Chapter 1 Introduction to Computers, the Internet and the Web

C How to Program, 7/e

OBJECTIVES

In this chapter, you'll learn:

- Basic computer concepts.
- The different types of programming languages.
- The history of the C programming language.
- The purpose of the C Standard Library.
- The elements of a typical C program development environment.
- To test-drive a C application in Windows, Linux and Mac OS X.
- Some basics of the Internet and the World Wide Web.

1.1 Introduction

- The core of the book emphasizes effective software engineering through the proven methodologies of structured programming in C and object-oriented programming in C++.
- All of these example programs may be downloaded from our website `www.deitel.com/books/chtp7/`.
- You'll learn how to command computers to perform tasks.
  - **Software** (i.e., the instructions you write to command computers to perform **actions** and make **decisions**) controls computers (often referred to as **hardware**).

1.2 Computers and the Internet in Industry and Research

- Figure 1.1 provides a few examples of how computers are used in industry and research.

Name	Description
Electronic health records	These might include a patient's medical history, prescriptions, immunizations, lab results, allergies, insurance information and more. Making this information available to health care providers across a secure network improves patient care, reduces the probability of error and increases overall efficiency of the health care system.
Human Genome Project	The Human Genome Project was founded to identify and analyze the 20,000+ genes in human DNA. The project used computer programs to analyze complex genetic data, determine the sequences of the billions of chemical base pairs that make up human DNA and store the information in databases which have been made available over the Internet to researchers in many fields.
AMBER™ Alert	The AMBER (America's Missing: Broadcast Emergency Response) Alert System is used to find abducted children. Law enforcement notifies TV and radio broadcasters and state transportation officials, who then broadcast alerts on TV, radio, computerized highway signs, the Internet and wireless devices. AMBER Alert recently partnered with Facebook, whose users can "Like" AMBER Alert pages by location to receive alerts in their news feeds.

Fig. 1.1 | A few uses for computers. (Part 1 of 7.)

Name	Description
World Community Grid	People worldwide can donate their unused computer processing power by installing a free secure software program that allows the World Community Grid (`www.worldcommunitygrid.org`) to harness unused capacity. This computing power, accessed over the Internet, is used in place of expensive supercomputers to conduct scientific research projects that are making a difference—providing clean water to third-world countries, fighting cancer, growing more nutritious rice for regions fighting hunger and more.
Medical imaging	X-ray computed tomography (CT) scans, also called CAT (computerized axial tomography) scans, take X-rays of the body from hundreds of different angles. Computers are used to adjust the intensity of the X-rays, optimizing the scan for each type of tissue, then to combine all of the information to create a 3D image. MRI scanners use a technique called magnetic resonance imaging, also to produce internal images non-invasively.
One Laptop Per Child (OLPC)	One Laptop Per Child (`one.laptop.org`) is providing low-power, inexpensive, Internet-enabled laptops to children in third-world countries—enabling learning and reducing the digital divide.

Fig. 1.1 | A few uses for computers. (Part 2 of 7.)

Name	Description
Cloud computing	Cloud computing allows you to use software, hardware and information stored in the "cloud"—i.e., accessed on remote computers via the Internet and available on demand—rather than having it stored on your personal computer. These services allow you to increase or decrease resources to meet your needs at any given time, so they can be more cost effective than purchasing expensive hardware to ensure that you have enough storage and processing power to meet your needs at their peak levels. Business applications often are expensive, and require significant hardware to run them and knowledgeable support staff to ensure that they're running properly and securely. Using cloud computing services shifts the burden of managing these applications from the business to the service provider, saving businesses money.

Fig. 1.1 | A few uses for computers. (Part 3 of 7.)

Name	Description
GPS	Global Positioning System (GPS) devices use a network of satellites to retrieve location-based information. Multiple satellites send time-stamped signals to the GPS device, which calculates the distance to each satellite based on the time the signal left the satellite and the time the signal arrived. This information is used to determine the exact location of the device. GPS devices can provide step-by-step directions and help you locate nearby businesses (restaurants, gas stations, etc.) and points of interest. GPS is used in numerous location-based Internet services such as check-in apps to help you find your friends (e.g., Foursquare and Facebook), exercise apps such as RunKeeper that track the time, distance and average speed of your outdoor jog, dating apps that help you find a match nearby and apps that dynamically update changing traffic conditions.

Fig. 1.1 | A few uses for computers. (Part 4 of 7.)

Name	Description
Robots	Robots can be used for day-to-day tasks (e.g., iRobot's Roomba vacuuming robot), entertainment (e.g., robotic pets), military combat, deep sea and space exploration (e.g., NASA's Mars rover) and more. RoboEarth (`www.roboearth.org`) is "a World Wide Web for robots." It allows robots to learn from each other by sharing information and thus improving their abilities to perform tasks, navigate, recognize objects and more.
E-mail, Instant Messaging, Video Chat and FTP	Internet-based servers support all of your online messaging. E-mail messages go through a mail server that also stores the messages. Instant Messaging (IM) and Video Chat apps, such as AIM, Skype, Yahoo! Messenger and others allow you to communicate with others in real time by sending your messages and live video through servers. FTP (file transfer protocol) allows you to exchange files between multiple computers (e.g., a client computer such as your desktop and a file server) over the Internet.

Fig. 1.1 | A few uses for computers. (Part 5 of 7.)

Name	Description
Internet TV	Internet TV set-top boxes (such as Apple TV, Google TV and TiVo) allow you to access an enormous amount of content on demand, such as games, news, movies, television shows and more, and they help ensure that the content is streamed to your TV smoothly.

Fig. 1.1 | A few uses for computers. (Part 6 of 7.)

Name	Description
Game programming	Analysts expect global video game revenues to reach $91 billion by 2015 (`www.vg247.com/2009/06/23/global-industry-analysts-predicts-gaming-market-to-reach-91-billion-by-2015/`). The most sophisticated games can cost as much as $100 million to develop. Activision's *Call of Duty: Black Ops*—one of the best-selling games of all time—earned $360 million in just one day (`www.forbes.com/sites/insertcoin/2011/03/11/call-of-duty-black-ops-now-the-best-selling-video-game-of-all-time/`)! Online *social gaming*, which enables users worldwide to compete with one another over the Internet, is growing rapidly. Zynga—creator of popular online games such as *Farmville* and *Mafia Wars*—was founded in 2007 and already has over 200 million monthly users. To accommodate the growth in traffic, Zynga is adding nearly 1,000 servers each week (`techcrunch.com/2010/09/22/zynga-moves-1-petabyte-of-data-daily-adds-1000-servers-a-week/`)!

Fig. 1.1 | A few uses for computers. (Part 7 of 7.)

1.3 Computers: Hardware and Software

- In use today are more than a billion general-purpose computers, and billions more *embedded* computers are used in cell phones, smartphones, tablet computers, home appliances, automobiles and more.
- Computers can perform computations and make logical decisions phenomenally faster than human beings can.
- Today's personal computers can perform billions of calculations in one second—more than a human can perform in a lifetime.
- *Supercomputers* are already performing *thousands of trillions* (*quadrillions*) of instructions per second!
- Computers process data under the control of sets of instructions called computer programs.
- These programs guide the computer through ordered actions specified by people called computer programmers.

1.3 Computers: Hardware and Software (Cont.)

- The programs that run on a computer are referred to as software.
- You'll learn key programming methodology that are enhancing programmer productivity, thereby reducing software-development costs—*structured programming* (in C) and *object-oriented programming* in C++.
- A computer consists of various devices referred to as hardware
  - (e.g., the keyboard, screen, mouse, hard disks, memory, DVD drives and processing units).
- Computing costs are *dropping dramatically*, owing to rapid developments in hardware and software technologies.

1.3 Computers: Hardware and Software (Cont.)

- Computers that might have filled large rooms and cost millions of dollars decades ago are now inscribed on silicon chips smaller than a fingernail, costing perhaps a few dollars each.
- Silicon-chip technology has made computing so economical that more than computers have become a commodity.

1.3.1 Moore's Law

- For many decades, hardware costs have fallen rapidly.
- Every year or two, the capacities of computers have approximately *doubled* inexpensively.
- This remarkable trend often is called Moore's Law, named for the person who identified it, Gordon Moore, co-founder of Intel.

1.2 Computers: Hardware and Software (Cont.)

- Moore's Law and related observations apply especially to the amount of memory that computers have for programs, the amount of secondary storage (such as disk storage) they have to hold programs and data over longer periods of time, and their processor speeds—the speeds at which computers execute their programs (i.e., do their work).
- Similar growth has occurred in the communications field.

1.2 Computers: Hardware and Software (Cont.)

- Costs have plummeted as enormous demand for communications bandwidth (i.e., information-carrying capacity) has attracted intense competition.
- Such phenomenal improvement is fostering the *Information Revolution.*

1.3.2 Computer Organization

- Regardless of differences in physical appearance, computers can be envisioned as divided into various logical units or sections (Fig. 1.2).

