Chapter 1 Introduction to Computers, the Internet and the World Wide Web

C++ How to Program, Late Objects Version, 7/e

1.1 Introduction

- Download the example programs from `www.deitel.com/books/cpphtp7LOV/`
- Computers (often referred to as hardware) are controlled by software (i.e., the instructions you write to command the computer to perform actions and make decisions).
- C++ is standardized in the United States through the American National Standards Institute (ANSI) and worldwide through the efforts of the International Organization for Standardization (ISO).

1.1 Introduction (cont.)

- To keep up to date with C++ developments at Deitel & Associates, please register for our free e-mail newsletter, the *Deitel® Buzz Online*, at
  `www.deitel.com/newsletter/subscribe.html`
- C++ and related Resource Centers at
  `www.deitel.com/ResourceCenters.html`
- Errata and updates for this book are posted at
  `www.deitel.com/books/cpphtp7LOV/`
- You are embarking on a challenging and rewarding path.
- As you proceed, if you have any questions, please send e-mail to
  `deitel@deitel.com`

1.2 Computers: Hardware and Software

- A computer is a device that can perform computations and make logical decisions billions of times faster than human beings can.
- Today's fastest supercomputers can perform thousands of trillions (quadrillions) of instructions per second!
- Computers process data under the control of sets of instructions called computer programs.
- Programs guide the computer through orderly sets of actions specified by people called computer programmers.
- A computer consists of various devices referred to as hardware.
- The programs that run on a computer are referred to as software.

1.3 Computer Organization

- Virtually every computer may be envisioned as divided into six logical units or sections:
  - Input unit. This "receiving" section obtains from input devices and places it at the disposal of the other units so that it can be processed.
  - 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.
  - 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. It also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is volatile. The memory unit is often called either memory or primary memory.

1.3 Computer Organization (cont.)

- Arithmetic and logic unit (ALU). This " manufacturing" section performs calculations. It also contains the computer's decision mechanisms. 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.
  - 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
  - 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—such computers are called multiprocessors. A multi-core processor implements multiprocessing on a single integrated circuit chip.

1.3 Computer Organization (cont.)

- 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. Therefore, information on secondary storage devices is said to be persistent.

1.4 Personal, Distributed and Client/Server Computing

- In 1977, Apple Computer popularized personal computing.
- In 1981, IBM, the world's largest computer vendor, introduced the IBM Personal Computer (PC).
  - This quickly legitimized personal computing in business, industry and government organizations, where IBM mainframes were heavily used.
- Machines could be linked together in computer networks, sometimes over telephone lines and sometimes in local area networks (LANs) within an organization.
  - Led to the phenomenon of distributed computing, in which an organization's computing is distributed over networks to the sites where the organization's work is performed.

1.4 Personal, Distributed and Client/Server Computing (cont.)

- Today's personal computers are as powerful as the million-dollar machines of just a few decades ago.
- Information is shared easily across computer networks, where computers called servers offer a common data store that may be used by client computers distributed throughout the network, hence the term client/server computing.
- C++ has become widely used for writing software for operating systems, for computer networking and for distributed client/server applications.

1.5 The Internet and the World Wide Web

- The Internet was initiated in the late 1960s with funding supplied by the U.S. Department of Defense.
- Originally designed to connect the main computer systems of about a dozen universities and research organizations.
- With the introduction of the World Wide Web, the Internet has exploded into the world's premier communication mechanism.
- The Internet and the World Wide Web are surely among humankind's most important and profound creations.

1.6 Web 2.0

- Web 2.0 has no single definition but can be explained through a series of Internet trends, one being the empowerment of the user.
- Companies are built almost entirely on community-generated content.
- Takes advantage of collective intelligence
  - the idea that collaboration will result in intelligent ideas.
  - Example: wikis, such as the encyclopedia Wikipedia, allow users access to edit content.
- Tagging, or labeling content, is another key part of the collaborative theme of Web 2.0
  - seen in sites such as Flickr and del.icio.us.
- Social networking sites have experienced extraordinary growth as part of Web 2.0.
  - Rely heavily on network effects, attracting users only if their friends or colleagues are also members.

