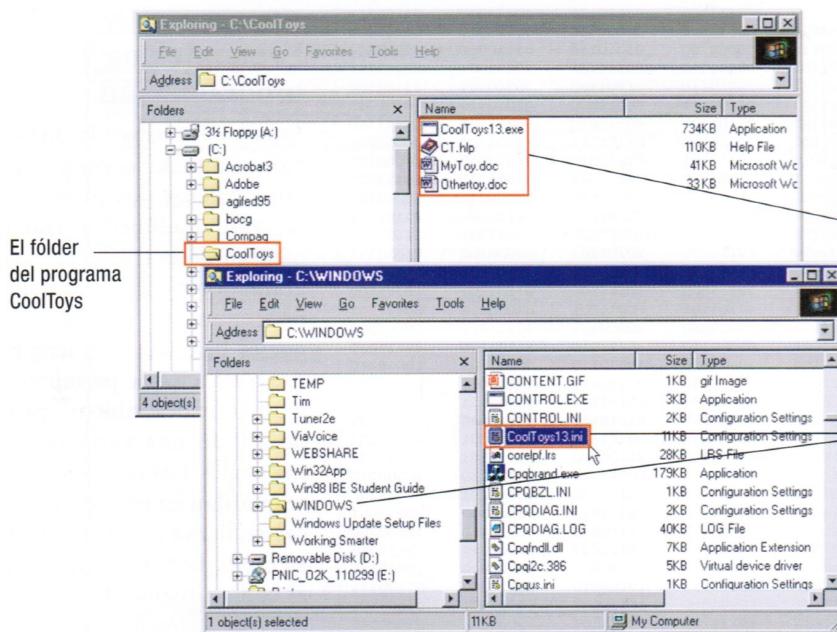


FIGURA 13A.2

Las aplicaciones suelen tener archivos ejecutables, de ayuda y de inicialización.

```

13a18 - Microsoft Visual C++ - [13a18.cpp]
File Edit View Insert Project Build Tools Window Help
(Globals) (All global members) main
13a18 classes
13a18.cpp
float average;
getAges(age1, age2, age3);
average = averageAges(age1, age2, age3);
cout << "The average age is " << average << endl;
}
}

Compiling...
13a18.cpp
D:\My Documents\books\norton intro\Unit 13 original\13a18.cpp(10) : error C2065:
D:\My Documents\books\norton intro\Unit 13 original\13a18.cpp(10) : error C2146:
D:\My Documents\books\norton intro\Unit 13 original\13a18.cpp(10) : error C2146:
D:\My Documents\books\norton intro\Unit 13 original\13a18.cpp(10) : error C2001:
D:\My Documents\books\norton intro\Unit 13 original\13a18.cpp(10) : error C2065:
D:\My Documents\books\norton intro\Unit 13 original\13a18.cpp(10) : error C2146:
D:\My Documents\books\norton intro\Unit 13 original\13a18.cpp(10) : error C2065:
D:\My Documents\books\norton intro\Unit 13 original\13a18.cpp(10) : error C2143:
D:\My Documents\books\norton intro\Unit 13 original\13a18.cpp(11) : error C2143:
Error executing cl.exe.

Build Debug Find In File 1 Find In File 2 Recent Ln 10, Col 1 REC COL DMR READ

```

FIGURA 13A.3

Las comillas faltantes al inicio de "The" es un error de sintaxis común que causa muchos otros errores ficticios.

La programación es tediosa pero también emocionante. Es tediosa debido a que todos los lenguajes de programación, al igual que los idiomas hablados, tienen un conjunto de reglas obligatorias. Las personas que hablan inglés pueden cometer varios errores gramaticales y de todas maneras poder comunicarse con la persona que las escucha. Sin embargo, un fragmento de código debe ser perfecto antes de que se pueda ejecutar. No se permiten errores gramaticales o de sintaxis en la programación. El programador debe corregir estos errores antes de probar el programa. La figura 13A.3 muestra un error de sintaxis en un programa C++.

La programación es emocionante en varios niveles. Primero y principalmente, la escritura de código le proporciona al programador la oportunidad de crear algo nuevo. El desarrollador logra ejercitarse sus músculos creativos. Segundo, la emoción proviene del desafío de resolver un problema. El problema puede ser tan simple como calcular un valor o tan complejo como determinar la ruta de un satélite en órbita. La posibilidad de resolver problemas, aunque sean menores, mediante un fragmento de código es un reto difícil de resistir para los programadores.

Norton EN LÍNEA

Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre el código máquina.

FIGURA 13A.4

Un ejemplo de código máquina.

Código máquina

Como aprendió en el capítulo 1, la memoria e interruptores de procesamiento de una computadora utilizan el sistema binario de numeración, el cual consiste en unos y ceros. Cualquier comando de software que afecte directamente al hardware debe estar escrito en el sistema binario de numeración. Debido a que estos unos y ceros forman el lenguaje del hardware de computadoras, este código se conoce normalmente como **código máquina** o **lenguaje máquina**. El lenguaje máquina consiste sólo en unos y ceros. Aunque este formato carezca de sentido para usted, lo tiene para la computadora y puede concebirse como el lenguaje de computadora de nivel más bajo. La figura 13A.4 muestra un ejemplo de código máquina.

Lenguajes de programación

Aunque el código de una computadora puede consistir sólo en unos y ceros, los programadores de computadoras no trabajan ni piensan de esa manera. Los programadores utilizan **lenguajes de programación** en lugar de lenguajes binarios. Los lenguajes de programación le permiten al programador describir un programa utilizando una variante del idioma inglés básico. Los resultados se guardan en un archivo y entonces se llaman **código fuente**. Por ejemplo, la figura 13A.5 muestra el código fuente de Visual Basic.NET de un programa.

Norton EN LÍNEA

Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre lenguajes de programación.

simnet™

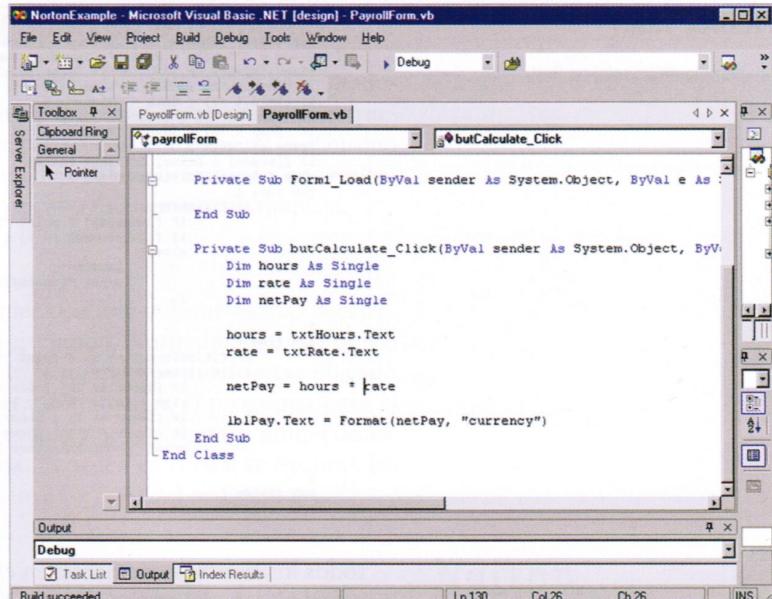
| | | | | |
|----------|----------|----------|-----------|-----------|
| 11010010 | 10101010 | 10100010 | 01100111 | 10100010 |
| 01000110 | 10111000 | 10111100 | 01101101 | 10111100 |
| 10100010 | 10101101 | 00011001 | 01010110 | 00011001 |
| 10111100 | 11010010 | 10000100 | 10111101 | 10000010 |
| 00011001 | 01000110 | 10111010 | 11101100 | 10111010 |
| 10000100 | 10100010 | 10010110 | 00011000 | 10010110 |
| 10111010 | 10111100 | 01100111 | 10111010 | 01100111 |
| 10010110 | 00011001 | 01010101 | 01010111 | 01010101 |
| 01100111 | 10000100 | 01010110 | 01100010 | 01010110 |
| 01101011 | 10111010 | 10111101 | 10011001 | 10111101 |
| 01010110 | 10010110 | 11101100 | 00101011 | 11101100 |
| 10111101 | 01100111 | 00011000 | 01010110 | 10010001 |
| 11010100 | 01010110 | 01010111 | 11010100 | 00011001 |
| 00011000 | 01010110 | 01010111 | 11101100 | 00011001 |
| 10111010 | 10111100 | 01010101 | 01000010 | 10101010 |
| 01101011 | 11010010 | 10010001 | 01000010 | 10111010 |
| 01101010 | 11010110 | 10100010 | 01100111 | 10100010 |
| 01100010 | 11010110 | 00101010 | 10100010 | 10010110 |
| 01100111 | 10000100 | 01010110 | 01100010 | 10010110 |
| 01101100 | 10111010 | 10111101 | 10011001 | 10111100 |
| 00011001 | 01000110 | 10111010 | 11101100 | 10111010 |
| 10000100 | 10100010 | 10010110 | 00011000 | 10010110 |
| 10111010 | 10111100 | 01010101 | 01000010 | 10100010 |
| 11010100 | 01010110 | 01010111 | 10111101 | 10111100 |
| 01100111 | 10000100 | 01010110 | 01100010 | 10010110 |
| 01101101 | 10111010 | 10111101 | 10011001 | 10111101 |
| 01010101 | 10010110 | 11101100 | 00110101 | 11101100 |
| 10111101 | 01100111 | 00011000 | 01010110 | 10100010 |
| 11101000 | 01010110 | 10111010 | 10111101 | 10111100 |
| 00011000 | 01010110 | 01101011 | 11101100 | 00011001 |
| 10111010 | 10111101 | 01110010 | 11010010 | 100000100 |
| 01101011 | 11010100 | 10011001 | 01000010 | 10111010 |
| 01100101 | 11010010 | 00110101 | 10100010 | 10010110 |
| 10011001 | 01000010 | 10101010 | 10111100 | 01100111 |
| 00110101 | 10100000 | 10111000 | 00011001 | 01101101 |
| 10101010 | 10111100 | 10101101 | 100000100 | 01010110 |

Aprenderá más sobre lenguajes de programación específicos más adelante en este capítulo.

Compiladores e intérpretes

Después de crear un fragmento del código fuente, el programador debe convertirlo en código máquina (en una serie de unos y ceros) antes de que pueda ejecutarse en una computadora. El trabajo de convertir el código fuente lo hace uno de dos tipos de programas:

- » Un **compilador** convierte todo el código fuente en código máquina y crea un archivo ejecutable. El resultado del compilador se conoce como **código objeto**. En algunos lenguajes, el código objeto debe estar vinculado para producir un verdadero archivo ejecutable. En otros lenguajes, el código objeto directamente ejecutable. El programador puede copiar el código objeto ejecutable en cualquier sistema similar y ejecutar el programa. En otras palabras, una vez que está compilado el programa se convierte en un archivo ejecutable independiente que no necesita del compilador para ejecutarse. Desde luego, cada lenguaje de programación necesita un compilador propio para traducir el código escrito en ese lenguaje. Por ejemplo, el lenguaje de programación C++ requiere de un compilador C++, mientras que Pascal necesita un compilador Pascal. La figura 13A.6 muestra un ejemplo de compilador.
- » Un **intérprete** también convierte el código fuente en código máquina. Sin embargo, en lugar de crear un archivo de código de objeto ejecutable, lo traduce y luego ejecuta cada línea del programa, una a la vez. Los intérpretes traducen el código sobre la marcha, por lo que tienen cierta flexibilidad que no tienen los compiladores. El código interpretado se ejecuta de manera más lenta que el código compilado debido a que se tiene que interpretar cada vez que se ejecuta y una copia del intérprete debe acompañar al código durante todo el tiempo. Por tanto,



The screenshot shows the Microsoft Visual Basic .NET IDE interface. The main window displays the code for 'PayrollForm.vb [Design]'. The code includes event handlers for the 'Form1_Load' and 'butCalculate_Click' events. The 'butCalculate_Click' handler calculates net pay based on hours and rate input from text boxes and displays the result in a label. The code uses standard VB syntax with declarations for 'Dim' and assignments for variables like 'hours' and 'rate'.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As EventArgs)
    ' ...
End Sub

Private Sub butCalculate_Click(ByVal sender As System.Object, ByVal e As EventArgs)
    Dim hours As Single
    Dim rate As Single
    Dim netPay As Single

    hours = txtHours.Text
    rate = txtRate.Text

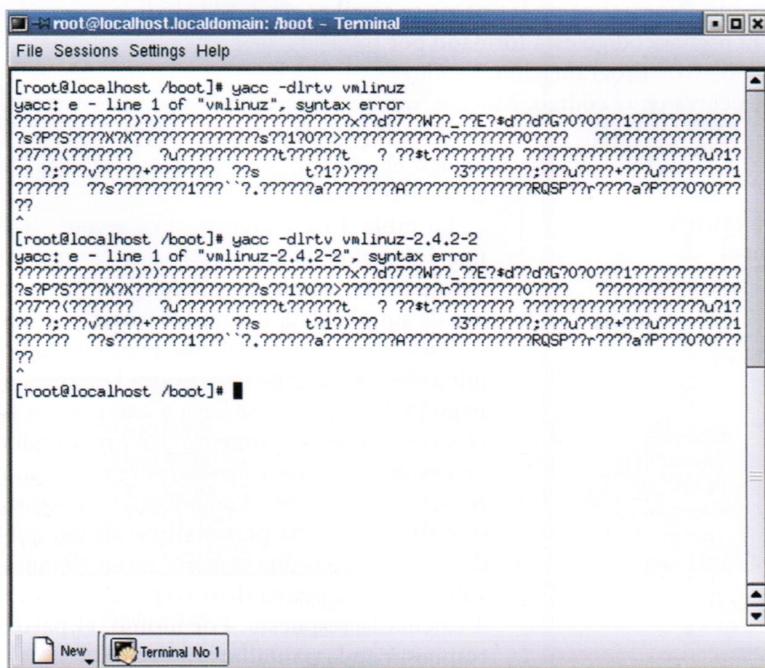
    netPay = hours * rate

    lblPay.Text = Format(netPay, "currency")
End Sub

End Class
```

FIGURA 13A.5

Un ejemplo de código fuente de una aplicación creada en Visual Basic.NET.



The screenshot shows a terminal window on a Linux system with the title 'root@localhost.localdomain: /boot - Terminal'. The user is running the 'yacc' command with the '-dLrtv' option on the 'vmlinuz' file. The output shows two separate yacc runs, both resulting in syntax errors. The errors are identical, indicating a problem with line 1 of the grammar file. The errors include numerous question marks and exclamation marks, typical of a parser error message.