Logical unit	Description
Input unit	This "receiving" section obtains information (data and computer programs) from input devices and places it at the disposal of the other units for processing. Most information is entered into computers through keyboards, touch screens and mouse devices. Other forms of input include receiving voice commands, scanning images and barcodes, reading from secondary storage devices (such as hard drives, DVD drives, Blu-ray Disc™ drives and USB flash drives—also called "thumb drives" or "memory sticks"), receiving video from a webcam and having your computer receive information from the Internet (such as when you download videos from YouTube™ or e-books from Amazon). Newer forms of input include position data from a GPS device, and motion and orientation information from an accelerometer in a smartphone or game controller (such as Microsoft® Kinect™, Wii™ Remote and PlayStation® Move).

Fig. 1.2 | Logical units of a computer. (Part 1 of 4.)

Logical unit	Description
Output unit	This "shipping" section takes information that the computer has processed and places it on various output devices to make it available for use outside the computer. Most information that's output from computers today is displayed on screens, printed on paper, played as audio or video on PCs and media players (such as Apple's popular iPods) and giant screens in sports stadiums, transmitted over the Internet or used to control other devices, such as robots and "intelligent" appliances.
Memory unit	This rapid-access, relatively low-capacity "warehouse" section retains information that has been entered through the input unit, making it immediately available for processing when needed. The memory unit also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is *volatile*—it's typically lost when the computer's power is turned off. The memory unit is often called either memory or primary memory. Typical main memories on desktop and notebook computers contain between 1 and 8 GB (GB stands for gigabytes; a gigabyte is approximately one billion bytes).

Fig. 1.2 | Logical units of a computer. (Part 2 of 4.)

Logical unit	Description
Arithmetic and logic unit (ALU)	This "manufacturing" section performs *calculations*, such as addition, subtraction, multiplication and division. It also contains the *decision* mechanisms that allow the computer, for example, to compare two items from the memory unit to determine whether they're equal. In today's systems, the ALU is usually implemented as part of the next logical unit, the CPU.
Central processing unit (CPU)	This "administrative" section coordinates and supervises the operation of the other sections. The CPU tells the input unit when information should be read into the memory unit, tells the ALU when information from the memory unit should be used in calculations and tells the output unit when to send information from the memory unit to certain output devices. Many of today's computers have multiple CPUs and, hence, can perform many operations simultaneously. A multi-core processor implements multiple processors on a single integrated-circuit chip—a *dual-core processor* has two CPUs and a *quad-core processor* has four CPUs. Today's desktop computers have processors that can execute billions of instructions per second.

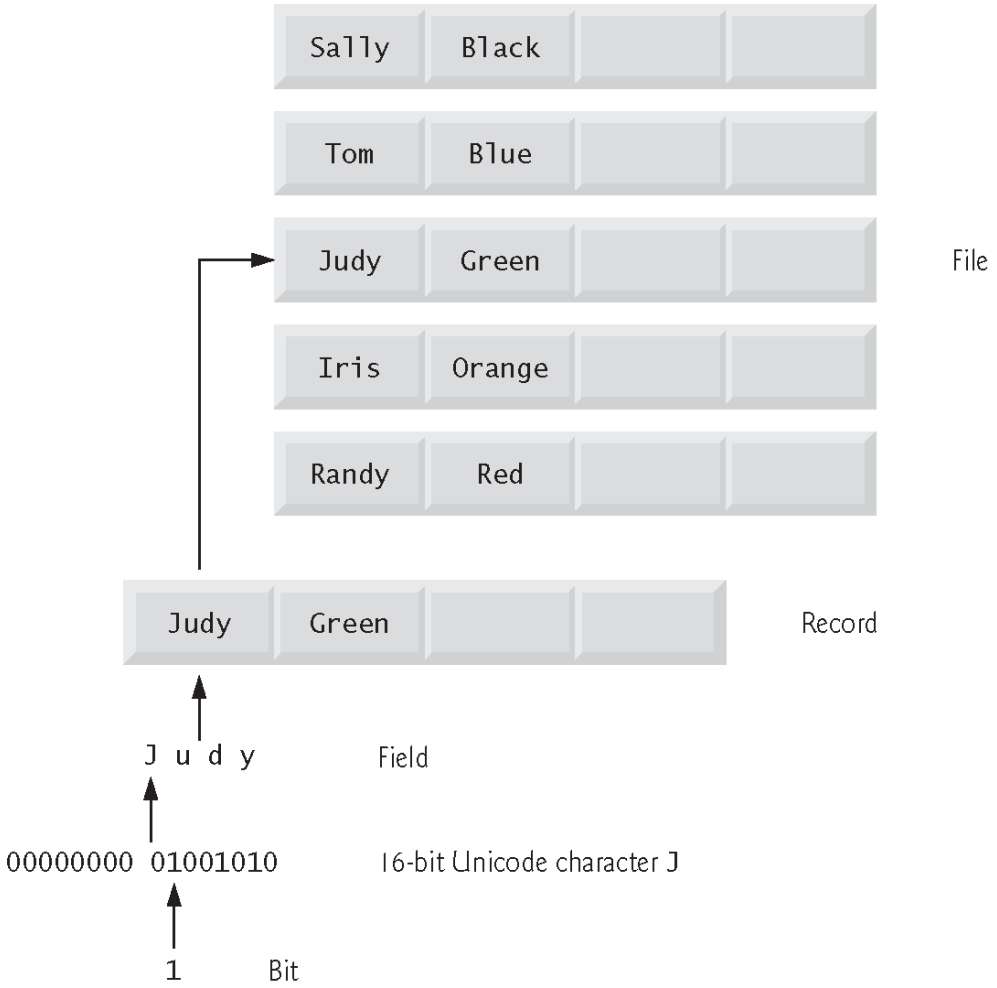
Fig. 1.2 | Logical units of a computer. (Part 3 of 4.)

Logical unit	Description
Secondary storage unit	This is the long-term, high-capacity "warehousing" section. Programs or data not actively being used by the other units normally are placed on secondary storage devices (e.g., your *hard drive*) until they're again needed, possibly hours, days, months or even years later. Information on secondary storage devices is *persistent*—it's preserved even when the computer's power is turned off. Secondary storage information takes much longer to access than information in primary memory, but the cost per unit of secondary storage is much less than that of primary memory. Examples of secondary storage devices include CD drives, DVD drives and flash drives, some of which can hold up to 512 GB. Typical hard drives on desktop and notebook computers can hold up to 2 TB (TB stands for terabytes; a terabyte is approximately one trillion bytes).

Fig. 1.2 | Logical units of a computer. (Part 4 of 4.)

1.4 Data Hierarchy

- Data items processed by computers form a data hierarchy that becomes larger and more complex in structure as we progress from bits to characters to fields, and so on.
- Figure 1.3 illustrates a portion of the data hierarchy.
- Figure 1.4 summarizes the data hierarchy's levels.



Level	Description
Bits	The smallest data item in a computer can assume the value `0` or the value `1`. Such a data item is called a bit (short for "binary digit"—a digit that can assume one of two values). It's remarkable that the impressive functions performed by computers involve only the simplest manipulations of `0`s and `1`s—*examining a bit's value*, *setting a bit's value* and *reversing a bit's value* (from `1` to `0` or from `0` to `1`).
Characters	It's tedious for people to work with data in the low-level form of bits. Instead, they prefer to work with *decimal digits* (0–9), *letters* (A–Z and a–z), and *special symbols* (e.g., $, @, %, &, *, (,), –, +, ", :, ? and /). Digits, letters and special symbols are known as characters. The computer's character set is the set of all the characters used to write programs and represent data items. Computers process only `1`s and `0`s, so every character is represented as a pattern of `1`s and `0`s. The Unicode character set contains characters for many of the world's languages. C supports several character sets, including 16-bit Unicode® characters

Fig. 1.4 | Levels of the data hierarchy. (Part 1 of 4.)

Level	Description
Characters (cont.)	that are composed of two bytes, each composed of eight bits. See Appendix B for more information on the ASCII (American Standard Code for Information Interchange) character set—the popular subset of Unicode that represents uppercase and lowercase letters, digits and some common special characters.
Fields	Just as characters are composed of bits, fields are composed of characters or bytes. A field is a group of characters or bytes that conveys meaning. For example, a field consisting of uppercase and lowercase letters could be used to represent a person's name, and a field consisting of decimal digits could represent a person's age.

Fig. 1.4 | Levels of the data hierarchy. (Part 2 of 4.)