1.6 Web 2.0 (cont.)

- Social media sites have gained immense popularity, partly due to the increased availability of broadband Internet, often referred to as high-speed Internet.
- Blogs—websites characterized by short postings in reverse chronological order.
  - blogosphere often used to track consumer opinions.
- The increased popularity of open source software has made it cheaper and easier to start Web 2.0 companies.
- Web services—software components accessible by applications (or other software components) over the Internet
  - Favoring the "webtop" over the desktop in much new development.
- Mashups combine two or more existing web applications to serve a new purpose and are dependent on open access to web services.

1.7 Machine Languages, Assembly Languages and High-Level Languages

- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate translation steps.
- Computer languages may be divided into three general types:
  - Machine languages
  - Assembly languages
  - High-level languages
- Any computer can directly understand only its own machine language.
  - The "natural language" of a computer and as such is defined by its hard-ware de-sign.
- Machine languages are machine dependent.
  - Too slow (for development), tedious and error prone for most programmers.

1.7 Machine Languages, Assembly Languages and High-Level Languages (cont.)

- English-like abbreviations that represent elementary operations formed the basis of assembly languages.
- Translator programs called assemblers convert assembly-language programs to machine language.
- Programmers still had to use many instructions to accomplish even the simplest tasks.
- To speed the programming process, high-level languages were developed in which single statements could be written to accomplish substantial tasks.
- Translator programs called compilers convert high-level language programs into machine language.
- Interpreter programs execute high-level language programs directly (without the delay of compilation), although slower than compiled programs run.

1.8 History of C and C++

- C++ evolved from C, which evolved from two previous programming languages, BCPL and B.
- BCPL was developed in 1967 by Martin Richards as a language for writing operating systems software and compilers for operating systems.
- Ken Thompson modeled many features in B after their counterparts in BCPL and used B to create early versions of the UNIX operating system at Bell Laboratories in 1970.
- The C language was evolved from B by Dennis Ritchie at Bell Laboratories.
  - C initially became widely known as the development language of the UNIX operating system.
  - Today, most operating systems are written in C and/or C++.
  - C is available for most computers and is hardware independent.
  - It's possible to write C programs that are portable to most computers.

Portability Tip 1.1

Because C is a standardized, hardware-independent, widely available language, applications written in C can be run with little or no modification on a wide range of computers.

1.8 History of C and C++ (cont.)

- C++, an extension of C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories.
- C++ provides capabilities for object-oriented programming.
- Objects are essentially reusable software components that model items in the real world.
- Modular, object-oriented design and implementation makes programmers much more productive than can previous popular programming techniques.
- Object-oriented programs are easier to understand, correct and modify.

1.9 C++ Standard Library

- C++ programs consist of pieces called classes and functions.
- Most C++ programmers take advantage of the rich collections of classes and functions in the C++ Standard Library.
- Two parts to learning the C++ "world."
  - The C++ language itself, and
  - How to use the classes and functions in the C++ Standard Library.
- Many special-purpose class libraries are supplied by independent software vendors.

Software Engineering Observation 1.1

Use a "building-block" approach to create programs. Avoid reinventing the wheel. Use existing pieces wherever possible. Called ***software reuse****, this practice is central to object-oriented programming.*

Software Engineering Observation 1.2

When programming in C++, you'll typically use the following building blocks: classes and functions from the C++ Standard Library, classes and functions you and your colleagues create, and classes and functions from various popular third-party libraries.

Performance Tip 1.1

Using C++ Standard Library functions and classes instead of writing your own versions can improve program performance, because they're written carefully to perform efficiently. This technique also shortens program development time.

Portability Tip 1.2

Using C++ Standard Library functions and classes instead of writing your own improves program portability, because they're included in every C++ implementation.