```
[root@localhost /boot]# yacc -dLrtv vmlinuz
yacc: e - line 1 of "vmlinuz", syntax error
?????????????x??d????W??_??E?*d??d?G?0?0?0?1?????????????
?s?P?S?????K?????????????s??1?0?2?????????????r?????????0?????_
?????????????_u?????????l????t ? ??*t?????????;?????????????u?1?
?? ?;????v????+???????? ??s t?1)???? ?3????????;??u????+??u?????1
?????? ??s????????1???.??????a?????????A????????????RQSP??r?????a?P?????0?0??
??
^
[root@localhost /boot]# yacc -dLrtv vmlinuz-2.4.2-2
yacc: e - line 1 of "vmlinuz-2.4.2-2", syntax error
?????????????x??d????W??_??E?*d??d?G?0?0?0?1?????????????
?s?P?S?????K?????????????s??1?0?2?????????????r?????????0?????_
?????????????_u?????????l????t ? ??*t?????????;?????????????u?1?
?? ?;????v????+???????? ??s t?1)???? ?3????????;??u????+??u?????1
?????? ??s????????1???.??????a?????????A????????????RQSP??r?????a?P?????0?0??
??
^
[root@localhost /boot]#
```

FIGURA 13A.6

Yet Another C Compiler (YACC), que se encuentra en muchas versiones de Linux.

FIGURA 13A.7

Un ejemplo de un reloj en Perl.