Level	Description
Records	Several related fields can be used to compose a record. In a payroll system, for example, the record for an employee might consist of the following fields (possible types for these fields are shown in parentheses): • Employee identification number (a whole number) • Name (a string of characters) • Address (a string of characters) • Hourly pay rate (a number with a decimal point) • Year-to-date earnings (a number with a decimal point) • Amount of taxes withheld (a number with a decimal point) Thus, a record is a group of related fields. In the preceding example, all the fields belong to the same employee. A company might have many employees and a payroll record for each one.

Fig. 1.4 | Levels of the data hierarchy. (Part 3 of 4.)

Level	Description
Files	A file is a group of related records. [*Note:* More generally, a file contains arbitrary data in arbitrary formats. In some operating systems, a file is viewed simply as a *sequence of bytes*—any organization of the bytes in a file, such as organizing the data into records, is a view created by the application programmer.] It's not unusual for an organization to have many files, some containing billions, or even trillions, of characters of information.
Database	A database is an electronic collection of data that's organized for easy access and manipulation. The most popular database model is the relational database in which data is stored in simple *tables*. A table includes *records* and *fields*. For example, a table of students might include first name, last name, major, year, student ID number and grade point average. The data for each student is a record, and the individual pieces of information in each record are the fields. You can search, sort and manipulate the data based on its relationship to multiple tables or databases. For example, a university might use data from the student database in combination with databases of courses, on-campus housing, meal plans, etc.

Fig. 1.4 | Levels of the data hierarchy. (Part 4 of 4.)

1.5 Programming Languages

- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate *translation* steps.
- Any computer can directly understand only its own machine language, defined by its hardware design.
- Machine languages generally consist of numbers (ultimately reduced to 1s and 0s). Such languages are cumbersome for humans.
- Programming in machine language—the numbers that computers could directly understand—was simply too slow and tedious for most programmers.
- Instead, they began using Englishlike abbreviations to represent elementary operations.
- These abbreviations formed the basis of assembly languages.
- *Translator programs* called assemblers were developed to convert assembly-language programs to machine language.

1.5 Types of Programming Languages (cont.)

- Although assembly-language code is clearer to humans, it's incomprehensible to computers until translated to machine language.
- To speed the programming process even further, high-level languages were developed in which single statements could be written to accomplish substantial tasks.
- High-level languages allow you to write instructions that look almost like everyday English and contain commonly used mathematical expressions.
- Translator programs called compilers convert high-level language programs into machine language.
- Interpreter programs were developed to execute high-level language programs directly, although more slowly than compiled programs.
- Scripting languages such as JavaScript and PHP are processed by interpreters.

Performance Tip 1.1

Interpreters have an advantage over compilers in Internet scripting. An interpreted program can begin executing as soon as it's downloaded to the client's machine, without needing to be compiled before it can execute. On the downside, interpreted scripts generally run slower than compiled code.

1.6 The C Programming Language

- C evolved from two previous languages, BCPL and B.
- BCPL was developed in 1967 by Martin Richards as a language for writing operating-systems software and compilers.
- Ken Thompson modeled many features in his B language after their counterparts in BCPL, and in 1970 he used B to create early versions of the UNIX operating system at Bell Laboratories.

1.6 The C Programming Language (Cont.)

- The C language was evolved from B by Dennis Ritchie at Bell Laboratories and was originally implemented in 1972.
- C initially became widely known as the development language of the UNIX operating system.
- Many of todays leading operating systems are written in C and/or C++.
- C is mostly hardware independent.
- With careful design, it's possible to write C programs that are portable to most computers.

1.6 The C Programming Language (Cont.)

Built for Performance

- C is widely used to develop systems that demand performance, such as operating systems, embedded systems, real-time systems and communications systems (Figure 1.5).

Application	Description
Operating systems	C's portability and performance make it desirable for implementing operating systems, such as Linux and portions of Microsoft's Windows and Google's Android. Apple's OS X is built in Objective-C, which was derived from C. We discuss some key popular desktop/notebook operating systems and mobile operating systems in Section 1.12.

Fig. 1.5 | Some popular performance-oriented C applications.

Application	Description
Embedded systems	The vast majority of the microprocessors produced each year are embedded in devices other than general-purpose computers. These embedded systems include navigation systems, smart home appliances, home security systems, smartphones, robots, intelligent traffic intersections and more. C is one of the most popular programming languages for developing embedded systems, which typically need to run as fast as possible and conserve memory. For example, a car's anti-lock brakes must respond immediately to slow or stop the car without skidding; game controllers used for video games should respond instantaneously to prevent any lag between the controller and the action in the game, and to ensure smooth animations.

Fig. 1.5 | Some popular performance-oriented C applications.

Application	Description
Real-time systems	Real-time systems are often used for "mission-critical" applications that require nearly instantaneous response times. For example, an air-traffic-control system must constantly monitor the positions and velocities of the planes and report that information to air-traffic controllers without delay so that they can alert the planes to change course if there's a possibility of a collision.
Communications systems	Communications systems need to route massive amounts of data to their destinations quickly to ensure that things such as audio and video are delivered smoothly and without delay.

Fig. 1.5 | Some popular performance-oriented C applications.

1.6 The C Programming Language (Cont.)

- By the late 1970s, C had evolved into what is now referred to as "traditional C." The publication in 1978 of Kernighan and Ritchie's book, *The C Programming Language*, drew wide attention to the language.

Standardization

- The rapid expansion of C over various types of computers (sometimes called hardware platforms) led to many variations that were similar but often incompatible.
- In 1989, the C standard was approved; this standard was updated in 1999 and is often referred to as C99.

1.6 The C Programming Language (Cont.)

The New C Standard

- The new C standard (referred to as C11) refines and expands the capabilities of C.
- Not all popular C compilers support the new features. Of those that do, most implement only a subset of the new features.

Portability Tip 1.1

Because C is a hardware-independent, widely available language, applications written in C often can run with little or no modification on a range of different computer systems.

1.7 C Standard Library

- As you'll learn in Chapter 5, C programs consist of pieces called functions.
- You can program all the functions you need to form a C program, but most C programmers take advantage of the rich collection of existing functions called the C Standard Library.
- Visit the following website for the C Standard Library documentation:

  `www.dinkumware.com/manuals/#Standard%20C%20Library`
- This textbook encourages a *building-block approach* to creating programs.

1.7 C Standard Library (Cont.)

- Avoid reinventing the wheel.
- Instead, use existing pieces—this is called software reuse.
- When programming in C you'll typically use the following building blocks:
  - C Standard Library functions
  - Functions you create yourself
  - Functions other people (whom you trust) have created and made available to you

1.7 C Standard Library (Cont.)

- The advantage of creating your own functions is that you'll know exactly how they work. You'll be able to examine the C code.
- The disadvantage is the time-consuming effort that goes into designing, developing and debugging new functions.

Performance Tip 1.2

Using Standard C library functions instead of writing your own comparable versions can improve program performance, because these functions are carefully written to perform efficiently.

Portability Tip 1.2

Using Standard C library functions instead of writing your own comparable versions can improve program portability, because these functions are used in virtually all Standard C implementations.

1.8 C++ and Other C-Based Languages

- C++ was developed by Bjarne Stroustrup at Bell Laboratories.
- It has its roots in C, providing a number of features that "spruce up" the C language.
- More important, it provides capabilities for object-oriented programming.
- Objects are essentially reusable software components that model items in the real world.
- Using a modular, object-oriented design and implementation approach can make software development groups more productive.

1.8 C++ and Other C-Based Languages (Cont.)

- Chapters 15-24 present a condensed treatment of C++ selected from our book *C++ How to Program, 8/e*.
- As you study C++, check out the C++ Resource Center at `www.deitel.com/cplusplus/`.
- Figure 1.6 introduces several other popular C-based programming languages.

Programming language	Description
Objective-C	Objective-C is an object-oriented language based on C. It was developed in the early 1980s and later acquired by NeXT, which in turn was acquired by Apple. It has become the key programming language for the Mac OS X operating system and all iOS-based devices (such as iPods, iPhones and iPads).
Visual C#	Microsoft's three primary object-oriented programming languages are Visual Basic, Visual C++ (based on C++) and C# (based on C++ and Java, and developed for integrating the Internet and the web into computer applications).

Fig. 1.6 | Popular C-based programming languages. (Part 1 of 3.)

Programming language	Description
Java	Sun Microsystems in 1991 funded an internal corporate research project which resulted in the C++-based object-oriented programming language called Java. A key goal of Java is to enable the writing of programs that will run on a broad variety of computer systems and computer-controlled devices. This is sometimes called "write once, run anywhere." Java is used to develop large-scale enterprise applications, to enhance the functionality of web servers (the computers that provide the content we see in our web browsers), to provide applications for consumer devices (smartphones, television set-top boxes and more) and for many other purposes.
PHP	PHP—an object-oriented, open-source (see Section 1.12) scripting language based on C and supported by a community of users and developers—is used by many websites including Wikipedia and Facebook. PHP is *platform independent*—implementations exist for all major UNIX, Linux, Mac and Windows operating systems. PHP also supports many databases, including MySQL. Other languages similar in concept to PHP are Perl and Python.