1.10 Java

- Microprocessors are having a profound impact in intelligent consumer electronic devices.
- Recognizing this, Sun Microsystems in 1991 funded an internal corporate research project code-named Green.
- Resulted in the development of a C++-based language called Java
  - Created by James Gosling.
- The World Wide Web exploded in popularity in 1993, and Sun saw the immediate potential of using Java to add dynamic content to web pages.
- Java garnered the attention of the business community because of the phenomenal interest in the World Wide Web.
- Java is now used to develop large-scale enterprise applications, to enhance the functionality of web servers, to provide applications for consumer devices and for many other purposes.

1.11 Fortran, COBOL, Pascal and Ada

- FORTRAN (FORmula TRANslator) was developed by IBM Corporation in the mid-1950s to be used for scientific and engineering applications that require complex mathematical computations.
  - Still widely used in engineering applications.
- COBOL (COmmon Business Oriented Language) was developed in the late 1950s by computer manufacturers, the U.S. government and industrial computer users.
  - Used for commercial applications that require precise and efficient manipulation of large amounts of data.

1.11 Fortran, COBOL, Pascal and Ada (cont.)

- Structured programming—an approach to writing programs that are clearer and easier to test, debug and modify than large programs produced with previous techniques.
- Pascal programming language was created by Professor Niklaus Wirth in 1971.
  - Designed for teaching structured programming and rapidly became the preferred programming language in most colleges.
- The Ada language was developed under the sponsorship of the U.S. Department of Defense (DoD) during the 1970s and early 1980s.
  - The DoD wanted one language that would fill most of its needs.
  - One important capability of Ada, called multitasking (and called multithreading in more recent languages), allows programmers to specify that many activities are to occur in parallel.

1.12 BASIC, Visual Basic, Visual C++, C# and .NET

- The BASIC (Beginner's All-purpose Symbolic Instruction Code) programming language was developed in the mid-1960s at Dartmouth College as a means of writing simple programs.
  - Primary purpose was to familiarize novices with programming techniques.
- Microsoft's Visual Basic language was introduced in the early 1990s to simplify Windows application development
  - One of the most popular programming languages.
- The .NET platform provides capabilities for creating and running applications that can execute on computers distributed across the Internet.
- Three primary programming languages
  - Visual Basic (based on the original BASIC)
  - Visual C++ (based on C++)
  - Visual C# (based on C++ and Java; developed expressly for the .NET platform).

1.13 Key Software Trend: Object Technology

- Object technology dates back to the mid 1960s.
- The C++ programming language, developed at AT&T by Bjarne Stroustrup in the early 1980s, is based on two languages—C and Simula 67, a simulation programming language developed in Europe and released in 1967.
- C++ absorbed the features of C and added Simula's capabilities for creating and manipulating objects.
- Object technology is a packaging scheme that helps us create meaningful software units.
- Before object-oriented languages appeared, procedural programming languages (such as Fortran, COBOL, Pascal, BASIC and C) were focused on actions (verbs) rather than on things or objects (nouns).

1.13 Key Software Trend: Object Technology (cont.)

- A key problem with procedural programming is that the program units do not effectively mirror real-world entities, so these units are not particularly reusable.
- With object technology, classes, if properly designed, tend to be reusable on future projects.
- Using libraries of reusable componentry can greatly reduce effort required to implement certain kinds of systems (compared to the effort that would be required to reinvent these capabilities on new projects).

Software Engineering Observation 1.3

Extensive class libraries of reusable software components are available on the Internet. Many of these libraries are free.

1.13 Key Software Trend: Object Technology (cont.)

- Key benefit of object-oriented programming is that the software is
  - more understandable
  - better organized and
  - easier to maintain, modify and debug
- Significant because perhaps as much as 80 percent of software costs are associated not with the original efforts to develop the software, but with the continued evolution and maintenance of that software throughout its lifetime.