```
sub divide {@months = ('January','February','March','April','May','June','July','August','Se
@days = ('Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday');
($sec,$min,$hour,$day,$month,$year,$day2) = (localtime(time))[0,1,2,3,4,5,6];
if ($sec < 10) { $sec = "0$sec"; }
if ($min < 10) { $min = "0$min"; }
if ($hour < 10) { $hour = "0$hour"; }
if ($day < 10) { $day = "0$day"; }
$year += "1900";
if ($order) { &time; &divide; &date; }
else { &date; &divide; &time; }
sub date {
    if ($dateformat) {
        if ($weekday) { print "$days[$day2], "; }
        if ($dateformat == 1) { print "$months[$month] $day, $year"; }
        else {
            $month++;
            print "$month/$day/$year";
        }
    }
}
sub time {
    if ($timeformat == 2) { print "$hour:$min:$sec"; }
    elsif ($timeformat == 1) {
        if ($hour >= 12) { $ex = "P.M.:"; }
        else { $ex = "A.M.:"; }
        if ($hour == 0) { $hour = 12; }
        if ($hour > 12) { $hour -= 12; }
        print "$hour:$min:$sec $ex";
    }
}
sub divide {
```

todos los sistemas que necesiten el programa deben contar con una copia del intérprete además del código fuente. Algunos lenguajes de intérprete son LISP, BASIC y Visual Basic. La figura 13A.7 ilustra un programa de reloj escrito en Perl.

Planeación de un programa de cómputo



Norton
EN LÍNEA

Visite el sitio <http://www.mhhe.com/peternorton>.

mhhe.com/peternorton para obtener más información sobre la planeación de un programa de cómputo.

La escritura de programas puede ser extremadamente difícil. Sin un plan, el programador escribirá un programa defectuoso o completamente inútil para realizar una tarea. Cuando se planea, el programador tendrá una idea de lo que debe hacer y sabrá por dónde comenzar. Dos herramientas de planeación que utilizan los programadores con frecuencia son los **diagramas de entrada-proceso-salida** (IPO, por sus siglas en inglés) y **pseudocódigo**.

El diagrama IPO ayuda al programador a determinar lo que es necesario para escribir el programa. Consiste en tres columnas. En la primera columna el programador lista qué datos son necesarios para resolver la tarea. En la última columna, el programador lista los resultados deseados. La columna de en medio es la parte difícil. Aquí el programador lista los pasos que son necesarios para obtener el resultado deseado. Normalmente los pasos se escriben en pseudocódigo.

El pseudocódigo está formado por frases del lenguaje natural que tienen apariencia de código de programación. La idea es escribir en el idioma hablado lo que se necesita que ocurra en el código. Muchas veces el programador no sabe perfectamente lo que debe escribir en cada paso del programa. Al describir el código el programador al menos tiene un inicio y puede comenzar a pensar sobre cómo implementar el código.

La tabla 13A.1 muestra una gráfica IPO que detalla una aplicación de nómina. El programa calculará el pago neto para un empleado que trabaja por horas. Por razones de simplicidad, ignora los pagos por horas extras.

Para resolver el problema, el programador debe determinar la cantidad de horas que el empleado trabajó y su pago por hora. Esto se lleva a cabo en los pasos 1 y 2 de la sección de procesamiento. Un programador inteligente se asegurará de que estos datos tengan sentido. El paso 3 realiza esta revisión. Las horas válidas pueden ser entre 0 y 40 horas. Una persona que afirme que trabajó más de 100 horas en una semana es un ejemplo de datos no válidos. El programa debe responder a este error. El paso 4 calcula la respuesta. Por último, el paso 5 imprime la respuesta en la pantalla.

TABLA 13A.1

El diagrama IPO de un programa que calcula el pago neto de un empleado por horas

| Entrada | Proceso | Salida |
|------------------|---|-----------|
| Horas trabajadas | Introducir horas trabajadas | Pago neto |
| Salario por hora | Introducir salario por hora Validar datos $\text{Pago} = \text{horas trabajadas}^*$ salario por hora | |
| | Desplegar pago neto | |

La forma en que los programas resuelven problemas

Ahora ya sabe que un programa es un conjunto de pasos que controlan una computadora, pero es probable que no sepa qué apariencia tienen estos pasos. Su apariencia o estructura depende un poco del lenguaje de programación, pero el concepto general es el mismo sin importar qué lenguaje se utiliza. Cada paso del código es una instrucción que realiza una sola tarea en una secuencia de pasos que llevan a cabo una tarea más compleja.

Control de flujo de un programa

Cuando inicia un programa, la computadora comienza a leer y llevar a cabo declaraciones en el punto de entrada del principal archivo ejecutado. Normalmente, este punto de entrada es la primera línea (o declaración) del archivo, aunque puede estar ubicado en otro lugar. Después de la ejecución de la primera declaración, el programa pasa el control (o flujo) a otra declaración y así en adelante hasta que se ejecuta la última declaración del programa; entonces el programa termina. El orden en el cual se ejecutan las declaraciones de un programa se conoce como **control de flujo del programa**.

Algoritmos

Los pasos que se representan en un diagrama IPO normalmente conducen hacia un resultado deseado. En conjunto, estos pasos se conocen como **algoritmo**. Un algoritmo es una serie de instrucciones paso a paso que, cuando se siguen, producen un resultado conocido o esperado. Los pasos para encontrar una solución son los mismos sin importar que se encuentre la solución con una computadora o a mano, por lo cual se puede tener un programa y un diagrama IPO hecho a mano para realizar la misma tarea.

La figura 13A.8 muestra un algoritmo de **diagrama de flujo** para encontrar el punto más alto de un mapa topográfico. Los diagramas de flujo son una forma muy estructurada para diseñar programas. El algoritmo simplemente crea muestras de cada

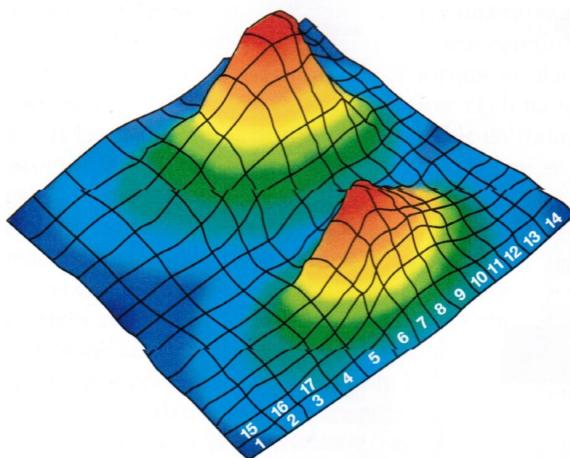
Norton
EN LÍNEA



Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre algoritmos.

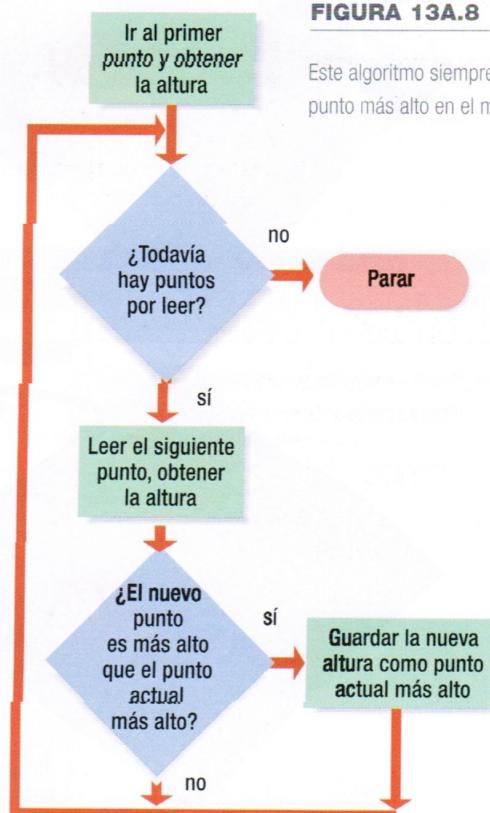
FIGURA 13A.8

Este algoritmo siempre encuentra el punto más alto en el mapa.



| | | |
|----------|---------------|-----------|
| $1 = 2'$ | $115 = 5000'$ | Terminado |
| $2 = 2'$ | $116 = 5100'$ | |
| $3 = 3'$ | $117 = 5200'$ | |
| $4 = 2'$ | $118 = 5100'$ | |

Punto actual Punto actual Punto actual
más alto = 3 a 3' más alto = 117 a 5200'
Punto actual más alto = 117 a 5200'



cuadro en la cuadrícula (un cuadro por vez) y compara los resultados con el punto más alto actual. El proceso es lento pero siempre encuentra el punto más alto.

Los algoritmos pueden tener muchos otros usos. Un programa de hoja de cálculo, por ejemplo, puede contener un algoritmo que despliegue la suma de las celdas que marque el usuario. Este tipo de algoritmo permitiría que el usuario marcase celdas, leyera los números de esas celdas, calculara el total de los números y los desplegará en la pantalla. Otro algoritmo de una hoja de cálculo podría buscar la palabra más larga en una columna y entonces ajustar el ancho de la columna para que se ajuste al texto.



Norton EN LÍNEA

Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre heurística.

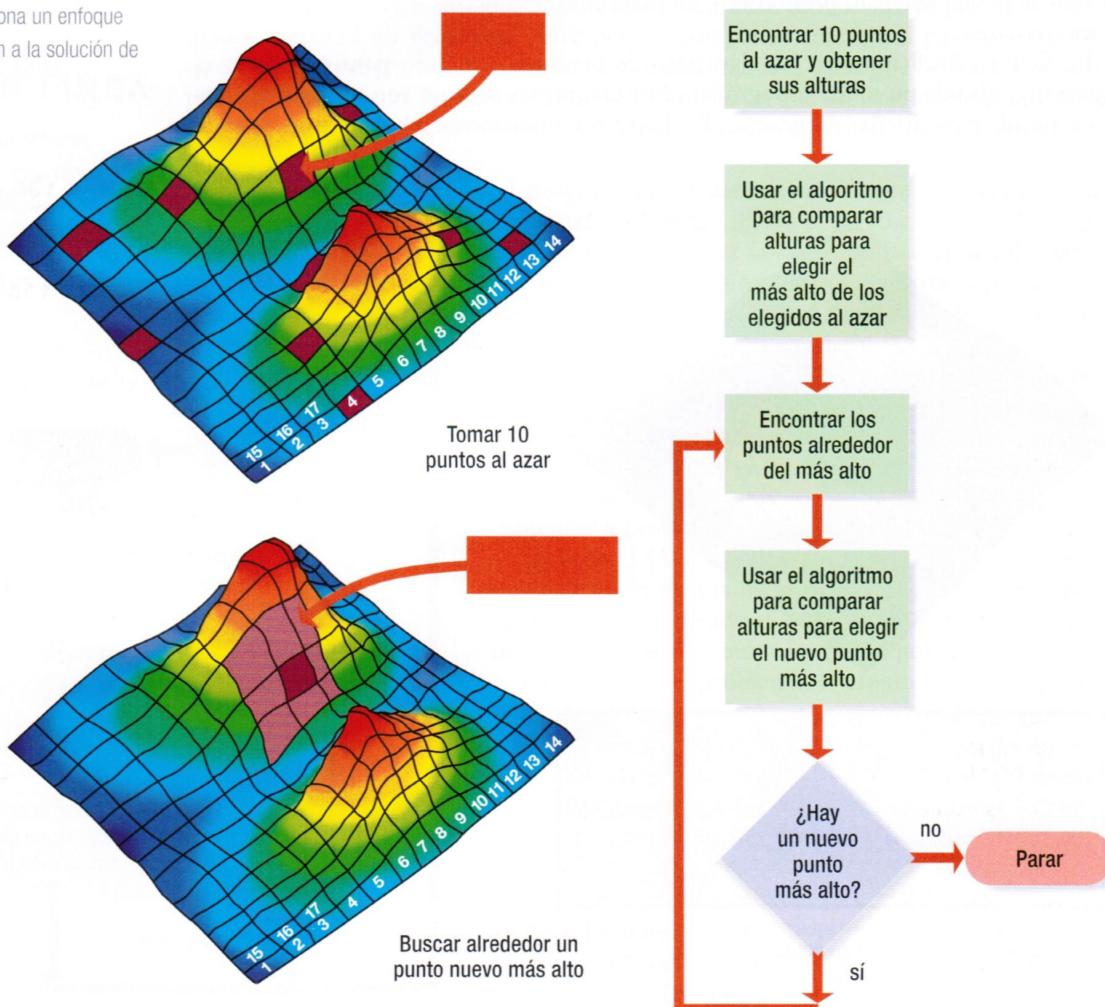
Heurística

Algunas veces no existe un algoritmo para resolver cierto problema, o el algoritmo es tan complejo o lento que no puede codificarse o ejecutarse. En estos casos, los programadores utilizan la heurística para ayudarse a resolver problemas o realizar tareas. La **heurística** es parecida a los algoritmos; es un conjunto de pasos para encontrar la solución de un problema. Pero a diferencia de un algoritmo, una solución heurística no proporciona una garantía de encontrar la mejor solución posible. La heurística ofrece probabilidades de encontrar una solución, aunque no necesariamente la mejor.

La figura 13A.9 muestra una solución heurística para encontrar el punto más alto en un mapa topográfico. La heurística trabaja bajo la suposición de que las montañas se construyen con colinas y que el pico no puede estar en un cuadro por sí mismo en

FIGURA 13A.9

La heurística proporciona un enfoque de mejor aproximación a la solución de problemas.



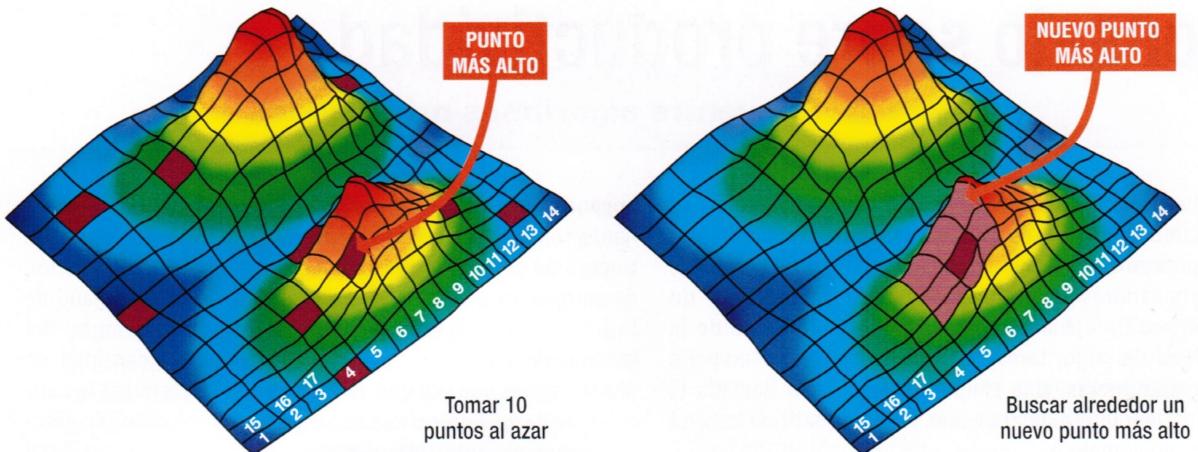


FIGURA 13A.10

La heurística no siempre garantiza encontrar la mejor respuesta posible.

medio del valle. Siguiendo esta heurística, el programa crea muestras de cien puntos aleatorios en el mapa y los compara para ver cuál es el más alto.

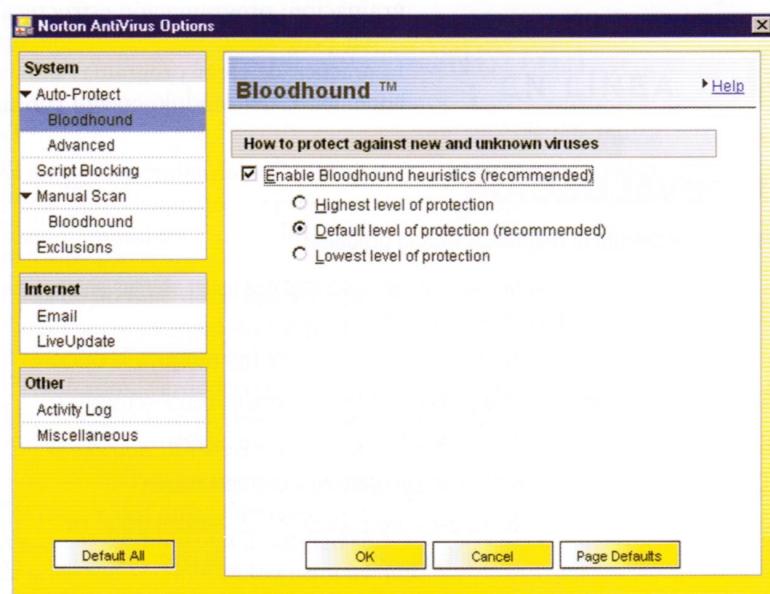
La heurística asume que, debido a que se tiene que subir para alcanzar el punto más alto, el punto más alto actual debe conducir hasta la cima. Para encontrar el siguiente paso hacia la cima, el programa crea muestras de unos cuantos puntos que están cerca del punto más alto actual y entonces los compara. Una vez que el programa ha encontrado un nuevo punto más alto, repite el proceso de tomar muestras alrededor del punto más alto actual y busca uno nuevo. De esta forma, la heurística sube la montaña hasta la cima. Como puede ver, la heurística siempre va hacia arriba. Es posible que, basándose en los puntos de inicio originales, la heurística suba a la montaña más baja. Aunque la heurística produce una solución, es probable que no siempre proporcione la mejor. La figura 13A.10 muestra la heurística llegando a la cima incorrecta.

Es menos probable que la heurística aparezca en las aplicaciones comunes (por ejemplo, una hoja de cálculo o procesador de texto) debido a que estas aplicaciones muy pocas veces requieren que se realice una tarea para la cual no existe una solución correcta. La heurística es extremadamente común en los programas más avanzados que dan seguimiento a cantidades enormes de datos en forma compleja. Por ejemplo, el software que ayuda a pronosticar el clima no tiene la posibilidad de procesar todas las piezas de información disponible en todas las formas posibles. Incluso si el programa pudiera procesar todos los datos, no se esperaría que predijera de forma exacta algo tan complejo como el clima. Sin embargo, utilizando la heurística, los programas pueden seleccionar datos relacionados con el clima para encontrar información útil y después crear una suposición razonablemente exacta del clima basándose en esos datos.

Algunos programas utilizan tanto la heurística como los algoritmos. El software antivirus utiliza algoritmos para encontrar virus muy conocidos como MS Blaster y CodeRed. Los virus nuevos se encuentran utilizando heurística que busca patrones de comportamiento parecidos a los de los virus. Entre éstos se incluyen los archivos que intentan escribir en partes del disco duro o que abren Outlook. Los virus más dañinos son aquellos que conocen cómo trabaja la heurística y la evitan. La figura 13A.11 muestra la pantalla de configuración de la heurística Bloodhound Heuristics de Norton AntiVirus.

FIGURA 13A.11

Norton Antivirus incluye heurística para proteger su máquina de nuevos virus.



Consejo sobre productividad

Encuentre algoritmos más rápidos

En un principio, el estudio de la complejidad de un algoritmo puede parecer que no es importante, especialmente por las computadoras rápidas de la actualidad. No obstante, los programadores y matemáticos han desarrollado un campo entero de estudio dedicado al entendimiento de la complejidad del algoritmo y sus efectos en el desempeño de las computadoras. Una simple demostración llamada El Vendedor Itinerante muestra cómo se puede salir de control un algoritmo complejo.

En este problema, un vendedor que desea visitar todas las ciudades de su ruta siguiendo la ruta más corta posible. Cuando sólo se tienen unas cuantas ciudades, el problema parece ser muy sencillo; existen algunas combinaciones posibles y el vendedor puede comparar las distancias fácilmente. Sin embargo, si se añade otra ciudad, la cantidad de pasos para comparar todas las rutas distintas posibles crece exponencialmente. Después de que se añaden algunas ciudades, la cantidad de comparaciones hace que el problema consuma demasiado tiempo para intentar resolverlo.

¿Cómo se determina la complejidad?

Determinar la complejidad es relativamente simple en teoría, aunque se requiere de habilidades matemáticas para

encontrar los números exactos. En general, puede determinar la complejidad de un algoritmo creando una ecuación de tiempo de ejecución que sea una función de la información de entrada en el algoritmo. La ecuación revisa el tamaño de la información de entrada del algoritmo (por ejemplo, del tamaño de una lista que será ordenada o la cantidad de dígitos en un número que será dividido) y determina la cantidad de pasos necesario para llegar a una solución. En efecto, estará creando una ejecución en tiempo real artificial basándose en el tamaño de la información de entrada. En la vida real, es probable que el algoritmo no tenga que realizar siempre todos los pasos, ya que la ecuación representa una información de entrada del peor de los casos. Una vez que consiga la ecuación del tiempo de ejecución, puede probar varios tamaños de información de entrada y ver la manera en que aumenta el tiempo de ejecución. Si éste crece demasiado rápido en comparación con el tamaño de la entrada, entonces tiene un problema.

Categorías de algoritmos

Para discutir la complejidad de los algoritmos, los matemáticos dividen los algoritmos en las categorías de tratables o intratables:

Programación estructurada y orientada a objetos

Cuando un programador escribe un programa, primero crea un IPO del algoritmo o heurística que utilizará. Una vez que está listo, debe utilizar un lenguaje de programación para crear el código que producirá el resultado esperado. Para crear el código fuente de un programa, los programadores suelen seguir uno de dos métodos de programación: programación estructurada u orientada a objetos.

La **programación estructurada** surgió en los años setenta. El nombre se refiere a la práctica de crear programas utilizando módulos pequeños que son fáciles de leer y entender. Cada módulo cuenta con una sola entrada y salida y realiza una sola tarea.

Las prácticas de la programación estructurada se pueden utilizar con cualquier lenguaje de programación. Uno de los objetivos de la programación estructurada es crear código fácil de leer. Las técnicas de codificación anteriores daban como resultado el “código espagueti” que era prácticamente imposible de leer, entender y mantener. Los desarrolladores de software han descubierto que la programación estructurada da como resultado una mayor eficiencia pero siguen luchando con el proceso de crear software de una manera rápida y correcta.

El reciclaje es considerado como la mejor solución a estos problemas de desarrollo de software. El reciclaje de código permite que los programas se creen de una forma rápida y correcta. En los años ochenta, la computación dio un gran paso hacia

AUTOEVALUACIÓN ::

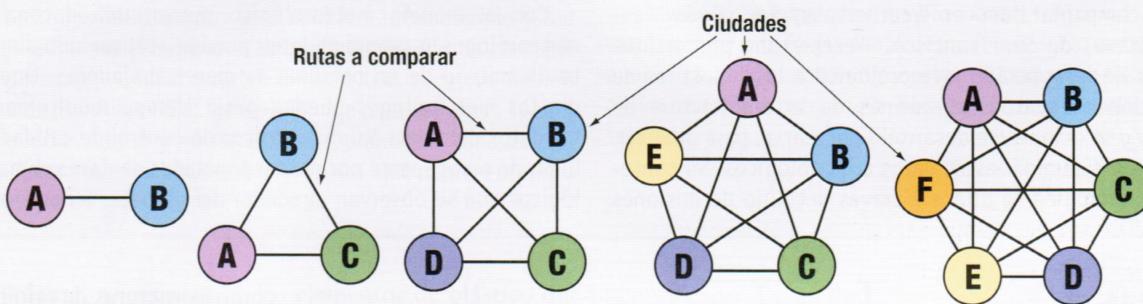
Encierre en un círculo la respuesta a cada pregunta.

1. El término _____ se refiere a las declaraciones que están escritas en un lenguaje que la computadora puede entender.
a. proceso b. sintaxis c. código
2. ¿Qué parte de un diagrama IPO describe los datos que debe introducir el usuario?
a. entrada b. proceso c. salida
3. ¿Cuál de las siguientes frases es falsa con respecto a la programación?
a. puede ser emocionante b. no debe preocuparse por la sintaxis c. existen reglas en la programación

- » **Problemas tratables.** Estos problemas (los cuales también se conocen como rápidos, eficientes o sencillos) tienen un tiempo de ejecución que crece lentamente si se compara con el tamaño de la información de entrada. Técnicamente, un problema tratable es aquel en el que el tiempo de la ecuación es una función polinomial del tamaño de la entrada. Puede resolver un problema tratable dentro de una cantidad de tiempo razonable o con una cantidad razonable de recursos computacionales.

- » **Problemas no tratables.** Estos problemas (los cuales también se conocen como lentos, ineficientes o difi-

ciles) tienen un tiempo de ejecución que crece rápidamente en comparación con el tamaño de la información de entrada. Para ser exactos, un problema intratable tiene un tiempo de ejecución que es una función exponencial del tamaño de la información de entrada. En ocasiones no se pueden resolver estos problemas, pero muy a menudo simplemente requieren grandes cantidades de tiempo o enormes recursos computacionales para alcanzar una solución.



El agregar una ciudad aumenta exponencialmente la cantidad de comparaciones entre rutas para el vendedor.

delante con el desarrollo de la **programación orientada a objetos** (OOP, por sus siglas en inglés). Los bloques de construcción de OOP llamados **objetos**, son componentes reciclables y modulares (los cuales se explicarán con detalles posteriormente).

OOP se basa en la programación estructurada y la mejora. No se excluye a la programación estructurada cuando se trabaja con un lenguaje orientado a objetos. Los objetos están compuestos de fragmentos de programas estructurados y la lógica para manipular los objetos también es estructurada.

Estructuras de programación

Los investigadores en los años sesenta demostraron que los programas podían ser escritos utilizando tres estructuras de control:

- » La **estructura secuencial** define el flujo de control predeterminado de un programa. Normalmente, esta estructura está integrada en los lenguajes de programación. A menos que se indique lo contrario, una computadora ejecuta líneas de código en el orden en el que fueron escritas.
- » Las **estructuras de selección** utilizan construcciones de un flujo de programa especial llamadas **declaraciones condicionales**. Una declaración condicional simplemente es una prueba que determina lo que hará a continuación el programa. Este tipo de prueba normalmente sólo proporcionará dos resultados, verdadero o falso. Cuando la declaración de condición es verdadera, se ejecutarán ciertas líneas de código. Si la declaración de condición es falsa, no se ejecutarán esas líneas de código. Cuando el programa toma una decisión utilizando una declaración condicional, el flujo del programa frecuentemente se dirige en una de dos direcciones distintas dependiendo del resultado de la decisión. Estas secuencias diferentes de código se llaman **ramas**. Por esta razón, las estructuras de selección se conocen frecuentemente

Norton
EN LÍNEA



Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre programación estructurada.

A discusión

Caliente y tan refrescante: la tecnología pronostica el clima

Aunque las condiciones del clima pueden cambiar en un instante, el pronóstico del clima no ha cambiado mucho desde la Segunda Guerra Mundial.