Fig. 1.6 | Popular C-based programming languages. (Part 2 of 3.)

Programming language	Description
JavaScript	JavaScript—developed by Netscape—is the most widely used scripting language. It's primarily used to add programmability to web pages—for example, animations and interactivity with the user. It's provided with all major web browsers.

Fig. 1.6 | Popular C-based programming languages. (Part 3 of 3.)

1.9 Object Technology

- *Objects*, or more precisely the *classes* objects come from, are essentially *reusable* software components.
- Almost any *noun* can be reasonably represented as a software object in terms of *attributes* (e.g., name, color and size) and *behaviors* (e.g., calculating, moving and communicating).
- Software developers are discovering that using a modular, object-oriented design and implementation approach can make software-development groups much more productive than was possible with earlier techniques—object-oriented programs are often easier to understand, correct and modify.

1.9 Object Technology

The Automobile as an Object

- Suppose you want to *drive a car and make it go faster by pressing its accelerator pedal*.
- Before you can drive a car, someone has to *design* it.
- A car typically begins as engineering drawings, similar to the *blueprints* that describe the design of a house.
- These drawings include the design for an accelerator pedal.

1.9 Object Technology (cont.)

- The pedal *hides* from the driver the complex mechanisms that actually make the car go faster, just as the brake pedal hides the mechanisms that slow the car, and the steering wheel *hides* the mechanisms that turn the car.
- This enables people with little or no knowledge of how engines, braking and steering mechanisms work to drive a car easily.
- Before you can drive a car, it must be *built* from the engineering drawings that describe it.
- A completed car has an *actual* accelerator pedal to make the car go faster, but even that's not enough—the car won't accelerate on its own (hopefully!), so the driver must *press* the pedal to accelerate the car.

1.9 Object Technology (cont.)

Methods and Classes

- Performing a task in a program requires a method.
- The method houses the program statements that actually perform its tasks.
- It hides these statements from its user, just as a car's accelerator pedal hides from the driver the mechanisms of making the car go faster.
- In object-oriented programming languages, we create a program unit called a class to house the set of methods that perform the class's tasks.
- For example, a class that represents a bank account might contain one method to *deposit* money to an account, another to *withdraw* money from an account and a third to *inquire* what the account's current balance is.
- A class is similar in concept to a car's engineering drawings, which house the design of an accelerator pedal, steering wheel, and so on.

1.9 Object Technology (cont.)

Instantiation

- Just as someone has to *build a car* from its engineering drawings before you can actually drive a car, you must *build an object* from a class before a program can perform the tasks that the class's methods define.
- The process of doing this is called *instantiation*. An object is then referred to as an instance of its class.

Reuse

- Just as a car's engineering drawings can be *reused* many times to build many cars, you can *reuse* a class many times to build many objects.
- Reuse of existing classes when building new classes and programs saves time and effort.
- Reuse also helps you build more reliable and effective systems, because existing classes and components often have gone through extensive *testing*, *debugging* and *performance tuning*.

1.9 Object Technology (cont.)

Messages and Method Calls

- When you drive a car, pressing its gas pedal sends a *message* to the car to perform a task—that is, to go faster. Similarly, you *send messages to an object*.
- Each message is implemented as a method call that tells a method of the object to perform its task.
- For example, a program might call a particular bank-account object's *deposit* method to increase the account's balance.

1.9 Object Technology (cont.)

Attributes and Instance Variables

- A car, besides having capabilities to accomplish tasks, also has *attributes*, such as its color, its number of doors, the amount of gas in its tank, its current speed and its record of total miles driven (i.e., its odometer reading).
- Like its capabilities, the car's attributes are represented as part of its design in its engineering diagrams (which, for example, include an odometer and a fuel gauge).
- As you drive an actual car, these attributes are carried along with the car.
- Every car maintains its *own* attributes.
- For example, each car knows how much gas is in its own gas tank, but *not* how much is in the tanks of *other* cars.
- An object, similarly, has attributes that it carries along as it's used in a program.

1.9 Object Technology (cont.)

- These attributes are specified as part of the object's class.
- For example, a bank-account object has a *balance attribute* that represents the amount of money in the account.
- Each bank-account object knows the balance in the account it represents, but *not* the balances of the *other* accounts in the bank.
- Attributes are specified by the class's instance variables.

1.9 Object Technology (cont.)

Encapsulation

- Classes encapsulate (i.e., wrap) attributes and methods into objects—an object's attributes and methods are intimately related.
- Objects may communicate with one another, but normally they're not allowed to know how other objects are implemented—implementation details are *hidden* within the objects themselves.
- This information hiding is crucial to good software engineering.

1.9 Object Technology (cont.)

Inheritance

- A new class of objects can be created quickly and conveniently by inheritance—the new class absorbs the characteristics of an existing class, possibly customizing them and adding unique characteristics of its own.
- In our car analogy, an object of class "convertible" certainly *is an* object of the more *general* class "automobile," but more *specifically*, the roof can be raised or lowered.

1.10 Typical C Program Development Environment

- C systems generally consist of several parts: a program development environment, the language and the C Standard Library.
- C programs typically go through six phases to be executed (Fig. 1.7).
- These are: edit, preprocess, compile, link, load and execute.
- Although *C How to Program, 7/e* is a generic C textbook (written independently of the details of any particular operating system), we concentrate in this section on a typical Linux-based C system.

1.10 Typical C Program Development Environment (Cont.)

- [Note: The programs in this book will run with little or no modification on most current C systems, including Microsoft Windows-based systems.] If you're not using a Linux system, refer to the manuals for your system or ask your instructor how to accomplish these tasks in your environment.
- Check out our C Resource Center at `www.deitel.com/C` to locate "getting started" tutorials for popular C compilers and development environments.

1.10.1 Phase 1: Creating a Program

- Phase 1 consists of editing a file.
- This is accomplished with an editor program.
- Two editors widely used on Linux systems are `vi` and `emacs`.
- Software packages for the C/C++ integrated program development environments such as Eclipse and Microsoft Visual Studio have editors that are integrated into the programming environment.
- You type a C program with the editor, make corrections if necessary, then store the program on a secondary storage device such as a hard disk.
- C program file names should end with the `.c` extension.

1.10.2 Phases 2 and 3: Preprocessing and Compiling a C Program

- In Phase 2, then you give the command to compile the program.
- The compiler translates the C program into machine language-code (also referred to as object code).
- In a C system, a preprocessor program executes automatically before the compiler's translation phase begins.
- The C preprocessor obeys special commands called preprocessor directives, which indicate that certain manipulations are to be performed on the program before compilation.

1.10.2 Phases 2 and 3: Preprocessing and Compiling a C Program (Cont.)

- These manipulations usually consist of including other files in the file to be compiled and performing various text replacements.
- The most common preprocessor directives are discussed in the early chapters; a detailed discussion of preprocessor features appears in Chapter 13.
- In Phase 3, the compiler translates the C program into machine-language code.
- A syntax error occurs when the compiler cannot recognize a statement because it violates the rules of the language.
- Syntax errors are also called compile errors, or compile-time errors.

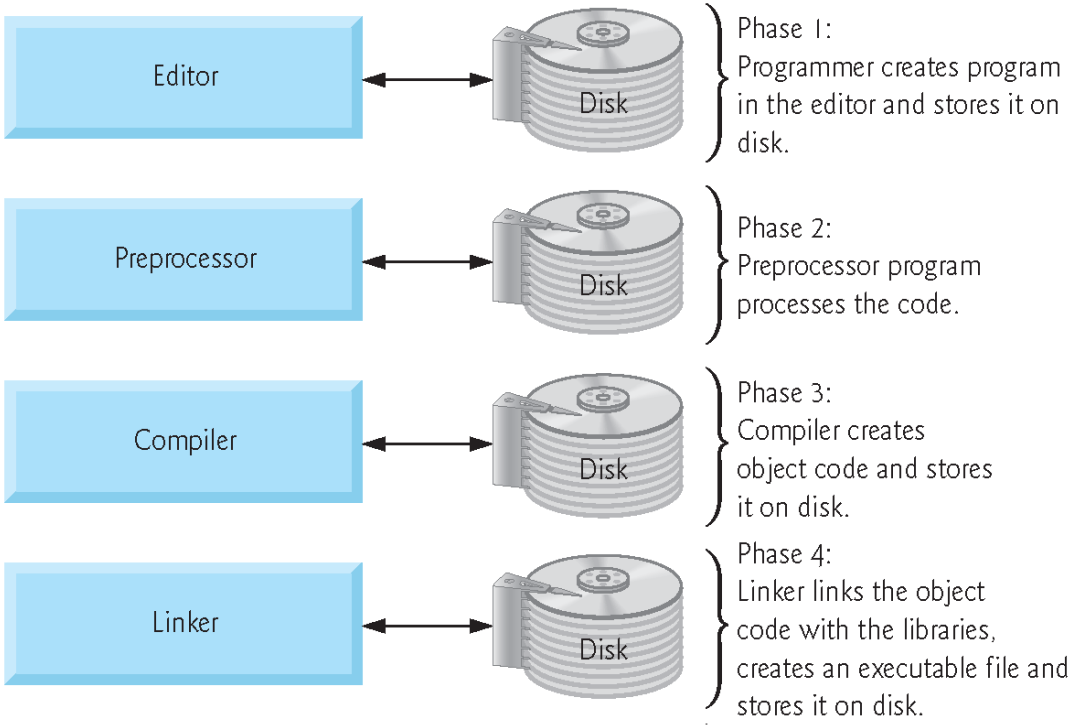


Fig. 1.7 | Typical C development environment. (Part 1 of 3.)