1.14 Typical C++ Development Environment

- C++ systems generally consist of three parts: a program development environment, the language and the C++ Standard Library.
- C++ programs typically go through six phases: edit, preprocess, compile, link, load and execute.

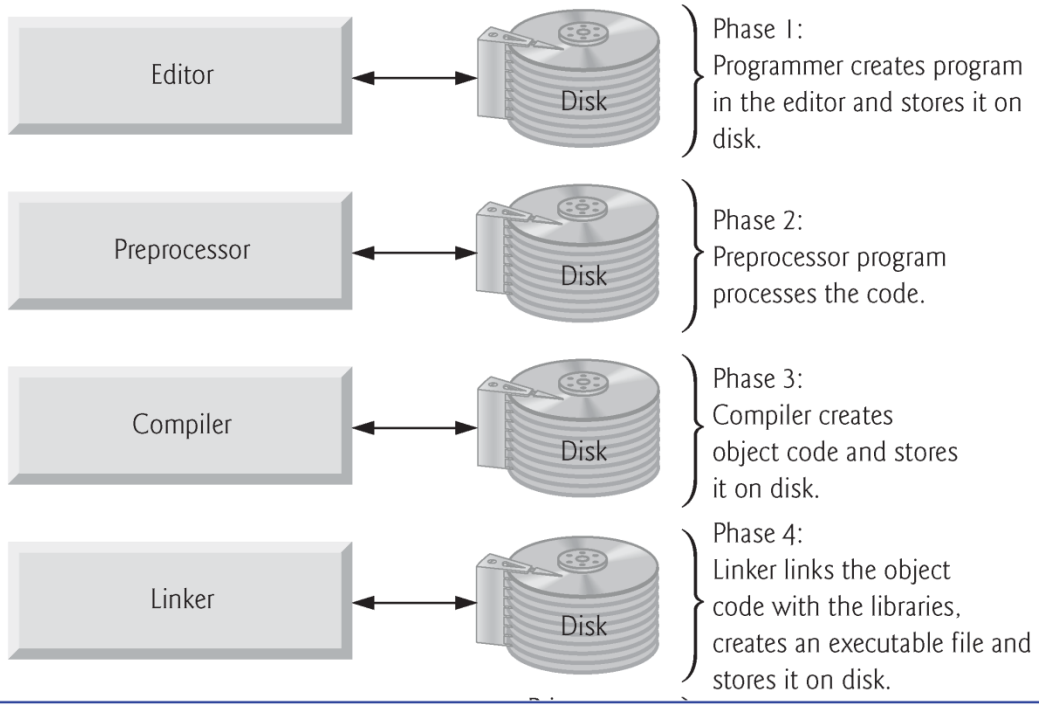

Fig. 1.1 | Typical C++ environment. (Part 1 of 2.)