Fueron necesarias la programación Java y la conectividad a Internet para romper la tradición de recrear el modelo meteorológico y permitir a los meteorólogos obtener el máximo provecho de las tecnologías actuales. Han quedado atrás los mapas y gráficas del clima sobre papel. En su lugar está la interfaz gráfica dinámica y altamente escalable junto con los sistemas de información.

Una compañía líder es WeatherLabs, Inc. (www.weatherlabs.com) de San Francisco. WeatherLabs proporciona servicios de información meteorológica a las industrias de publicaciones electrónicas además de la arquitectura de software y soluciones de desarrollo necesarias para analizar, presentar y disseminar estos datos climatológicos. Su contenido climatológico se ofrece a través del flujo de misiones

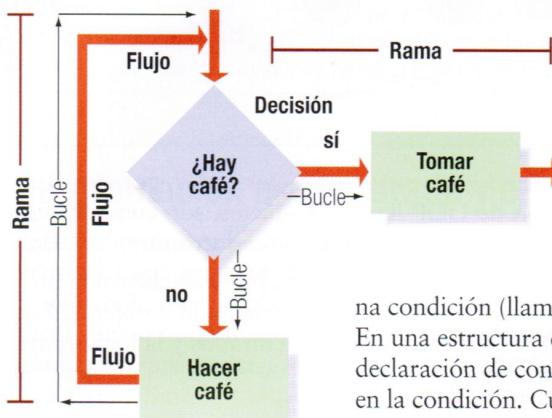
sobre Internet, por medio de tecnologías inalámbricas o móviles y a través de misiones satelitales.

El éxito de WeatherLabs se basa en un sistema de creación de pronósticos y contenido climatológico basado en una tecnología Java propietaria. De hecho, el mecanismo central en el lado del servidor representa todos los aspectos de WeatherLabs y es responsable de la inteligencia del pronóstico del tiempo y clima, análisis numérico y algoritmos de pronóstico, los cuales están incluidos directamente en los componentes JavaBeans.

Con el modelo meteorológico nuevo, una docena de meteorólogos y programadores pueden realizar actualmente un trabajo de un personal de cien trabajadores. Hoy en día los meteorólogos pueden pasar tiempo monitoreando los datos del clima para propósitos de control de calidad en lugar de preocuparse por revisar toneladas de datos climatológicos que se observan alrededor del planeta, o desarrollar

FIGURA 13A.12

En la programación estructurada, el control sigue al código secuencialmente a lo largo de una rama de código.



como estructuras de ramificación (véase la figura 13A.12).

» Las **estructuras de repetición** (o **estructuras de ciclo**) están basadas en construcciones llamadas bucles. Un bucle es un fragmento de código que se repite una y otra vez hasta que alguna condición (llamada **condición de salida**) se cumpla. En una estructura de repetición, el programa revisa una declaración de condición y ejecuta un bucle basándose en la condición. Cuando la condición es verdadera entonces se repite uno o más comandos hasta que la condición es falsa.

Programación orientada a objetos

Los conceptos de la programación orientada a objetos, por ejemplo, objetos y clases, pueden parecer abstractos en principio; sin embargo, muchos programadores afirman que la orientación a objetos es una forma natural de pensar. Debido a que OOP les ofrece una manera intuitiva de modelar el mundo, sostienen que los programas se vuelven más sencillos, la programación se hace más rápida y la carga de mantenimiento de un programa se reduce.

Objetos

Mire a su alrededor, está rodeado de objetos. En este momento, la lista de objetos que están alrededor de usted puede incluir un libro, una computadora, una lámpara, paredes, plantas, cuadros y otros objetos.

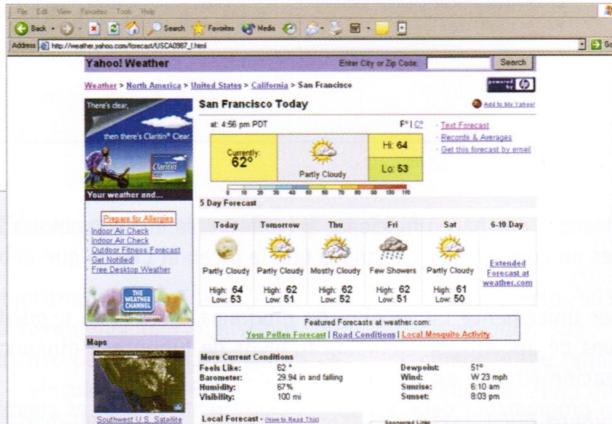
Piense por un momento sobre lo que percibe cuando observa un auto en la calle. Su primera impresión probablemente sea la del auto como una unidad. No se concentra



Norton
EN LÍNEA

Visite el sitio <http://www.mhhe.com/peternorton>

para obtener más información
sobre programación orientada a
objetos.



la presentación de los datos climatológicos. La tecnología en el lado del servidor realiza el trabajo por ellos.

El sistema WeatherLabs es altamente automatizado y fue creado para que el usuario lo utilizara fácilmente. El contenido climatológico básico se transmite a todos los clientes en forma de plantillas HTML de tipo "molde para galletas". Luego, los usuarios editan y personalizan las plantillas como deseen para crear interfaces de usuario únicas que representen su

contenido personalizado basándose en los datos climatológicos centrales. El resultado final es el contenido climatológico tan variado como las imágenes de satélites y radares Doppler, informes sobre las condiciones del cielo, pronósticos de retrasos en los aeropuertos, anécdotas de almanaques históricos del clima y contenido editorial personalizado en todo el mundo. Esto permite que el excitante centro de clima para el viajero CondéNet's tenga su propia apariencia, muy distinta que las familias de interfaz altamente gráfica de Prodigy o Netscape que utiliza Excite.

Por tanto, aunque muchos proveedores en línea aparentan tener su propio personal de meteorólogos, detrás de la apariencia, todos estos sitios están basados en las mismas aplicaciones de tecnología basada en Java en el lado del servidor.

en el volante, asientos ni en los elementos de plástico que lo conforman. La unidad entera u objeto es lo que registra en su mente.

Ahora, ¿cómo describiría ese auto a alguien sentado a su lado? Podría comenzar por sus **atributos**, por ejemplo, el color, tamaño, forma, velocidad máxima y otros aspectos. Un atributo describe las características de un objeto. Entonces podría hablar sobre lo que el auto puede hacer; es decir, describir sus funciones. Por ejemplo, se mueve hacia delante, en reversa, abre las ventanas y otras funciones. En conjunto, los atributos y las funciones definen al objeto. En el lenguaje de OOP, todos los objetos tienen atributos y funciones que pueden **encapsular** (contener) otros objetos.

Si mira con más detalle el auto, es probable que comience a notar muchos componentes más pequeños. Por ejemplo, el auto tiene un chasis, un sistema de tracción, una carrocería y un interior. Cada componente, a su vez, está compuesto de otros. El sistema de tracción incluye un motor, transmisión, eje trasero y delantero. De manera que un objeto puede ser una unidad entera o un componente de otros objetos. Los objetos pueden incluir otros. La figura 13A.13 muestra un auto lleno de atributos, funciones y objetos encapsulados.

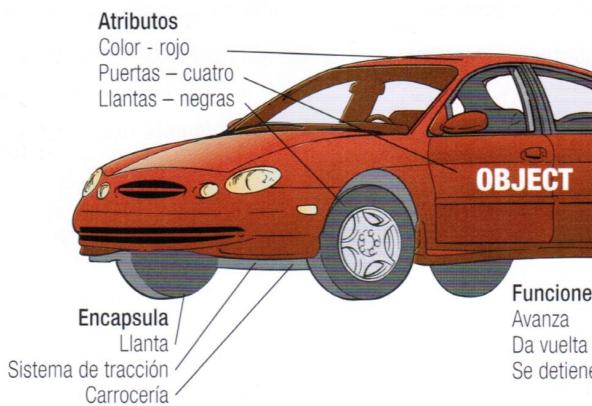


FIGURA 13A.13

Este objeto tiene atributos y funciones; también encapsula otros objetos, como llantas, sistema de tracción y carrocería, cada uno con sus atributos y funciones únicas.

Resumen ::

- » El software contiene los comandos que hacen funcionar al hardware de una computadora.
- » Un programa de cómputo es un conjunto de comandos que le dicen al CPU lo que debe hacer.
- » El software puede contener únicamente un archivo de programa ejecutable o puede tener algunos otros archivos de apoyo, por ejemplo, archivos de biblioteca dinámica de vínculos (.dll), inicialización (.ini) y ayuda (.hlp).
- » Para crear un programa, un programador debe escribir código fuente el cual es compilado o interpretado para crear código objeto.
- » El código objeto, también conocido como código máquina, es el archivo de lenguaje binario que le indica al CPU lo que debe hacer.
- » El orden en el cual se ejecutan las declaraciones de un programa se conoce como el flujo de control del programa.
- » Los programadores traducen los algoritmos y la heurística en programas diseñados para resolver problemas.
- » La programación estructurada utiliza funciones integradas y un flujo de programa lógico para realizar cada tarea que describe el algoritmo o heurística.
- » La programación orientada a objetos permite que un programador piense en función de módulos ya que los programas están ensamblados con base en componentes, llamados objetos.
- » En programación, un objeto es una unidad autocontenido que incluye funciones y atributos.

Términos importantes ::

algoritmo, 505
archivo de ayuda (.hlp, .chm), 500
archivo de biblioteca dinámica de vínculos (.dll), 500
archivo de inicialización (.ini), 500
archivo de secuencia de comandos (.bat), 501
archivo ejecutable (.exe, .com), 500
atributo, 511
bucle, 510
código, 501
código fuente, 502
código máquina, 502

código objeto, 503
compilador, 503
condición de salida, 510
control de flujo del programa, 505
declaración condicional, 509
diagrama de flujo, 505
diagrama de entrada-proceso-salida (IPO), 504
encapsular, 511
estructura de ciclo, 510
estructura de repetición, 510
estructura secuencial, 509
estructura de selección, 509

heurística, 506
intérprete, 503
interrupción, 501
lenguaje de programación, 502
lenguaje máquina, 502
objeto, 509
programación estructurada, 508
programación orientada a objetos (OOP), 509
pseudocódigo, 504
rama, 510

Prueba de términos importantes ::

Complete cada oración escribiendo alguno de los términos listados en “Términos importantes”, en cada espacio en blanco.

1. Un archivo _____ contiene información de configuración que coopera en la ejecución de un programa.
2. Los programadores utilizan _____ para convertir todo el código fuente de un programa en código máquina, con lo cual se crea el archivo ejecutable.
3. Para planear un programa, el programador debe utilizar un diagrama _____.
4. Un _____ es un conjunto de pasos que siempre conducen a una solución.
5. La programación _____ apareció en los años sesenta como una manera para que los desarrolladores evitaran la creación de “código espagueti”.
6. La _____ no ofrece la garantía de encontrar la mejor solución posible.
7. Un _____ convierte el código fuente en código máquina, pero no crea un archivo de código objeto ejecutable.
8. Una _____ es una declaración que determina lo que debe hacer el programa a continuación.
9. Los bloques de construcción de la programación orientada a objetos son bloques reciclables de código llamados _____.
10. C++ y Basic son ejemplos de un _____.

Opción múltiple ::

Circule la palabra o frase que complete mejor cada oración:

1. Un programa de computadoras es un conjunto de _____ que son ejecutadas por el CPU de la computadora.
a. instrucciones b. bucles c. pasos d. algoritmos
2. Los lenguajes de programación permiten que los programadores describan programas. Esta descripción se conoce como _____.
a. un OOP b. código fuente c. intérprete d. objeto
3. Un _____ es una versión simplificada del lenguaje hablado de un programa.
a. pseudocódigo b. flujo de control c. algoritmo d. diagrama de flujo
4. Un bucle se repite hasta que se cumple un(a) _____.
a. objeto b. error c. condición de salida d. sintaxis
5. Un objetivo de los(as) _____ es desarrollar un código que se pueda leer.
a. lenguajes de programación b. intérpretes c. compiladores d. programaciones estructuradas
6. _____ utiliza la programación estructurada y la mejora.
a. Un compilador b. Un lenguaje de programación c. La programación orientada a objetos d. El flujo de control del programa
7. Un(a) _____ son los pasos para resolver un problema que no garantizan una respuesta.
a. algoritmo b. diagrama IPO c. heurística d. diagrama de flujo
8. Los unos y ceros que los CPU pueden entender se conocen como su _____.
a. sintaxis b. interrupción c. intérprete d. lenguaje máquina
9. Un archivo _____ es un archivo .exe parcial que permite que otros programas accedan a otros comandos.
a. .ini b. .dll c. .bat d. .dat
10. El orden en el que se ejecutan las declaraciones de un programa se conocen como _____.
a. flujo de control b. sintaxis del programa c. lenguaje de programación d. ejecución de un programa

Preguntas de revisión ::

Con sus propias palabras, conteste brevemente las siguientes preguntas.

1. ¿Qué es lo que va dentro de las tres columnas de un diagrama IPO?
2. Describa las diferencias entre un compilador y un intérprete.
3. ¿Por qué razón se desarrollaron los lenguajes de programación?
4. ¿De qué son ejemplos las declaraciones y bucles?
5. ¿Qué es un archivo ejecutable?
6. Describa la forma en que el reciclaje de código puede simplificar el trabajo de un programador.
7. ¿Cuál es el beneficio de planear el código antes de intentar escribir?
8. ¿Cuáles son las ventajas de que un programa utilice archivos de bibliotecas dinámicas de vínculos (.dll)?
9. ¿Qué es un diagrama de flujo?
10. ¿Cuál es la diferencia entre un algoritmo y la heurística?

Laboratorios de la lección ::

Complete el siguiente ejercicio según las indicaciones de su instructor.

1. Realice los pasos siguientes para ver la forma en que muchos archivos ejecutables se almacenan en la carpeta Windows de su computadora:
 - a. Inicie Windows Explorer (el procedimiento para hacerlo depende de la versión de Windows que esté instalada en su computadora. Si es necesario pida ayuda a su instructor).
 - b. En el panel izquierdo de la ventana Explorar, encuentre y haga clic en la carpeta Windows. El contenido de la carpeta aparecerá en el panel derecho de la ventana.
 - c. En el panel derecho, haga clic en el encabezado de la columna Teclear. Esta acción ordena el contenido del panel por archivos o carpetas.
 - d. Liste todos los archivos ejecutables (.exe) que encuentre en la carpeta. La cantidad de archivos ejecutables puede variar de un sistema a otro. Compare la cantidad de archivos ejecutables de su sistema con la de otros estudiantes.
 - e. Encuentre el archivo Bloc de notas.exe en la carpeta Windows y haga doble clic en el archivo. Se abrirá el programa de edición de texto Bloc de notas en su pantalla.
 - f. Cierre Bloc de notas y luego cierre la ventana Explorar.

Lenguajes de programación y el proceso de programación

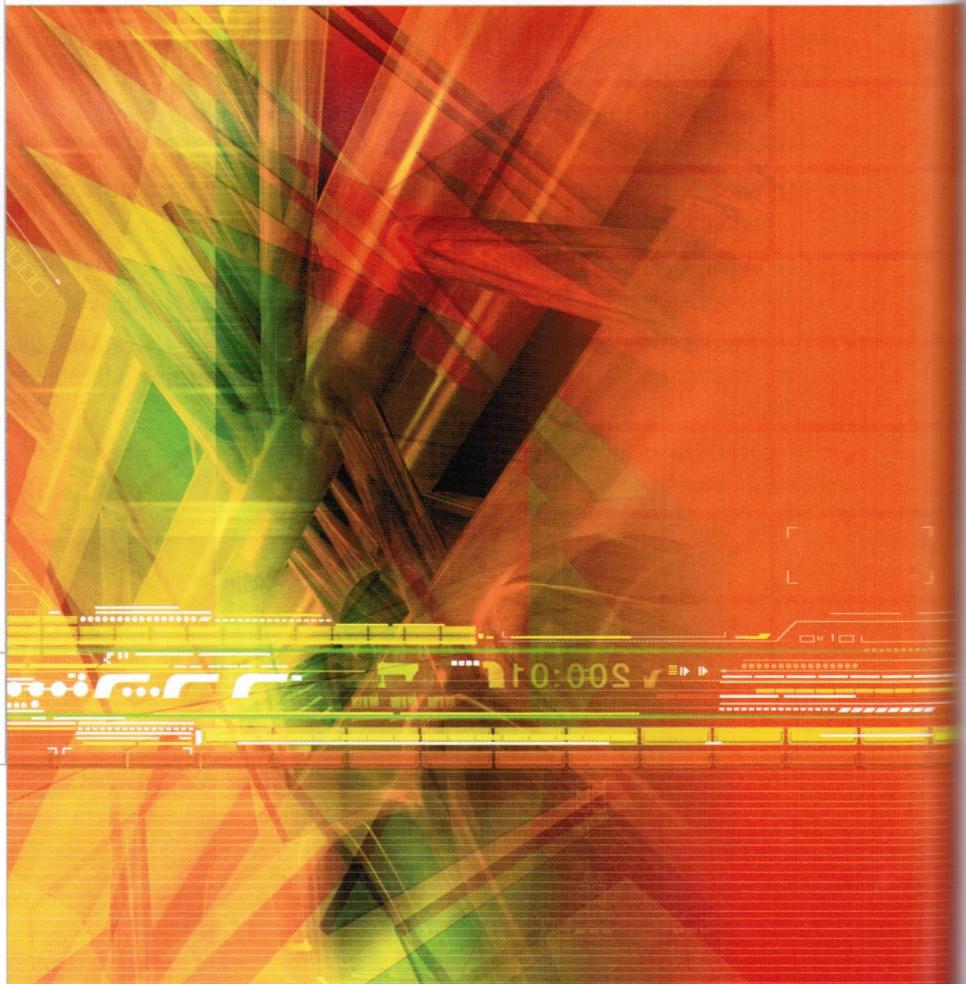
Panorama general: las claves para una programación exitosa

La programación puede ser un proceso completo; requiere de capacitación, planeación y algunas herramientas especializadas. Si piensa que esta tarea es parecida a cualquier otro proceso de construcción tiene razón; crear un programa de software es parecido a construir un edificio de oficinas o un centro comercial. Tiene que estudiar el mercado, crear un plan, contratar personas, obtener herramientas especiales y entonces (finalmente) crear la construcción. Pero el trabajo no se detiene ahí. Debe actualizar continuamente su creación de manera que satisfaga las necesidades cambiantes de las personas que la utilicen.

Los programadores exitosos están bien informados en dos áreas importantes: las herramientas de programación (el software y los lenguajes que se utilizan para desarrollar aplicaciones) y el proceso de programación (los procedimientos de paso a paso que los programadores siguen para asegurar la consistencia de productos bien desarrollados). Los desarrolladores de software de todos los tipos (y de muchos entornos distintos de trabajo) siguen un conjunto uniforme de procedimientos en sus trabajos. Por tanto, los programadores pueden trabajar más fácilmente en proyectos grandes y pueden predecir de forma exacta cómo funcionarán sus programas. Aprenderá más sobre el proceso de desarrollo más tarde en esta lección. Primero, conocerá las herramientas especiales que utilizan los programadores: los lenguajes de programación.

OBJETIVOS ::

- » Identificar las tres categorías principales de lenguajes de programación.
- » Describir las cinco generaciones de lenguajes de programación.
- » Nombrar al menos cinco lenguajes de programación importantes.
- » Listar las cinco fases del ciclo de vida del desarrollo de sistemas para la programación.



La evolución de los lenguajes de programación

Como aprendió anteriormente en este capítulo, la programación es una forma de crear un conjunto de instrucciones para la computadora. Para crear estas instrucciones, los programadores utilizan lenguajes de programación que están definidos de forma rígida con el fin de crear código fuente; entonces convierten el código fuente en código máquina (u objeto) para la computadora.

El código máquina (los unos y los ceros) es el único lenguaje que una computadora entiende. Sin embargo, las personas tienen dificultad para entender el código máquina. Para hacer que el desarrollo de software sea más sencillo, los investigadores desarrollaron de forma progresiva lenguajes de programación más sofisticados. Esta evolución en las herramientas de desarrollo permitió que los programadores se enfocaran menos en cadenas de números y más en las secuencias de comandos. En otras palabras, los desarrolladores pudieron pensar sobre sus programas en términos humanos en lugar de términos de computadoras. Como resultado, los programadores actuales pueden crear secuencias de comando que usted puede leer como si fuera cualquier otro lenguaje. Las herramientas de programación manejan la parte tediosa de convertir las instrucciones de tipo humano en cadenas de números que las computadoras pueden entender.

Como aprenderá en esta lección, los programadores pueden escoger entre muchas herramientas de desarrollo que difieren enormemente en capacidad, flexibilidad y facilidad de uso. Sin embargo, a pesar de sus diferencias la mayoría de los lenguajes de programación comparten una característica: cada lenguaje de programación requiere que el programador siga algunas reglas muy estrictas. Por ejemplo, los lenguajes de programación requieren que los desarrolladores:

- » Proporcionen información en un orden y estructura específicos.
- » Utilicen símbolos especiales.
- » Utilicen puntuaciones (algunas veces).

Estas reglas se conocen como la **sintaxis** del lenguaje de programación y pueden variar bastante de un lenguaje a otro. La figura 13B.1 muestra un fragmento de código fuente escrito en un lenguaje de programación llamado C++.

Observe que en la figura 13B.1, cada línea de código incluye abreviaciones especiales, espaciado o puntuación. Este estilo de escritura especial es la sintaxis del lenguaje de programación C++. Si el código estuviera escrito en un lenguaje distinto, usted vería diferencias en el espaciado, frases, puntuación y otras características. No obstante, sin importar qué lenguaje utilice el programador debe seguir la sintaxis correcta para ese lenguaje. Cuando la sintaxis no es correcta, el compilador o intérprete emitirá errores y advertencias y no podrá entender el código fuente. Además fallará en la creación del código objeto o será incorrecto.

Categorías de los lenguajes de programación

Cientos de lenguajes de programación se utilizan actualmente en todo el mundo. Algunos son altamente especializados y se usan únicamente en una rama de la ciencia o industria, mientras que otros son bien conocidos y se utilizan casi en cualquier otro lado. Algunos lenguajes son obsoletos y únicamente se utilizan para mantener los sistemas más viejos, mientras que otros son tan nuevos que muchos programadores ni siquiera saben que existen.

Puede agrupar los lenguajes de programación de maneras muy distintas. Por ejemplo, puede agruparlos en aquellos que utilizan la programación estructurada y aquellos que no. O puede agruparlos por los que se utilizan en las empresas a diferencia de los que se utilizan en círculos científicos. Sin embargo, los lenguajes de programación normalmente

Norton EN LÍNEA

Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre la historia de los lenguajes de programación.



simnet™

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "definitions - Microsoft Visual C++ [design] - definitionsDlg.cpp". The main window displays C++ code for the "DefinitionsDlg" class. The code is as follows:

```
void DefinitionsDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKND, reinterpret_cast<WPARAM>(dc
            // Center icon in client rectangle
            int cxIcon = GetSystemMetrics(SM_CXICON);
            int cyIcon = GetSystemMetrics(SM_CYICON);
            CRect rect;
            GetClientRect(rect);
            int x = (rect.Width() - cxIcon + 1) / 2;
            int y = (rect.Height() - cyIcon + 1) / 2;

            // Draw the icon
            dc.DrawIcon(x, y, m_hIcon);
        }
    }
}
```

FIGURA 13B.1

La sintaxis de cada lenguaje de programación determina el espacio, los símbolos, palabras y puntuación exactos usados en el código.



Norton EN LÍNEA

Visite el sitio <http://www.mhhe.com/peternorton>

para obtener más información sobre los lenguajes máquina y ensambladores.

se agrupan por su lugar en la evolución de los lenguajes de programación. En relación con su historia evolutiva, los lenguajes de programación están divididos en tres categorías:

- » Lenguajes máquina
- » Lenguajes ensamblador
- » Lenguajes de alto nivel

Lenguajes de código máquina y ensamblador

Los lenguajes de código máquina son los lenguajes más fundamentales. Utilizando un lenguaje máquina, un programador crea instrucciones en la forma de código máquina (unos y ceros) que una computadora puede seguir. Los lenguajes máquina están definidos por el diseño del hardware. En otras palabras, el lenguaje máquina de una Macintosh no es el mismo que el lenguaje máquina de una PC Pentium. De hecho, el lenguaje máquina para las distintas versiones de Pentium es ligeramente distinto. Una computadora sólo entiende su lenguaje máquina nativo, los comandos de su conjunto de instrucciones. Estos comandos le indican a la computadora que realice operaciones elementales como cargar, almacenar, sumar y restar.

Los **lenguajes ensamblador** fueron desarrollados mediante el uso de abreviaciones cortas del idioma inglés para representar elementos comunes del código máquina. Para desarrollar software con un lenguaje ensamblador, un programador debe utilizar un editor de texto (un procesador de texto simple) para crear archivos de código. Para convertir los archivos fuente en código objeto, el desarrollador utiliza un programa de traducción especial, llamado **ensamblador**, para convertir cada línea de código ensamblador en una línea de código máquina. De ahí proviene el nombre *lenguaje ensamblador*. Aunque los lenguajes ensamblador son altamente detallados y arduos, son mucho más sencillos de usar que el lenguaje máquina. Prácticamente la única ocasión en que los programadores escriben programas de un tamaño significativo en un lenguaje ensamblador es cuando les preocupa que el código sea eficiente y rápido (un ejemplo de esta regla son los juegos de acción, en donde la velocidad del programa es muy importante). De otra forma, los programadores utilizan los lenguajes ensamblador para afinar partes importantes de programas que están escritos en un lenguaje de nivel más alto.

No puede decir mucho al observar el código máquina (unos y ceros) y tendría que saber una enorme cantidad de información especializada antes de poder escribirlo. Sin embargo, el código ensamblador es un poco menos tedioso... pero sólo un poco. Como puede ver en la figura 13B.2, el código ensamblador utiliza frases parecidas al idioma inglés especializadas junto con números específicos de hardware.

FIGURA 13B.2

El código máquina sólo tiene unos y ceros, pero el código ensamblador se encuentra un paso más cerca del inglés.

Lenguajes de alto nivel

Los **lenguajes de alto nivel** fueron desarrollados para hacer que la programación fuera más sencilla. Estos lenguajes se conocen como lenguajes de alto nivel debido a que sus sintaxis son más parecidas a los idiomas humanos que el código de lenguaje

| Código ensamblador | Código máquina |
|---|--|
| <pre>;CLEAR SCREEN USING BIOS CLR: MOV AX,0600H ;SCROLL SCREEN MOV BH,30 ;COLOUR MOV CX,0000 ;FROM MOV DX,184FH ;TO 24,79 INT 10H ;CALL BIOS; ;INPUTTING OF A STRING KEY: MOV AH,0AH ;INPUT REQUEST LEA DX,BUFFER ;POINT TO BUFFER WHERE STRING STORED INT 21H ;CALL DOS RET ;RETURN FROM SUBROUTINE TO MAIN PROGRAM; ;DISPLAY STRING TO SCREEN SCR: MOV AH,09 ;DISPLAY REQUEST LEA DX,STRING ;POINT TO STRING INT 21H ;CALL DOS RET ;RETURN FROM THIS SUBROUTINE;</pre> | <pre>000101001011010101010101010101 111011010101010101010101011100 00101001010100101111010111 1001010010110101010101010101 01101001001100101111010111 00010001010111010101010001 10101001010100101011010111 00010100101101010101010101</pre> |

ensamblador o máquina. Los lenguajes de programación de alto nivel utilizan palabras familiares en lugar de cadenas detalladas de dígitos que forman las instrucciones máquina. Para expresar operaciones de computadoras, estos lenguajes utilizan operadores como los signos de más y de menos, los cuales son componentes familiares de las matemáticas. Como resultado, las personas pueden leer, escribir y entender programas de computadoras de manera mucho más sencilla cuando utilizan un lenguaje de nivel más alto. Aún así, las instrucciones deben ser traducidas al lenguaje máquina antes de que la computadora pueda entenderlas y llevarlas a cabo. Una línea de un lenguaje de alto nivel normalmente se traduce en muchas líneas de lenguaje máquina. Esto hace que sea más rápido escribir en un lenguaje de alto nivel, pero sacrifica parte del control sobre el código real que se produce.

Los lenguajes de programación son discutidos con frecuencia en términos de generaciones. Las últimas generaciones incluyen lenguajes que son fáciles de usar y más poderosos que los de las generaciones anteriores. Por tanto, los lenguajes máquina se conocen como **lenguajes de primera generación** y los lenguajes ensamblador son **lenguajes de segunda generación**. Los lenguajes de alto nivel comenzaron con la tercera generación.

Lenguajes de tercera generación

Los **lenguajes de tercera generación (3GL)**, por sus siglas en inglés) hacen que sea más fácil escribir programas estructurados. Debido a que son los primeros lenguajes en utilizar realmente frases parecidas al idioma inglés, también hacen que sea más fácil que los programadores compartan su trabajo en el desarrollo de programas. Los miembros del equipo pueden leer el código fuente de los demás y entender la lógica y el flujo de control del programa. La figura 13B.3 muestra dos fragmentos de código fuente de los lenguajes de tercera generación, uno escrito en BASIC y el otro en C. Observe que aunque estos lenguajes de tercera generación son distintos uno del otro en sus sintaxis, ambos se parecen bastante al idioma inglés y no son tan difíciles de comprender.

Otro aspecto importante que hay que recordar sobre los lenguajes de tercera generación es que estos lenguajes son **transportables**. Si tiene un compilador o intérprete para una computadora y sistema operativo en particular, puede utilizar el ensamblador y compilador para crear un archivo ejecutable utilizando el código fuente (este procedimiento se conoce como **transportar** el código a otro sistema). Es probable que tenga que modificar el código fuente un poco cuando lo transporte, especialmente si lo está transportando a un tipo de computadora completamente distinto (por ejemplo, de una PC a una mainframe). Siempre y cuando tenga un compilador o intérprete para un tipo de sistema determinado, normalmente podrá convertir su código fuente a código objeto para ese tipo de máquina. En contraste, el código máquina y el código ensamblador son muy específicos para su procesador y deben estar escritos especialmente para cada tipo de computadora en la cual se ejecutan.

Existen muchos lenguajes de alto nivel y no existe ninguna razón por la cual deba conocer los detalles de cada uno. Sin embargo, siempre es útil conocer un poco sobre los lenguajes más comunes que probablemente haya escuchado en los círculos de programación. Algunos de los lenguajes populares en la actualidad son los siguientes:

- » **C.** A veces se considera como el “pura sangre” de los lenguajes de programación, C produce programas en código ejecutable rápido y eficiente. C también es un lenguaje poderoso. Con él, puede hacer que una computadora haga casi cualquier cosa que le sea posible hacer. Debido a la libertad de esta programación, C es extremadamente popular entre los desarrolladores profesionales, aunque actualmente está siendo reemplazado por C++.
- » **C++.** Es la implementación orientada a objetos de C. Al igual que C, C++ es un lenguaje extremadamente poderoso y eficiente. Aprender C++ significa conocer todo acerca de C y luego aprender sobre la programación orientada a objetos y su implementación con C++. Sin importar esto, cada vez más programadores C se cambian a C++ todos los años y el nuevo lenguaje ha reemplazado a C como el lenguaje de preferencia entre las compañías de desarrollo de software.
- » **Java.** Es un entorno de programación orientado a objetos para crear programas que funcionen en distintas plataformas. Cuando Internet se volvió popular a

Norton EN LÍNEA



Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre los lenguajes de alto nivel.

FIGURA 13B.3

A pesar de sus especiales reglas de sintaxis, los 3GLs usan palabras y frases parecidas al inglés.

Código BASIC