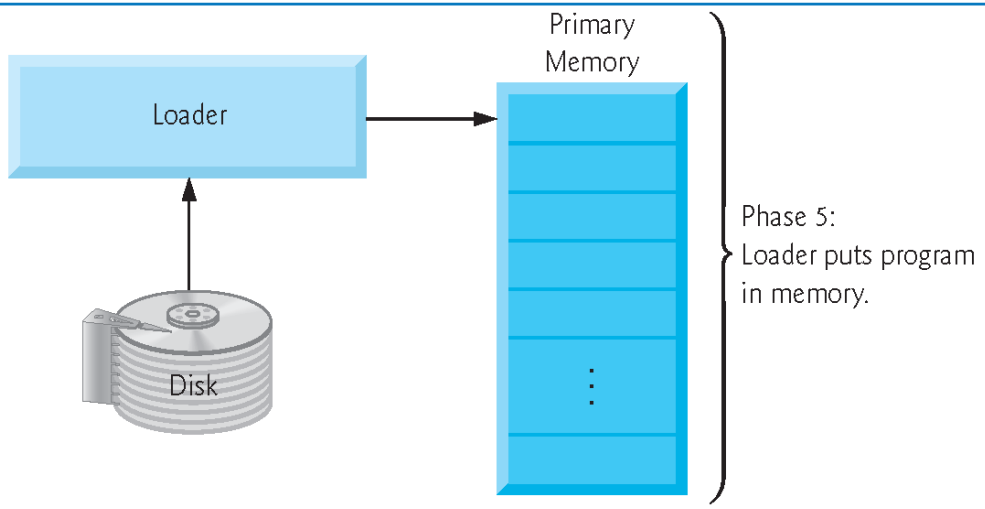


Fig. 1.7 | Typical C development environment. (Part 2 of 3.)

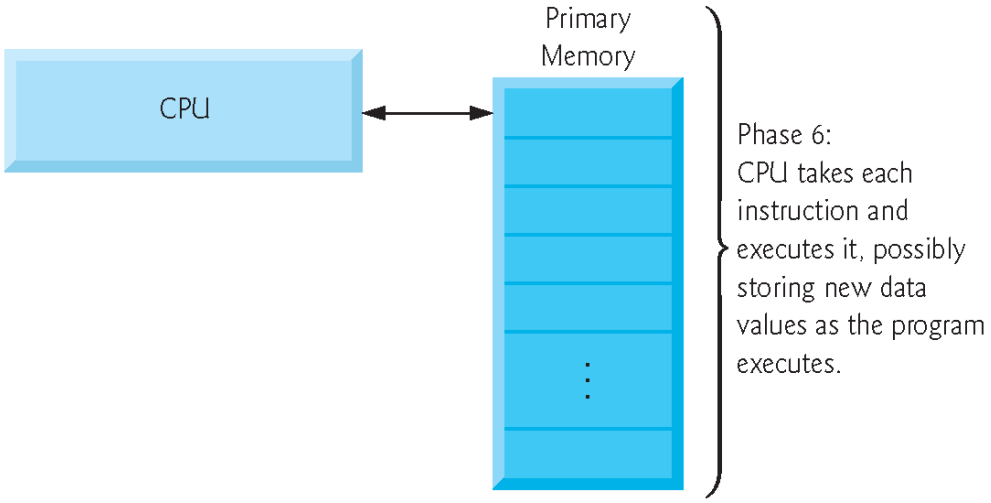


Fig. 1.7 | Typical C development environment. (Part 3 of 3.)

1.10.3 Phase 4: Linking

- The next phase is called linking.
- C programs typically contain references to functions defined elsewhere, such as in the standard libraries or in the private libraries of groups of programmers working on a particular project.
- The object code produced by the C compiler typically contains "holes" due to these missing parts.
- A linker links the object code with the code for the missing functions to produce an executable image (with no missing pieces).
- On a typical Linux system, the command to compile and link a program is called `gcc` (the GNU compiler).

1.10.3 Phase 4: Linking (Cont.)

- To compile and link a program named `welcome.c` type
  - `gcc welcome.c`
- at the Linux prompt and press the *Enter* key (or *Return* key).
- [Note: Linux commands are case sensitive; make sure that each `c` is lowercase and that the letters in the filename are in the appropriate case.]
- If the program compiles and links correctly, a file called `a.out` is produced.
- This is the executable image of our `welcome.c` program.

1.10.4 Phase 5: Loading

- The next phase is called loading.
- Before a program can be executed, the program must first be placed in memory.
- This is done by the loader, which takes the executable image from disk and transfers it to memory.
- Additional components from shared libraries that support the program are also loaded.

1.10.5 Phase 6: Execution

- Finally, the computer, under the control of its CPU, executes the program one instruction at a time.
- To load and execute the program on a Linux system, type `./a.out` at the Linux prompt and press *Enter*.

1.10.6 Problems That May Occur at Execution Time

- Programs do not always work on the first try.
- Each of the preceding phases can fail because of various errors that we'll discuss.
- For example, an executing program might attempt to divide by zero (an illegal operation on computers just as in arithmetic).
- This would cause the computer to display an error message.
- You would then return to the edit phase, make the necessary corrections and proceed through the remaining phases again to determine that the corrections work properly.

Common Programming Error 1.1

Errors such as division-by-zero occur as a program runs, so they are called runtime errors or execution-time errors. Divide-by-zero is generally a fatal error, i.e., one that causes the program to terminate immediately without successfully performing its job. Nonfatal errors allow programs to run to completion, often producing incorrect results.

1.10.7 Standard Input, Standard Output and Standard Error Streams

- Most C programs input and/or output data.
- Certain C functions take their input from `stdin` (the standard input stream), which is normally the keyboard, but `stdin` can be connected to another stream.
- Data is often output to `stdout` (the standard output stream), which is normally the computer screen, but `stdout` can be connected to another stream.
- When we say that a program prints a result, we normally mean that the result is displayed on a screen.

1.10.7 Standard Input, Standard Output and Standard Error Streams (Cont.)

- Data may be output to devices such as disks and printers.
- There is also a standard error stream referred to as `stderr`.
- The `stderr` stream (normally connected to the screen) is used for displaying error messages.
- It's common to route regular output data, i.e., `stdout`, to a device other than the screen while keeping `stderr` assigned to the screen so that the user can be immediately informed of errors.

1.11 Test-Driving a C Application in Windows, Linux and Mac OS X

- In this section, you'll run and interact with your first C application.
- You'll begin by running an entertaining guess-the-number game, which picks a number from 1 to 1000 and prompts you to guess it.
- If your guess is correct, the game ends.
- If your guess is not correct, the application indicates whether your guess is higher or lower than the correct number. There is no limit on the number of guesses you can make.

1.11 Test-Driving a C Application in Windows, Linux and Mac OS X (Cont.)

- For this test-drive only, we've modified this application from the exercise you'll be asked to create in Chapter 5.
- Normally this application randomly selects the correct answers.
- The modified application uses the same sequence of correct answers every time you execute the program (though this may vary by compiler), so you can use the same guesses we use in this section and see the same results.

1.11.1 Running a C Application from the Windows Command Prompt

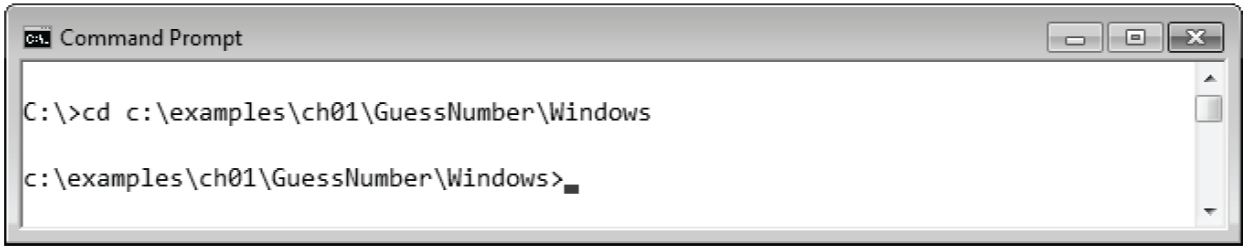


Fig. 1.8 | Opening a **Command Prompt** window and changing the directory.

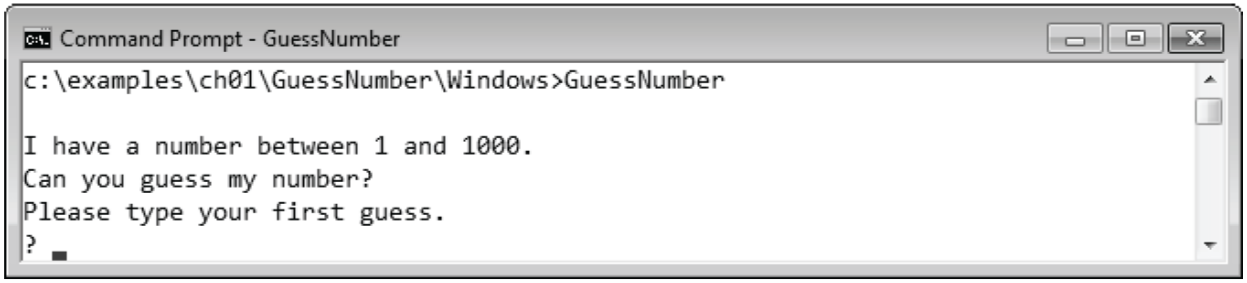


Fig. 1.9 | Running the GuessNumber application.

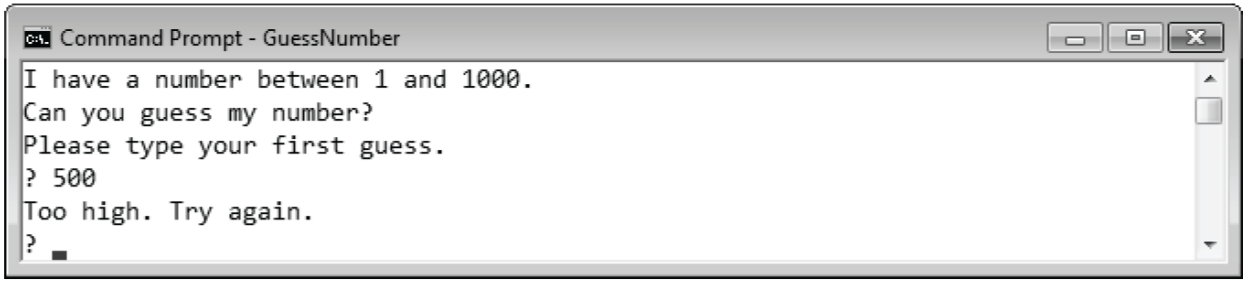


Fig. 1.10 | Entering your first guess.

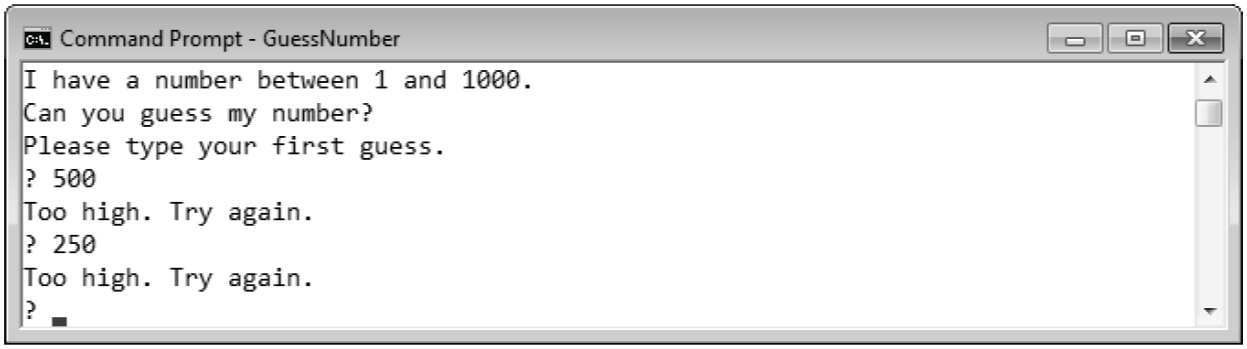


Fig. 1.11 | Entering a second guess and receiving feedback.

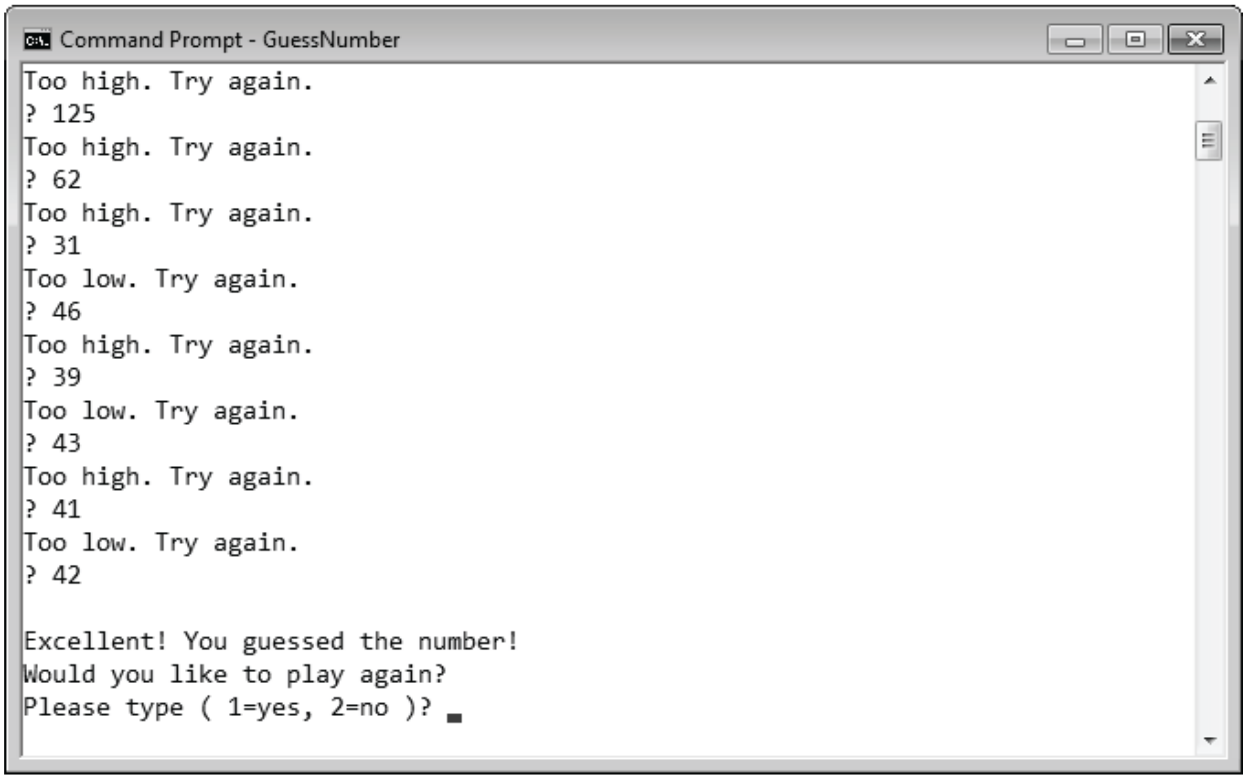


Fig. 1.12 | Entering additional guesses and guessing the correct number.

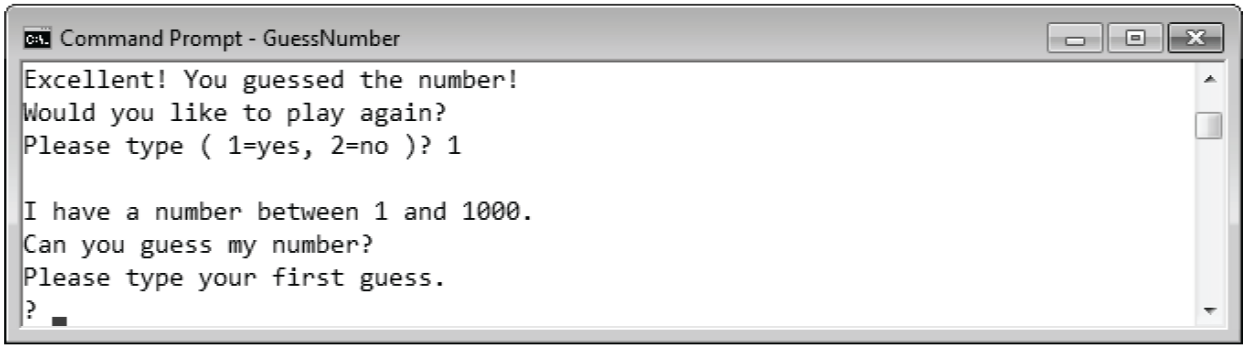


Fig. 1.13 | Playing the game again.