```
IF D& > 15 THEN
  DO WHILE D& > 1
    D& = D& - 1
  LOOP
END IF
```

Código C

```
if (d > 15)
{
  do
  {
    d--;
  } while (d > 1);
```

mediados de los años noventa, el desarrollador de Java, Sun Microsystems, desarrolló Java para que fuera un entorno de programación para Internet. Más tarde, Sun añadió la capacidad de escribir programas que no se ejecutaran dentro de un navegador. Con Java, los diseñadores de la Web pueden crear programas interactivos y dinámicos (llamados **applets**) para páginas Web.

- » **ActiveX.** La respuesta de Microsoft a Java es ActiveX. El código de ActiveX crea funciones autocontenido similares a las applets de Java que pueden ser accedidas y ejecutadas por cualquier otro programa compatible con ActiveX en cualquier sistema o red ActiveX. ActiveX puede crear aplicaciones de firma. Una aplicación de firma es aquella que se ha verificado como segura para ser ejecutada en una computadora. Actualmente, ActiveX está implementado en Windows9x, Windows NT, Windows 2000, Windows XP y los sistemas Macintosh, y también existen planes para proporcionar soporte para UNIX.

Lenguajes de cuarta generación

Los **lenguajes de cuarta generación (4GL)**, por sus siglas en inglés) son más fáciles de usar que los lenguajes de tercera generación. Generalmente, un 4GL utiliza un entorno de texto, muy parecido al de 3GL o un entorno visual.

En el entorno de texto, el programador utiliza palabras del idioma inglés cuando genera código fuente. Normalmente, una sola declaración en un 4GL puede realizar las mismas tareas que muchas líneas de un 3GL.

En un entorno visual 4GL, el programador utiliza una barra de herramientas para arrastrar y soltar distintos elementos como, por ejemplo, botones, etiquetas y cuadros de texto con el fin de crear una definición visual de una aplicación. Una vez que el programador ha diseñado la apariencia del programa, puede asignar distintas acciones a los objetos de la pantalla. Por ejemplo, un programador puede colocar un botón en la pantalla y asignar una acción como "Abrir la tabla de clientes". La figura 13B.4 muestra un entorno de desarrollo visual, en este caso, Visual Basic.NET.

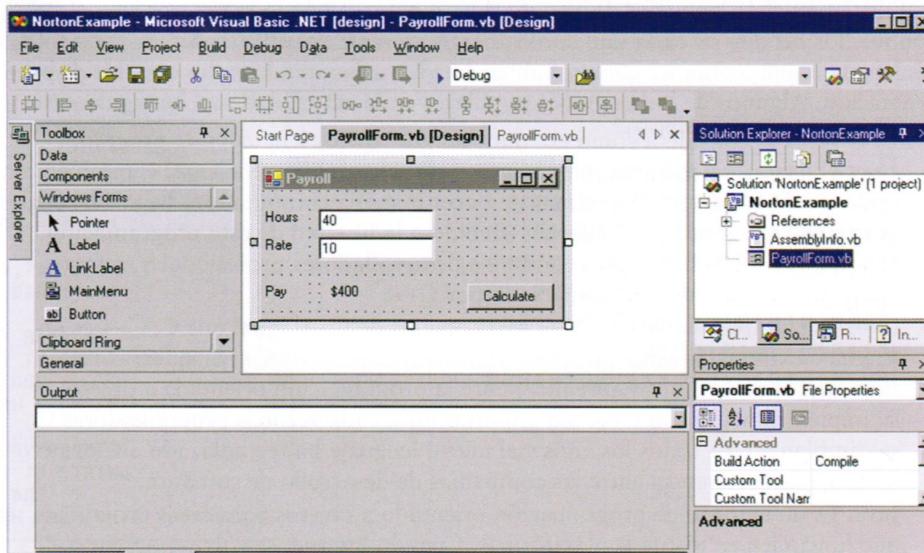
La mayoría de los 3GL y los 4GL permiten que el trabajador trabaje en un **entorno de desarrollo integrado**, o **IDE** (por sus siglas en inglés). Los IDE proporcionan al programador todas las herramientas necesarias para desarrollar aplicaciones en un programa. Incluyen compiladores y soporte de tiempo de ejecución para sus aplicaciones. Visual Studio de Microsoft y Java Studio de Sun, son dos IDE profesionales.

Entre los lenguajes de cuarta generación se incluyen los siguientes:

- » **.NET** es el nuevo producto de Microsoft en el campo de la programación. Combina varios lenguajes de programación en un IDE. Los lenguajes incluidos son Visual Basic, C++, C# y J#. .NET está incluido como el único entorno del desarrollador. Utilizando .NET, los desarrolladores pueden escribir programas para Windows, la

FIGURA 13B.4

Creación de una forma usando las herramientas visuales. Por ejemplo, para colocar un cuadro en una forma, los programadores en Visual Basic simplemente arrastran la caja desde un cuadro de herramientas hasta la forma. Pueden ajustar la longitud del cuadro al arrastrar sus esquinas.



World Wide Web y PocketPC (como discutimos en el capítulo 7, PocketPC es una versión de Windows diseñada para los PDA). .NET permite que la autoría de programas para todos estos entornos sea más fácil. La figura 13B.5 lista algunos de los tipos de aplicaciones que se pueden crear con la versión Educational de Visual Studio.NET.

- » **Entornos de autoría.** Los entornos de autoría son herramientas de programación de propósitos especiales para crear aplicaciones multimedia, programas de capacitación por computadora, páginas Web y otras aplicaciones. Un ejemplo de un entorno de autoría es Macromedia Director (el cual utiliza el lenguaje de secuencia de comandos Lingo). Puede utilizarlo para crear productos multimedia combinando clips musicales, texto, animación, imágenes y otros elementos. Como cualquier otro entorno de desarrollo visual, gran parte del código se escribe de forma automática. Sin embargo, la mayoría de los entornos de autoría robustos también incluyen sus propios lenguajes, llamados lenguajes de secuencias de comandos, los cuales proporcionan herramientas para añadir control sobre el producto final. Los programas que se utilizan para crear páginas de la World Wide Web caen dentro de otra categoría de herramientas que a menudo se agrupan en los entornos de autoría. Algunos de estos programas son Microsoft FrontPage, Netscape Visual JavaScript y NetObjects Fusion.
- » **Sun Studio One.** Es un editor visual para las applets Java y Swing. Un applet es un programa que se ejecuta dentro de una página Web. Studio One proporciona un IDE completo además de distintos asistentes para automatizar tareas comunes como crear una applet. Studio One tiene una ventaja en comparación con otros entornos Java: fue desarrollada por Sun, los creadores de Java.

Lenguajes de quinta generación

Los **lenguajes de quinta generación (5GL)**, por sus siglas en inglés) son en realidad un poco misteriosos. Dependiendo del experto al que se lo pregunte, es probable que ni siquiera estén de acuerdo en que los 5GL existen. Algunos expertos consideran que los entornos de autoría más avanzados son 5GL, mientras que otros piensan lo contrario. En principio un 5GL podría utilizar inteligencia artificial para crear software basándose en la descripción de lo que el software debe realizar. Este tipo de sistema está probando que es más difícil de inventar que el código que se supone que crearía.

Lenguajes de desarrollo de la World Wide Web

Pocos componentes tecnológicos de la actualidad han afectado nuestra cultura como Internet y la World Wide Web. Internet ha evolucionado de mensajes simples como texto, a sitios Web complejos que son visuales, interactivos y ofrecen respuestas. De igual forma, las herramientas de desarrollo que se relacionan con la Web han evolucionado en poder y capacidades. Por tanto, es imposible hablar en un contexto contemporáneo sobre programación y desarrollo sin tomar en cuenta las herramientas que hacen que sea posible el desarrollo de la Web.

- » **Lenguaje de marcación de hipertexto (HTML,** por sus siglas en inglés). El lenguaje de marcación de hipertexto es el lenguaje de programación que se utiliza para crear documentos para la World Wide Web. Utilizando HTML, puede definir la estructura de un documento Web empleando componentes tales como atributos y etiquetas. Las etiquetas, como recordará de las discusiones anteriores sobre páginas Web, proporcionan vínculos a otros puntos del documento, a otros documentos del mismo sitio o a documentos de otros sitios. Las etiquetas HTML también se utilizan para dar formato a la apariencia de una página Web, insertar imágenes y elementos multimedia e incorporar componentes que se crean en otros lenguajes de programación como Java o Flash.

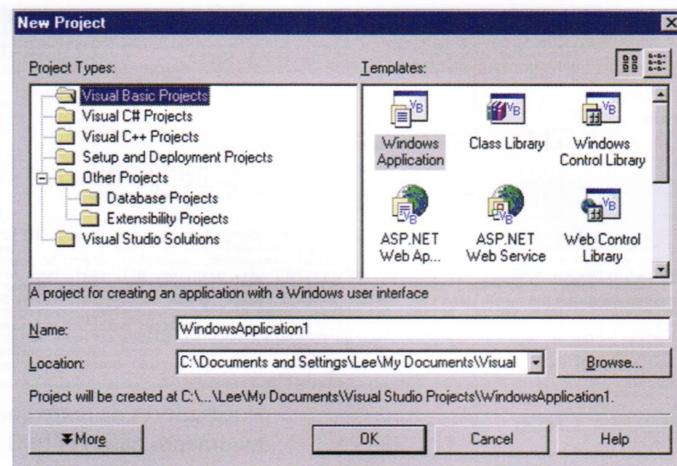


FIGURA 13B.5

Visual Studio.NET es un ambiente muy flexible, que soporta la mayoría de las necesidades de un programador.

Norton
EN LÍNEA



Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre el desarrollo de lenguajes para la World Wide Web.

simnet™

```

<!-- Begin: home-clamp.jsp -->
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>&gt; energy.gov : DOE Home</title>
    <link rel="stylesheet" href="/admin/css/header.css" type="text/css">
    <link rel="stylesheet" href="/engine/doe/css/screen.css" type="text/css">
    <link rel="stylesheet" href="/admin/css/authors.css" type="text/css">
    <LINK REL="SHORTCUT ICON" HREF="/doe.ico">
    <meta name="keywords" content="department of energy, energy, DOE, energy efficiency, science, research, education, alternative energy, renewable energy, fossil fuel, nuclear power, nuclear energy, environment, environmental management, education, electric power, medical sciences, electricity, consumer information, gas, gasoline, oil, coal, home heating oil, water power, wind power, natural gas, solar, national laboratories, human genome project, medical research, physics, fusion, climate change, energy passes, technology, petroleum, supplies, demand, prices, gas prices, home heating oil supplies, strategic petroleum reserves, safety, health, hydroelectricity, nanotechnology, material sciences, chemical sciences, medical isotopes, high performance computing, plasma science, schools, bioengineering, uranium, space power, powerplants, energy savers, Energy saving, saving, energy, Energy bill, Landscaping, Landscaping ideas, Landscaping tips, Energy conservation, " />