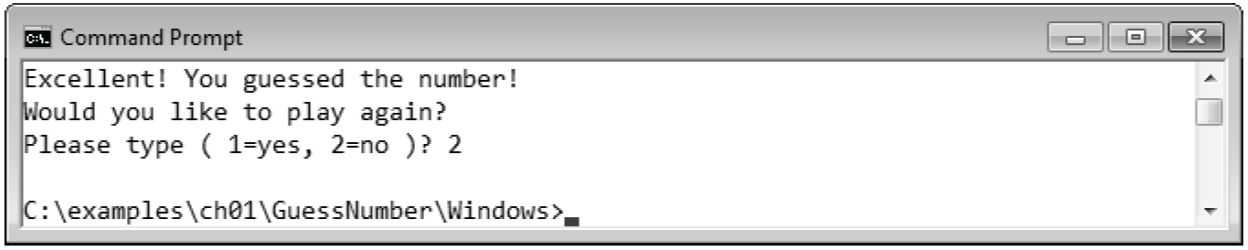


Fig. 1.14 | Exiting the game.

1.11.2 Running a C Application Using GNU C with Linux

```
~$ cd examples/ch01/GuessNumber/GNU
~/examples/ch01/GuessNumber/GNU$
```

Fig. 1.15 | Changing to the GuessNumber application's directory.

```
~/examples/ch01/GuessNumber/GNU$ gcc GuessNumber.c -o GuessNumber
~/examples/ch01/GuessNumber/GNU$
```

Fig. 1.16 | Compiling the GuessNumber application using the `gcc` command.

```
~/examples/ch01/GuessNumber/GNU$ ./GuessNumber

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
?
```

Fig. 1.17 | Running the GuessNumber application.

```
~/examples/ch01/GuessNumber/GNU$ ./GuessNumber

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too high. Try again.
?
```

Fig. 1.18 | Entering an initial guess.

```
~/examples/ch01/GuessNumber/GNU$ ./GuessNumber

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too high. Try again.
? 250
Too low. Try again.
?
```

Fig. 1.19 | Entering a second guess and receiving feedback.

```
Too low. Try again.
? 375
Too low. Try again.
? 437
Too high. Try again.
? 406
Too high. Try again.
? 391
Too high. Try again.
? 383
Too low. Try again.
? 387
Too high. Try again.
? 385
Too high. Try again.
? 384

Excellent! You guessed the number!
Would you like to play again?
Please type ( 1=yes, 2=no )?
```

Fig. 1.20 | Entering additional guesses and guessing the correct number.

```
Excellent! You guessed the number!
Would you like to play again?
Please type ( 1=yes, 2=no )? 1

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
?
```

Fig. 1.21 | Playing the game again.

```
Excellent! You guessed the number!
Would you like to play again?
Please type ( 1=yes, 2=no )? 2

~/examples/ch01/GuessNumber/GNU$
```

Fig. 1.22 | Exiting the game.

1.11.3 Running a C Application Using GNU C with Mac OS X

```
hostName:~ userFolder$ cd Documents/examples/ch01/GuessNumber/GNU
hostName:GNU$
```

Fig. 1.23 | Changing to the GuessNumber application's directory.

```
hostName:GNU~ userFolder$ gcc GuessNumber.c -o GuessNumber
hostName:GNU~ userFolder$
```

Fig. 1.24 | Compiling the GuessNumber application using the `gcc` command.

```
hostName:GNU~ userFolder$ ./GuessNumber

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
?
```

Fig. 1.25 | Running the GuessNumber application.

```
hostName:GNU~ userFolder$ ./GuessNumber

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too low. Try again.
?
```

Fig. 1.26 | Entering an initial guess.

```
hostName:GNU~ userFolder$ ./GuessNumber

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too low. Try again.
? 750
Too low. Try again.
?
```

Fig. 1.27 | Entering a second guess and receiving feedback.

```
Too low. Try again.
? 825
Too high. Try again.
? 788
Too low. Try again.
? 806
Too low. Try again.
? 815
Too high. Try again.
? 811
Too high. Try again.
? 808

Excellent! You guessed the number!
Would you like to play again?
Please type ( 1=yes, 2=no )?
```

Fig. 1.28 | Entering additional guesses and guessing the correct number.

```
Excellent! You guessed the number!
Would you like to play again?
Please type ( 1=yes, 2=no )? 1

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
?
```

Fig. 1.29 | Playing the game again.

```
Excellent! You guessed the number!
Would you like to play again?
Please type ( 1=yes, 2=no )? 2

hostName:GNU~ userFolder$
```

Fig. 1.30 | Exiting the game.

1.12 Operating Systems

- Operating systems are software systems that make using computers more convenient for users, application developers and system administrators.
- Operating systems provide services that allow each application to execute safely, efficiently and *concurrently* (i.e., in parallel) with other applications.
- The software that contains the core components of the operating system is called the kernel.
- Popular desktop operating systems include Linux, Windows 7 and Mac OS X.
- Popular mobile operating systems used in smartphones and tablets include Google's Android, Apple's iOS (for iPhone, iPad and iPod Touch devices), BlackBerry OS and Windows Phone 7.

1.12.1 Windows—A Proprietary Operating System

- In the mid-1980s, Microsoft developed the Windows operating system, consisting of a graphical user interface built on top of DOS—an enormously popular personal-computer operating system of the time that users interacted with by *typing* commands.
- Windows borrowed from many concepts (such as icons, menus and windows) developed by Xerox PARC and popularized by early Apple Macintosh operating systems.
- Windows is a *proprietary* operating system—it's controlled by Microsoft exclusively.

1.12.2 Linux—An Open-Source Operating System

- The Linux operating system is perhaps the greatest success of the *open-source* movement.
- Open-source software departs from the *proprietary* software development style that dominated software's early years.
- With open-source development, individuals and companies *contribute* their efforts in developing, maintaining and evolving software in exchange for the right to use that software for their own purposes, typically at no charge.
- Rapid improvements to computing and communications, decreasing costs and open-source software have made it much easier and more economical to create a software-based business now than just a decade ago.
- A great example is Facebook, which was launched from a college dorm room and built with open-source software.

1.12.2 Linux—An Open-Source Operating System

- The Linux kernel is the core of the most popular open-source, freely distributed, full-featured operating system.
- It's developed by a loosely organized team of volunteers and is popular in servers, personal computers and embedded systems.
- Unlike that of proprietary operating systems like Microsoft's Windows and Apple's Mac OS X, Linux source code (the program code) is available to the public for examination and modification and is free to download and install.
- Linux has become extremely popular on servers and in embedded systems, such as Google's Android-based smartphones.

1.12.3 Apple's Mac OS X; Apple's iOS for iPhone®, iPad® and iPod Touch® Devices

- In 1979, Steve Jobs and several Apple employees visited Xerox PARC (Palo Alto Research Center) to learn about Xerox's desktop computer that featured a graphical user interface (GUI).
- That GUI served as the inspiration for the Apple Macintosh, launched with much fanfare in a memorable Super Bowl ad in 1984.
- The Objective-C programming language, created by Brad Cox and Tom Love at Stepstone in the early 1980s, added capabilities for object-oriented programming (OOP) to the C programming language.

1.12.3 Apple's Mac OS X; Apple's iOS for iPhone®, iPad® and iPod Touch® Devices

- Steve Jobs left Apple in 1985 and founded NeXT Inc. In 1988, NeXT licensed Objective-C from StepStone and developed an Objective-C compiler and libraries which were used as the platform for the NeXTSTEP operating system's user interface and Interface Builder—used to construct graphical user interfaces.
- Jobs returned to Apple in 1996 when Apple bought NeXT. Apple's Mac OS X operating system is a descendant of NeXTSTEP.
- Apple's proprietary iPhone operating system, iOS, is derived from Apple's Mac OS X and is used in the iPhone, iPad and iPod Touch devices.

1.12.4 Google's Android

- Android—the fastest growing mobile and smartphone operating system—is based on the Linux kernel and Java.
- One benefit of developing Android apps is the openness of the platform. The operating system is open source and free.
- The Android operating system is used in numerous smartphones, e-reader devices, tablet computers, in-store touch-screen kiosks, cars, robots, multimedia players and more.

1.13 The Internet and the World Wide Web

- The Internet—a global network of computers—was made possible by the *convergence of computing and communications technologies*.
- In the late 1960s, ARPA (the Advanced Research Projects Agency) rolled out blueprints for networking the main computer systems of about a dozen ARPA-funded universities and research institutions.
- ARPA proceeded to implement the ARPANET, which eventually evolved into today's Internet.