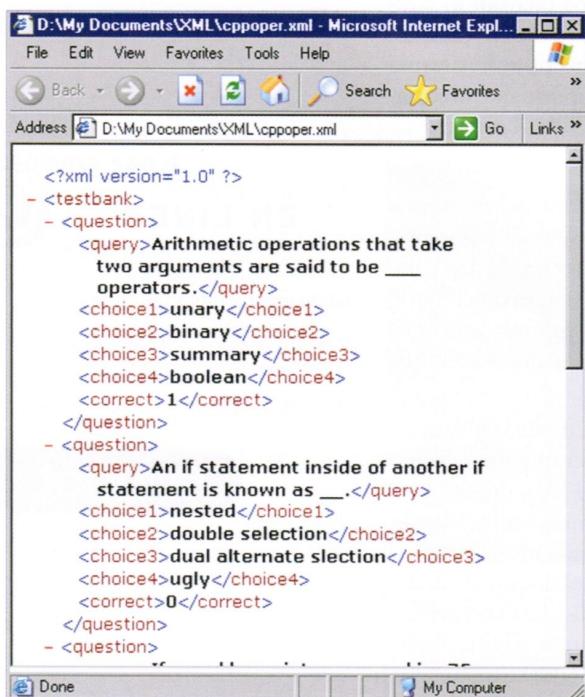
```

FIGURA 13B.6

Un ejemplo de código HTML del sitio Web del U. S. Department of Energy.

FIGURA 13B.7

Esta base de datos XML tiene preguntas de opción múltiple. Observe que las etiquetas son fáciles de entender.



Debido a que HTML carece de varias características importantes de los lenguajes de programación formales, por ejemplo, la capacidad de hacer selecciones, la mayoría de los programadores no piensan que HTML es un verdadero lenguaje de programación. Sin embargo, para un programador novato, HTML puede hacer cosas maravillosas en una página Web. Un hecho sobre HTML que hace que sea tan sencillo de usar es que puede escribir el código HTML en un editor de texto simple (por ejemplo, Bloc de notas) o cualquier otro programa de procesamiento de texto. De hecho, muchos

procesadores de texto cuentan con una opción HTML con la cual puede guardar su documento como HTML para ser publicado en la Web. Muchos otros tipos de programas de aplicación comunes, por ejemplo, hojas de cálculo y programas de presentación, también pueden convertir automáticamente un documento estándar en formato HTML, incluyendo hipervínculos y otras herramientas de navegación. Sin embargo, los diseñadores profesionales de la Web normalmente utilizan programas de edición de HTML como Microsoft FrontPage para crear páginas Web. La figura 13B.6 muestra un ejemplo de las etiquetas HTML en un documento de la Web.

- » **Lenguaje de marcación extensible (XML).** Un lenguaje de descripción del contenido de la Web de la siguiente generación, **Lenguaje de marcación extensible (XML)**, por sus siglas en inglés) normalmente se refiere a un nuevo lenguaje de marcación que permite que los desarrolladores describan una página de manera que un documento fuente pueda ser presentado en muchos formatos distintos, por ejemplo, una página Web, un documento que se puede imprimir y un archivo PDF. En estructura, XML tiene una apariencia similar a HTML, pero el desarrollador tiene la libertad de crear nuevas etiquetas. XML no reemplaza a HTML. XML necesita HTML y otras tecnologías para desplegar correctamente sus datos. La figura 13B.7 muestra una base de datos XML con preguntas de opción múltiple. Cada pregunta tiene cuatro opciones. La etiqueta correcta describe cuál de las opciones es correcta; 0 es la primera opción.

» **HTML extensible (XHTML),** por sus siglas en inglés). XHTML es la nueva versión de HTML. Actualmente XHTML es el estándar para desarrollar páginas Web. Es muy similar en todos los aspectos a HTML. Sin embargo, las reglas son más estrictas. HTML permite una codificación bastante “suelta”. XHTML requiere que todos los elementos estén “bien formados”. Esto significa que el desarrollador debe escribir código XHTML perfecto todas las veces. XHTML se está convirtiendo en el lenguaje estándar de los desarrolladores de la Web.

- » **Lenguaje de hojas de estilo extensible (XSL).** El **Lenguaje de hojas de estilo extensible (XSL)**, por sus siglas en inglés) es una de las tecnologías XML. Su propósito es desplegar y dar formato a documentos XML para los navegadores HTML como el Internet Explorer. El documento XSL se compone de distintas reglas que dictan la manera en que el documento debe ser formateado. Una vez que el documento XML se abre en un navegador, se aplican las reglas XSL. El usuario solamente ve una página HTML normal. Utilizando XSL es relativamente sencillo tener un documento XML con varias vistas distintas.
- » **Lenguaje de marcación extensible de perfil móvil (XHTML MP).** En los años recientes, cada vez más personas han comenzado a utilizar dispositivos pequeños (por ejemplo, los PDA) para conectarse a Internet utilizando una

tecnología inalámbrica, por ejemplo, módems celulares. Esta demanda ha creado la necesidad de nuevos entornos de desarrollo como el **Lenguaje de marcación extensible de perfil móvil (XHTML MP**, por sus siglas en inglés), que anteriormente se conocía como **Lenguaje de marcación inalámbrico (WML**, por sus siglas en inglés). Los diseñadores de la Web pueden utilizar WML para crear documentos que se pueden ver en dispositivos de bolsillo como los teléfonos celulares con capacidades para la Web, los PDA e incluso localizadores digitales. A medida que el hardware miniatura avance enormemente en la calidad de despliegue y las capacidades del procesamiento del ancho de banda, los lenguajes como XHTML MP se utilizarán más comúnmente.

- » **Dreamweaver** de Macromedia es un editor HTML que permite que el desarrollador escriba visualmente páginas Web. Los desarrolladores pueden utilizar Dreamweaver para crear formularios, tablas y otros componentes de páginas HTML. Sin embargo, Dreamweaver va más allá de los editores HTML estándar al utilizar HTML dinámico (DHTML) para añadir cierta funcionalidad como las líneas de tiempo para animaciones y el posicionamiento absoluto de contenido. Al igual que muchos editores HTML, Dreamweaver puede hacer la mayor parte de la tareas de codificación detrás de la interfaz; el usuario no necesita saber la forma en que DHTML crea páginas Web poderosas. La figura 13B.8 muestra el entorno de desarrollo Dreamweaver.
- » **Flash** de Macromedia es una herramienta de desarrollo para crear páginas Web muy sofisticadas, las cuales pueden incluir imágenes en movimiento, animaciones, sonido e interactividad. La figura 13B.9 muestra un ejemplo simple de Flash en acción.
- » **Director** de Macromedia es un entorno de autoría multimedia con todas las características que forma parte de la suite de programas Macromedia Shockwave Studio. Director ofrece a los programadores multimedia y desarrolladores de la Web la capacidad de crear componentes tridimensionales e interactivos, utilizando video de movimiento pleno, animaciones, herramientas de navegación, audio

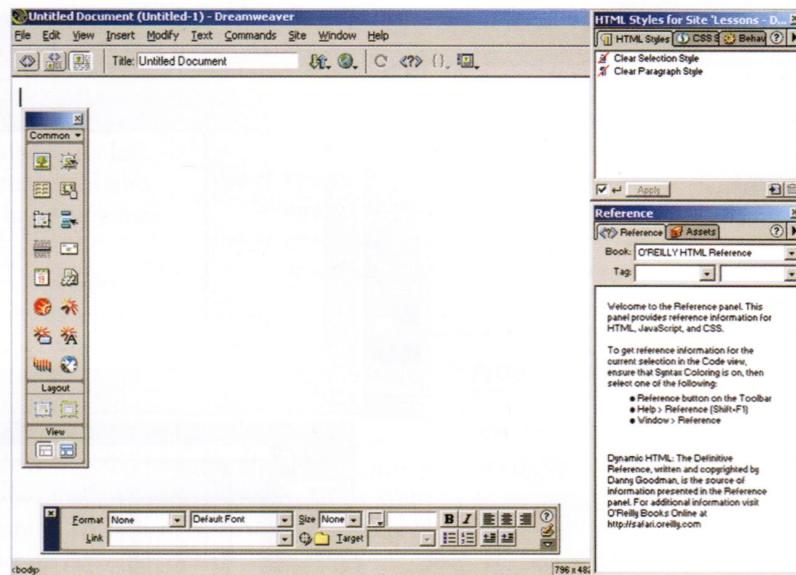


FIGURA 13B.8

El ambiente de desarrollo Dreamweaver.

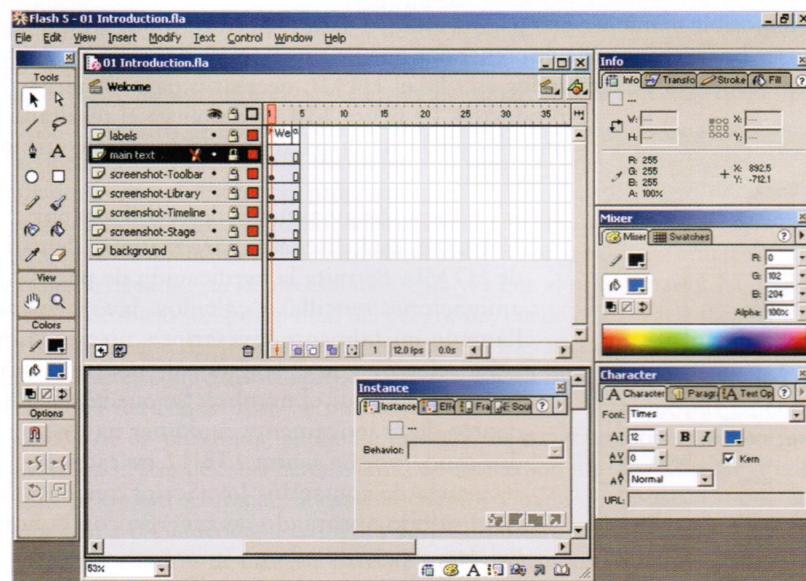
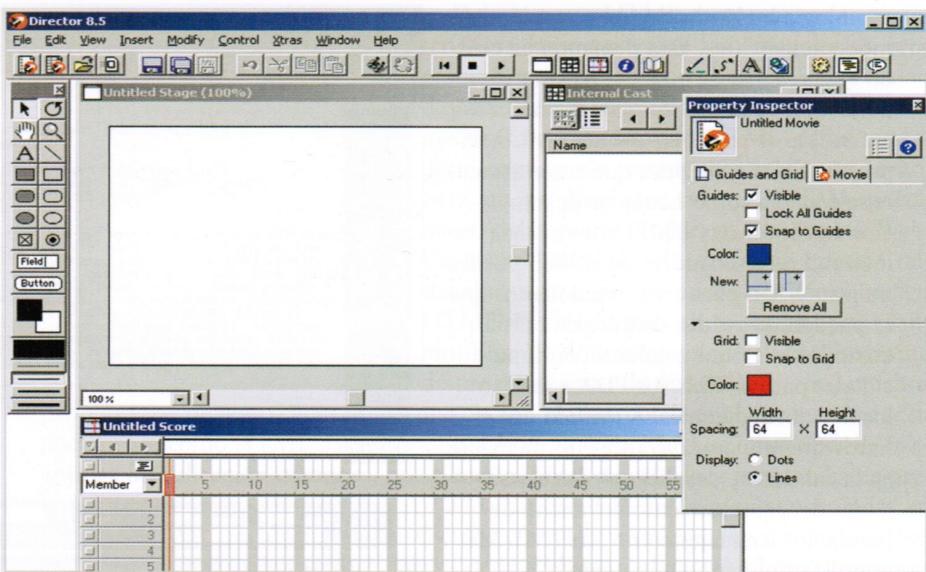


FIGURA 13B.9

La herramienta de creación Flash 5 en uso.

FIGURA 13B.10

Macromedia Director.



y muchas cosas más. Director se utiliza comúnmente para crear herramientas de capacitación en línea con gran riqueza gráfica y demostraciones de productos que se pueden ver en un disco duro, CD-ROM o Internet. La figura 13B.10 muestra a Director en acción.



Norton EN LÍNEA

Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre lenguajes de secuencia de comandos.

Este sitio web es una colección de artículos y ejemplos de lenguajes de secuencia de comandos. Los artículos cubren temas como la programación de la Web, la programación de aplicaciones de escritorio y la programación de dispositivos. Los ejemplos incluyen scripts de Perl, Python y Bash, así como aplicaciones de Java y C/C++.

FIGURA 13B.11

Una respuesta en secuencia de comandos para buscar *caffeine*. Observe la extensión .cgi en la página Web; esto significa que es un script de Perl.

A screenshot of a Microsoft Internet Explorer browser window. The address bar shows the URL: http://www.thinkgeek.com/brain/wheresit.cgi?i=caffeine&bsn=name&so=asc&s=25. The page title is 'Catalog Search'. On the left, there's a sidebar with links like 'T-shirts', 'Other Apparel', 'Cube Goodies', 'Gadgets', 'Computing', 'PC Mods', 'Caffeine', 'Electronics', 'Books', 'What's New', 'Clearance', 'Gift Certificates', 'Geek Points', and 'Site Index'. A 'WE'RE HIRING!' button is also visible. The main content area has a 'Catalog Search' form with fields for 'Search For:' (containing 'Caffeine'), 'Sort By' (set to 'name'), 'Sort Order' (set to 'ascending'), and 'Per Page' (set to '25'). Below the form, it says '33 Search Results'. A table lists products: 1. Caffeine (A ThinkGeek Classic, a simple shirt with a great molecule on it...), Tshirts > Generic Geek, \$14.99, checked, More Info. 2. Caffeine Babydoll Tee (A great lookin' fitted ladies t-shirt with the famous caffeine molecule on it!), Tshirts > Ladies, \$17.99, checked, More Info. 3. Caffeine Candy Sampler, v3.0 (Great divides...), Candy, \$19.99, checked, More Info.

Lenguajes de secuencias de comandos

HTML es adecuado para crear documentos que son visualmente impresionantes en la Web. Sin embargo, HTML es una tecnología **estática**. Esto significa que una vez que una página Web ha sido creada, no cambia sino hasta que alguien edita el código HTML. Esta situación es adecuada para documentos que no cambian casi nunca. Pero piense en una página Web que despliega la temperatura actual de Pittsburg, Pennsylvania. Esta página no puede ser estática; la temperatura cambia demasiado a menudo. Los lenguajes de secuencias de comandos de la Web satisfacen esta necesidad.

Existen distintos lenguajes de secuencias de comandos para la Web. La principal característica de estos lenguajes es su capacidad de crear una página Web dinámica. Las páginas **dinámicas** pueden cambiar de acuerdo con la información que proporciona el usuario. Un ejemplo común son las tiendas en línea. El cliente selecciona los tipos deseados de productos y la página Web se despliega. Para la mayoría de las tiendas, sería imposible mantener una página estática para todos los productos. En lugar de eso, se escribe una secuencia de comandos que lee una base de datos de productos. Luego, esta secuencia de comandos escribe el HTML necesario para desplegar los productos. La figura 13B.11 despliega el resultado de una búsqueda de la palabra caféína en www.thinkgeek.com.

» **JavaScript**, originalmente desarrollado por Netscape, está diseñado para trabajar dentro de HTML. Permite la verificación de páginas, animaciones sencillas y cálculos. JavaScript fue llamado inicialmente Livescript y no tiene ninguna relación con el lenguaje de programación Java excepto por el nombre. Se puede ejecutar dentro de prácticamente cualquier navegador moderno. La figura 13B.12 muestra una secuencia de comandos JavaScript que despliega el saludo apropiado de acuerdo con la hora del día.

- » Las **Páginas de Active Server** (**ASP**, por sus siglas en inglés) son el producto de Microsoft en el campo de la secuencia de comandos para la Web. ASP se basa en Visual Basic y es particularmente bueno para acceder a bases de datos de Microsoft. Esto hace que sea un buen candidato para las tiendas en línea. ASP sólo puede funcionar cuando el sitio Web está alojado en un servidor Windows. La versión más actual es **ASP.NET**.
- » Lenguaje práctico de extracción e informes (**Perl**, por sus siglas en inglés). Perl es uno de los primeros lenguajes de secuencias de comandos que se originaron en los sistemas UNIX como una forma de automatizar tareas administrativas. Ha sufrido una metamorfosis para convertirse en un lenguaje de secuencias de comandos de la Web. Perl, un lenguaje de código abierto, se encuentra en la mayoría de los proveedores de la Web basados en UNIX/Linux y también la mayor parte de los servidores Windows. Debido a que los sitios Web que utilizan Perl pueden estar alojados en ambas plataformas, Perl es un buen lenguaje que debe conocer un desarrollador de la Web.
- » El **Preprocesador de hipertexto** (**PHP**, por sus siglas en inglés) es un lenguaje de secuencias de comandos muy popular. Se ejecuta en servidores UNIX/Linux o Windows. PHP es adecuado especialmente para leer bases de datos como las de Oracle y MySQL. El compilador y el código se ofrecen al público en general como software de código abierto, lo cual hace que su uso sea gratuito. PHP se ofrece en la mayor parte de los sitios de alojamiento de la Web. Al igual que Perl, PHP es un buen lenguaje que un desarrollador de la Web debe conocer.

El ciclo de vida del desarrollo de sistemas en la programación

Los programas son los bloques de construcción de los sistemas de información. Cuando crean productos de software, los programadores siguen un proceso (o ciclo de vida) que es similar al ciclo de vida de los sistemas de información completos. El ciclo de vida de desarrollo de sistemas (SDLC) se detalla en el capítulo 12. Aquí discutiremos el ciclo de vida del desarrollo de software que es parecido.

- » **Fase 1: Análisis de las necesidades.** El análisis de las necesidades es la etapa en donde se identifica y entiende una necesidad o problema. En esta primera etapa, el programador revisa el diseño del programa para ver lo que el usuario necesita para crear una interfaz y punto de inicio, además de lo que el usuario necesita que el programa realice. Normalmente, el usuario final debe tener mucha información útil para la etapa del análisis de las necesidades. Una vez que el programador ha determinado el punto de inicio y el punto final del programa, puede comenzar a diseñar el código.
- » **Fase 2: Diseño del programa.** El diseño del programa es la etapa en la cual los programadores comienzan a aproximarse a la lógica que utilizarán cuando comience la creación del código real. Se pueden utilizar muchas herramientas en el proceso de diseño de un programa, aunque a menudo los programadores utilizan pizarrones y servilletas. Tres de estas herramientas de diseño son los diagramas IPO (para la programación estructurada), líneas con círculos y mensajes (programación orientada a objetos) y pseudocódigo. La figura 13B.13 muestra un conjunto simple de líneas de objetos y cuadros de mensaje parecidos a los que un programador podría desarrollar cuando diseña un programa orientado a objetos.