1.13 The Internet and the World Wide Web (Cont.)

Packet Switching

- A primary goal for ARPANET was to allow multiple users to send and receive information simultaneously over the same communications paths (e.g., phone lines).
- The network operated with a technique called packet switching, in which digital data was sent in small bundles called packets.
- The packets contained address, error-control and sequencing information.
  - The address information allowed packets to be routed to their destinations.
  - The sequencing information helped in reassembling the packets—which, because of complex routing mechanisms, could actually arrive out of order—into their original order for presentation to the recipient.

1.13 The Internet and the World Wide Web (Cont.)

- The network was designed to operate without centralized control.
- If a portion of the network failed, the remaining working portions would still route packets from senders to receivers over alternative paths for reliability.

TCP/IP

- The protocol (i.e., set of rules) for communicating over the ARPANET became known as TCP—the Transmission Control Protocol.
- TCP ensured that messages were properly routed from sender to receiver and that they arrived intact.

1.13 The Internet and the World Wide Web (Cont.)

- As the Internet evolved, organizations worldwide were implementing their own networks. One challenge was to get these different networks to communicate.
- ARPA accomplished this with the development of IP—the Internet Protocol, truly creating a network of networks, the current architecture of the Internet.
- The combined set of protocols is now commonly called TCP/IP.

1.13 The Internet and the World Wide Web (Cont.)

World Wide Web, HTML, HTTP

- The World Wide Web allows you to execute web-based applications and to locate and view multimedia-based documents on almost any subject over the Internet.
- In 1989, Tim Berners-Lee of CERN (the European Organization for Nuclear Research) began to develop a technology for sharing information via hyperlinked text documents. Berners-Lee called his invention the HyperText Markup Language (HTML).
- He also wrote communication protocols to form the backbone of his new information system, which he called the World Wide Web.
- In particular, he wrote the Hypertext Transfer Protocol (HTTP)—a communications protocol used to send information over the web.

1.13 The Internet and the World Wide Web (Cont.)

- The URL (Uniform Resource Locator) specifies the address (i.e., location) of the web page displayed in the browser window.
- Each web page on the Internet is associated with a unique URL.
- Hypertext Transfer Protocol Secure (HTTPS) is the standard for transferring encrypted data on the web.

1.13 The Internet and the World Wide Web (Cont.)

Mosaic, Netscape, Emergence of Web 2.0

- Web use exploded with the availability in 1993 of the Mosaic browser, which featured a user-friendly graphical interface.
- Marc Andreessen, whose team at the National Center for Supercomputing Applications developed Mosaic, went on to found Netscape, the company that many people credit with igniting the explosive Internet economy of the late 1990s.
- In 2003 there was a noticeable shift in how people and businesses were using the web and developing web-based applications. The term Web 2.0 was coined by Dale Dougherty of O'Reilly Media in 2003 to describe this trend.
- Generally, Web 2.0 companies use the web as a platform to create collaborative, community-based sites (e.g., social networking sites, blogs, wikis).

1.13 The Internet and the World Wide Web (Cont.)

- Web 2.0 involves the users—not only do they create content, but they help organize it, share it, remix it, critique it, update it, etc.
- Web 2.0 is a conversation, with everyone having the opportunity to speak and share views.

1.13 The Internet and the World Wide Web (Cont.)

Architecture of Participation

- Web 2.0 embraces an architecture of participation—a design that encourages user interaction and community contributions.
- You, the user, are the most important aspect of Web 2.0—so important, in fact, that in 2006, TIME magazine's "Person of the Year" was "You."
- The article recognized the social phenomenon of Web 2.0—the shift away from a powerful few to an empowered many.
- For websites like Facebook®, Twitter™, YouTube, eBay® and Wikipedia® users create the content, while the companies provide the platforms on which to enter, manipulate and share the information.

1.14 Some Key Software Development Terminology

- Figure 1.31 lists a number of buzzwords that you'll hear in the software development community.

Technology	Description
Ajax	Ajax is one of the premier Web 2.0 software technologies. Ajax helps Internet-based applications perform like desktop applications—a difficult task, given that such applications suffer transmission delays as data is shuttled back and forth between your computer and servers on the Internet.
Agile software development	Agile software development is a set of methodologies that try to get software implemented faster and using fewer resources than previous methodologies. Check out the Agile Alliance (`www.agilealliance.org`) and the Agile Manifesto (`www.agilemanifesto.org`).
Refactoring	Refactoring involves reworking programs to make them clearer and easier to maintain while preserving their correctness and functionality. It's widely employed with agile development methodologies. Many IDEs include *refactoring tools* to do major portions of the reworking automatically.
Design patterns	Design patterns are proven architectures for constructing flexible and maintainable object-oriented software. The field of design patterns tries to enumerate those recurring patterns, encouraging software designers to *reuse* them to develop better-quality software using less time, money and effort.

Fig. 1.31 | Software technologies. (Part 1 of 4.)

Technology	Description
LAMP	MySQL is an open-source database management system. PHP is the most popular open-source server-side Internet "scripting" language for developing Internet-based applications. LAMP is an acronym for the set of open-source technologies that many developers use to build web applications—it stands for Linux, Apache, MySQL and PHP (or Perl or Python—two other languages used for similar purposes).

Fig. 1.31 | Software technologies. (Part 2 of 4.)

Technology	Description
Software as a Service (SaaS)	Software has generally been viewed as a product; most software still is offered this way. If you want to run an application, you buy a software package from a software vendor—often a CD, DVD or web download. You then install that software on your computer and run it as needed. As new versions of the software appear, you upgrade your software, often requiring significant time and at considerable expense. This process can become cumbersome for organizations with tens of thousands of systems that must be maintained on a diverse array of computer equipment. With Software as a Service (SaaS), the software runs on servers elsewhere on the Internet. When that server is updated, all clients worldwide see the new capabilities—no local installation is needed. You access the service through a browser. Browsers are quite portable, so you can run the same applications on a wide variety of computers from anywhere in the world. Salesforce.com, Google, and Microsoft's Office Live and Windows Live all offer SaaS. SaaS is a capability of cloud computing.

Fig. 1.31 | Software technologies. (Part 3 of 4.)

Technology	Description
Platform as a Service (PaaS)	Platform as a Service (PaaS), another capability of cloud computing, provides a computing platform for developing and running applications as a service over the web, rather than installing the tools on your computer. PaaS providers include Google App Engine, Amazon EC2, Bungee Labs and more.
Software Development Kit (SDK)	Software Development Kits (SDKs) include the tools and documentation developers use to program applications.

Fig. 1.31 | Software technologies. (Part 4 of 4.)

Version	Description
Alpha	An *alpha* version is the earliest release of a software product that's still under active development. Alpha versions are often buggy, incomplete and unstable and are released to a relatively small number of developers for testing new features, getting early feedback, etc.
Beta	*Beta* versions are released to a larger number of developers later in the development process after most major bugs have been fixed and new features are nearly complete. Beta software is more stable, but still subject to change.
Release candidates	*Release candidates* are generally *feature complete* and (supposedly) bug free and ready for use by the community, which provides a diverse testing environment—the software is used on different systems, with varying constraints and for a variety of purposes. Any bugs that appear are corrected, and eventually the final product is released to the general public. Software companies often distribute incremental updates over the Internet.
Continuous beta	Software that's developed using this approach generally does not have version numbers (for example, Google search or Gmail). The software, which is hosted in the cloud (not installed on your computer), is constantly evolving so that users always have the latest version.

Fig. 1.32 | Software product-release terminology.

1.15 Keeping Up-to-Date with Information Technologies

- Figure 1.33 lists key technical and business publications that will help you stay up-to-date with the latest news and trends in technology.

Publication	URL
ACM TechNews	technews.acm.org/
ACM Transactions on Accessible Computing	www.gccis.rit.edu/taccess/index.html
ACM Transactions on Internet Technology	toit.acm.org/
Bloomberg BusinessWeek	www.businessweek.com
CNET	news.cnet.com
Communications of the ACM	cacm.acm.org/
Computer World	www.computerworld.com
Engadget	www.engadget.com
eWeek	www.eweek.com
Fast Company	www.fastcompany.com/
Fortune	money.cnn.com/magazines/fortune/
IEEE Computer	www.computer.org/portal/web/computer

Fig. 1.33 | Technical and business publications. (Part 1 of 2.)

Publication	URL
IEEE Internet Computing	www.computer.org/portal/web/internet/home
InfoWorld	www.infoworld.com
Mashable	mashable.com
PCWorld	www.pcworld.com
SD Times	www.sdtimes.com
Slashdot	slashdot.org/
Smarter Technology	www.smartertechnology.com
Technology Review	technologyreview.com
Techcrunch	techcrunch.com
Wired	www.wired.com

Fig. 1.33 | Technical and business publications. (Part 2 of 2.)