```

greeting.txt - Notepad
File Edit Format View Help
<script language="Javascript">
<!--
today = new Date();
if((today.getHours() >= 0) && (today.getHours() <12))
{
    document.write("Good morning from Peter Norton! ");
}
else if ((today.getHours() >= 12) && (today.getHours() <17))
{
    document.write("Good afternoon from Peter Norton! ");
}
else if ((today.getHours() >= 17) && (today.getHours() <=23))
{
    document.write("Good evening from Peter Norton! ");
}
-->
</script>

```

FIGURA 13B.12

Los cambios al saludo con base en la hora del día.

AUTOREVALUACIÓN ::

Encierre en un círculo la respuesta a cada pregunta.

1. Todos los lenguajes de programación requieren que los usuarios sigan ciertas reglas de _____.
 - a. estilo
 - b. sintaxis
 - c. gramática
2. El proceso de crear código objeto de un sistema para que trabaje en otro sistema se conoce como _____.
 - a. transportar
 - b. diseñar
 - c. desarrollar
3. Dreamweaver va más allá de los editores HTML estándar al utilizar _____.
 - a. Perl dinámico
 - b. XML dinámico
 - c. HTML dinámico

Norton
EN LÍNEA



Visite el sitio <http://www.mhhe.com/peternorton> para obtener más información sobre SDLC.



La inteligencia artificial (AI, por sus siglas en inglés) se puede definir como un programa o máquina que puede resolver problemas o reconocer patrones. Una definición más "pura" de AI podría ser una computadora o programa que pueda engañar a un humano haciéndolo pensar que está tratando con otro humano. Este tipo de computadora podría aprender y razonar; por tanto, otra definición de inteligencia artificial podría ser una computadora que puede aprender y razonar.

El software de inteligencia artificial se utiliza en muchas aplicaciones del mundo real, desde determinar si los bancos deben conceder préstamos hasta el reconocimiento de voz y los sistemas de guía para misiles. Incluso las aplicaciones como procesadores de texto o correo electrónico utilizan conceptos de AI. Por ejemplo, el revisor ortográfico de un procesador de texto intenta entender y corregir un concepto de lenguaje que muchos usuarios no se los pueden explicar completamente. Sin importar la tarea en cuestión, la inteligencia artificial se utiliza en dos áreas principales:

» **Solución de problemas.** En la solución de problemas, el programa de inteligencia artificial debe revisar un problema o colección de datos y determinar lo que debe hacer a continuación. Por ejemplo, un banco puede uti-

lizar un sistema de inteligencia artificial para que revise su historial crediticio y estilo de vida antes de decidir si debe o no prestarle el dinero. Este tipo de sistemas se conoce como sistema experto.

» **Reconocimiento de patrones.** En el reconocimiento de patrones, el programa inteligencia artificial debe buscar ocurrencias repetidas o conocidas de datos. Entre los ejemplos se incluye la visión artificial y el reconocimiento de voz.

Desde luego, muchos programas de inteligencia artificial combinan elementos de ambas áreas para resolver un problema. Por ejemplo, una herramienta de compresión de datos debe buscar patrones repetidos en los datos y luego decidir la forma de volver a escribir los datos para eliminar duplicados.

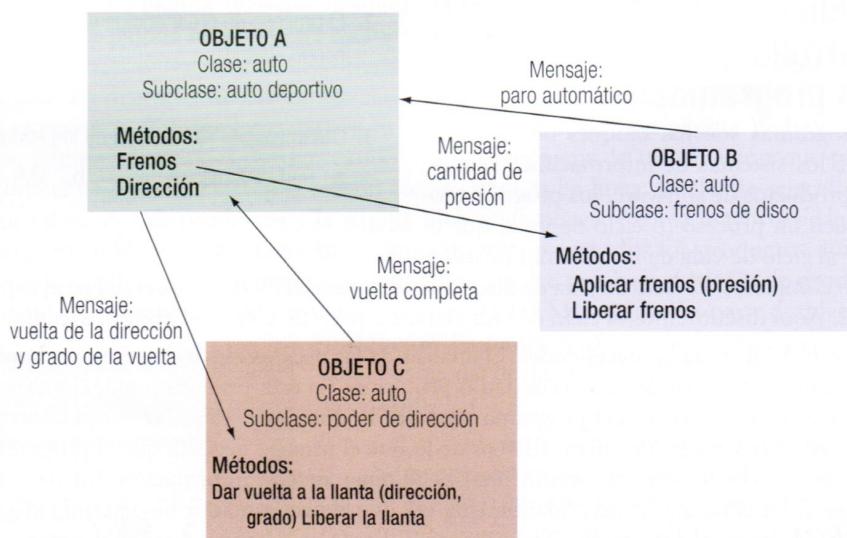
Algunos ejemplos de técnicas AI

La inteligencia artificial se puede aplicar de muchas formas distintas dependiendo del problema que se debe resolver y los recursos disponibles. Entre algunas de las técnicas comunes se incluyen las siguientes:

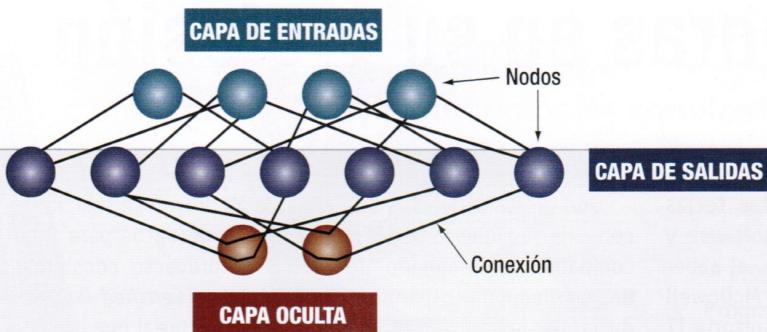
» **Árboles de decisiones.** Estas guías de software son simplemente mapas que le indican a la computadora lo

FIGURA 13B.13

En la fase de diseño, los programadores orientados a objetos usan objetos y cuadros de mensajes para diseñar sus programas.



» **Fase 3: Desarrollo.** El desarrollo (también conocido como codificación) se relaciona con la escritura y pruebas del código fuente. La fase de desarrollo del software es parecida a la fase de desarrollo del ciclo de vida de los sistemas, pero en lugar de determinar el diseño general del sistema, el programador escribe el código que implementa los requerimientos del usuario. El programador puede escribir el código fuente en un editor de texto y luego compilar el código, o puede utilizar un editor visual y crear una imagen de la aplicación antes de compilar el código.



Una red neurológica representada para crear una inteligencia artificial.

que debe hacer a continuación basándose en cada decisión que toma. Cada decisión lleva a una nueva rama con nuevas decisiones y consecuencias.

- » **Sistemas basados en reglas.** Estos sistemas trabajan al seguir un conjunto de reglas determinadas por el programador. Siempre y cuando el programador haya anticipado todas las circunstancias posibles que el programa podría encontrar, podrá resolver cualquier problema.
- » **Retroalimentación.** Esta técnica se utiliza para modificar programas. Básicamente, un sistema de retroalimentación supervisa los resultados de una solución para ver si la solución ha funcionado o en qué áreas ha fallado.
- » **Sistemas de bases de conocimiento.** Estos sistemas son parecidos al sistema basado en reglas pero utiliza la

retroalimentación para aprender de sus errores. Como resultado, los sistemas basados en conocimientos pueden aprender realmente a resolver problemas nuevos.

- » **Heurística.** Esta técnica de software es como una receta para un enfoque y solución de problemas en lugar del algoritmo que resuelve un problema específico.

Creación de un cerebro artificial

Para crear una inteligencia artificial real, los científicos podrían intentar construir un cerebro artificial llamado red neurológica. El cerebro humano consiste en billones e incluso trillones de neuronas cada una con un millón de conexiones a otras neuronas. Los científicos han identificado cientos de tipos distintos de neuronas y más de cincuenta patrones diferentes de conexiones entre neuronas. Este nivel de complejidad simplemente está fuera del alcance de cualquier computadora que exista en la actualidad. Incluso las computadoras paralelas más poderosas con decenas de miles de procesadores no se acercan a la cantidad o variedad de conexiones que existen en el cerebro humano.

La mayor parte del trabajo que se requiere para terminar un programa se ocupa en esta fase utilizando los lenguajes de programación de los que ha aprendido anteriormente. A pesar de sus mejores esfuerzos, los programadores inevitablemente crean errores en sus programas. Existen dos tipos principales de errores: errores de sintaxis y errores lógicos. Los **errores de sintaxis** violan las reglas del lenguaje de programación. Encontrar errores de sintaxis es relativamente sencillo debido a que el compilador o intérprete los señalará para el programador. Los **errores lógicos**, errores reales en el algoritmo, son más difíciles de encontrar y probablemente no aparecerán sino hasta después de semanas o meses después de que el programa ha sido implementado. El proceso de identificar y eliminar estos errores se conoce como depuración. La figura 13B.14 muestra una lista de errores de sintaxis encontrados por un compilador C++. Puede ver por su naturaleza oculta que el programador debe tener algún tipo de conocimientos especiales para entender los errores y repararlos.

```

Output
error C2065: 'The' : undeclared identifier
error C2146: syntax error : missing ';' before identifier 'average'
error C2146: syntax error : missing ';' before identifier 'age'
error C2001: newline in constant
error C2065: 'age' : undeclared identifier
error C2146: syntax error : missing ';' before identifier 'is'
error C2065: 'is' : undeclared identifier
error C2143: syntax error : missing ';' before 'string'

```

FIGURA 13B.14

Los compiladores ayudan a los programadores a localizar y arreglar errores de sintaxis.

Las computadoras en su profesión

Profesiones relacionadas con programación

En el mundo de Douglas McDowell, unas cuantas teclas harán que inicie una aplicación o programa de software y provoque que se iluminen los ojos de un cliente, al saber que sus problemas tecnológicos están resueltos. McDowell es un consultor ejecutivo de la compañía de consultoría IT Intellinet de Atlanta y trabaja con los clientes utilizando su experiencia e inteligencia en programación y empresas para crear soluciones tecnológicas.

“Como programador, resuelvo problemas que ofrecen un valor alto a las compañías de manera casi instantánea”, afirma McDowell, un cocinero profesional que se graduó en Wheaton College y obtuvo una maestría en tecnologías de información. Cambió de profesión después de haber tenido una lesión en la espalda cuando trabajaba en restaurantes y se dedicó a resolver problemas analíticos y la evaluación de problemas empresariales para lo cual sirve una profesión en programación.

McDowell ocupa diez horas del día manejando el lado administrativo del proyecto de desarrollo de bases de datos y trabajando con los clientes de Intellinet. Primero realiza un análisis empresarial para saber con qué tecnología cuenta la compañía y la forma en que puede crear e integrar sistemas de soluciones nuevas para ayudar a que la infraestructura funcione de una manera más productiva.

Uno de los proyectos recientes de McDowell se relaciona con una red que supervisa un almacén de datos para una compañía de televisión por cable. El proyecto consiguió un reconocimiento para Intellinet llamado *Certified Partner Award* por la solución de inteligencia empresarial que utilizó tecnologías de Microsoft.

“Es cuestión de entender lo que necesitan de una aplicación o solución que debo crear y comprender qué problemas necesita resolver —dice McDowell cuyas certificaciones incluyen arquitectura y desarrollo .NET; Windows, Windows Exchange, Seguridad y sistemas; inteligencia empresarial enfocada a SQL Server; MCP, y MCSE—. Entonces, termino la solución de acuerdo con esos requerimientos.”

El crecimiento en la economía y la popularidad de los juegos gráficos e Internet están provocando un aumento grande en la programación. No sólo existen trabajos disponibles en programación sino que también se están volviendo más interesantes. Los empleos en programación tienden a agruparse dentro de las categorías generales siguientes:

» **Programador científico.** Estos programadores utilizan un conocimiento especializado de ciencias e ingeniería para desarrollar programas de alta tecnología. Los programadores científicos trabajan en campos como la ingeniería aeroespacial, meteorología, oceanografía y astronomía.

- » **Fase 4: Implementación.** La implementación se relaciona con la instalación del software y con permitir que los usuarios lo prueben. Este paso normalmente incluye una gran cantidad de documentación, tanto dentro del código como en la forma de manuales para los usuarios. Muchos programadores también le dirán que realizan la mayor parte de la depuración en esta etapa. Es realmente en la etapa de implementación cuando cualquier error de concepción que haya tenido el programador en el código se encuentra y repara.
- » **Fase 5: Mantenimiento.** El mantenimiento comienza tan pronto como el programa ha sido instalado. El trabajo en los productos continúa por distintas razones. Es probable que algunos errores menores no se hayan reparado en el momento en que el programa fue terminado. También es probable que los programadores quieran añadir funciones nuevas importantes en respuesta a las demandas del mercado o las solicitudes de los usuarios. Ésta es la fase más larga de ciclo de vida del desarrollo de programas, y algunas veces puede durar muchos años.



- » **Programador empresarial.** Prácticamente todas las empresas necesitan computadoras y todas las computadoras necesitan programas. Por tanto, existe una gran demanda de programadores que combinan conocimientos de programación con conocimientos de operaciones empresariales.
- » **Programador de sistemas operativos.** Desde luego, todas las computadoras y máquinas controladas por computadoras necesitan un sistema operativo y algún tipo de control programado. En algunos casos, por ejemplo, las PC, los programadores desarrollan sistemas operativos como DOS, Windows o UNIX.
- » **Programador de entretenimiento.** Los programadores de juegos desarrollan software educativo y juegos de

video. En todos los casos, los programadores deben combinar conocimientos robustos de la teoría del juego con la programación de imágenes y multimedia.

» **Programador de la Web.** La World Wide Web ha creado un campo completamente nuevo en la programación. Naturalmente, la existencia de la Web ha generado una demanda para programadores HTML y Java. Sin embargo, la Web también provocó una demanda de programadores que puedan desarrollar herramientas que permitan a la Web ofrecer contenido multimedia. Debido a que la Web aún se encuentra en su infancia, nadie puede predecir qué contenido de la Web se desarrollará en el futuro, cuál será popular y lo que fracasará.

La Agencia de Estadísticas Laborales espera que el campo de la programación crezca en promedio tan rápido como todas las demás ocupaciones hasta el año 2010, con empleos para programadores de sistemas de aplicaciones con mayor disponibilidad en las compañías de servicios de procesamiento de datos, compañías de software y empresas de consultoría en computación. Las ganancias anuales promedio de los programadores de computadoras fueron de 57 590 dólares en el 2000, del que 50 por ciento las personas ganaron entre 44 850 y 74 500 dólares al año.