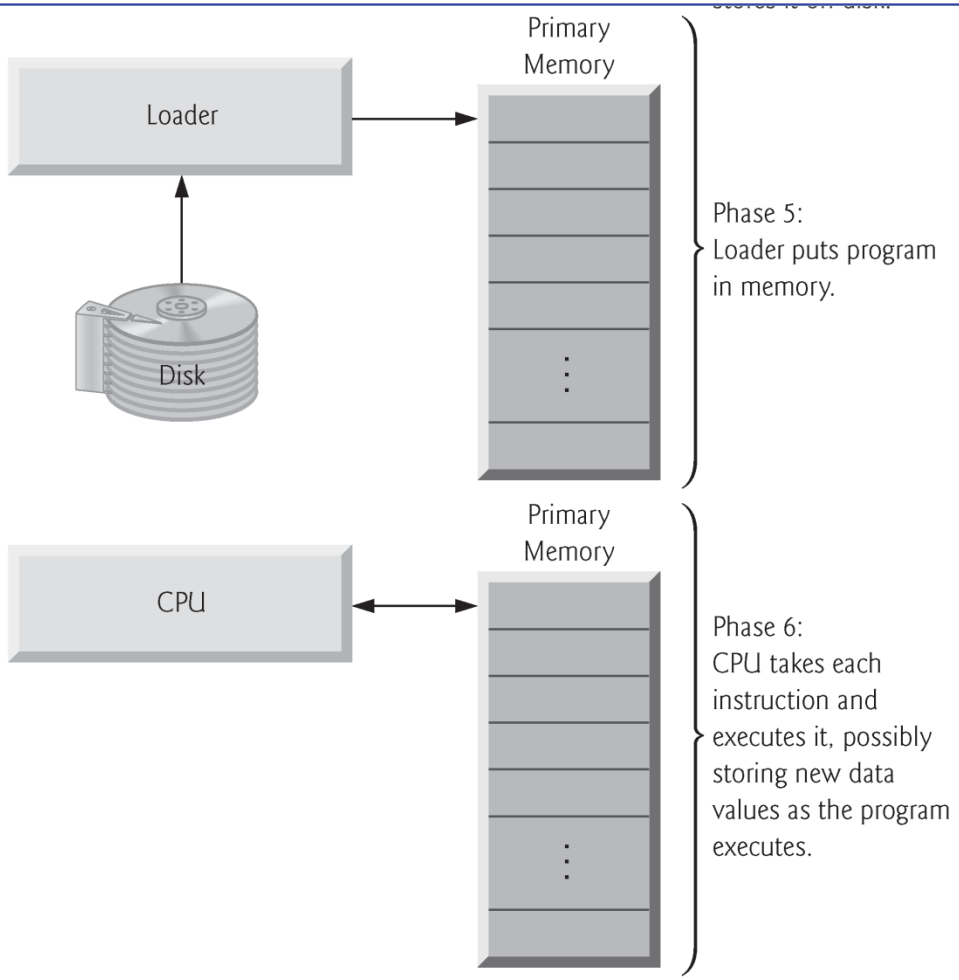

Fig. 1.1 | Typical C++ environment. (Part 2 of 2.)

1.14 Typical C++ Development Environment (cont.)

- Phase 1 consists of editing a file with an editor program (normally known simply as an editor).
  - You type a C++ program (typically referred to as source code) using the editor, make corrections and save the program.
  - C++ source code filenames often end with the `.cpp`, `.cxx`, `.cc` or `.C` extensions (note that `C` is in uppercase) which indicate that a file contains C++ source code.
  - Two editors widely used on UNIX systems are `vi` and `emacs`.
  - C++ software packages such as Microsoft Visual C++ (`msdn.microsoft.com/vstudio/express/visualc/default.aspx`) have editors integrated into the programming environment.

1.14 Typical C++ Development Environment (cont.)

- In phase 2, you give the command to compile the program.
- In a C++ system, a preprocessor program executes automatically before the compiler's translation phase begins.
- The C++ preprocessor obeys commands called preprocessor directives, which indicate that certain manipulations are to be performed on the program before compilation.
- These manipulations usually include other text files to be compiled, and perform various text replacements.
- The most common preprocessor directives are discussed in the early chapters; a detailed discussion of preprocessor features appears in Appendix E, Preprocessor.
- In phase 3, the compiler translates the C++ program into machine-language code (also referred to as object code).

1.14 Typical C++ Development Environment (cont.)

- Phase 4 is called linking.
- The object code produced by the C++ compiler typically contains "holes" due to missing parts, such as references to functions from standard libraries.
- A linker links the object code with the code for the missing functions to produce an executable program.

1.14 Typical C++ Development Environment (cont.)

- Phase 5 is called loading.
- Before a program can be executed, it 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.
- Finally, in Phase 6, the computer, under the control of its CPU, executes the program one instruction at a time.

1.14 Typical C++ Development Environment (cont.)

- Programs do not always work on the first try.
- Each of the preceding phases can fail because of various errors that we discuss throughout the book.
- If this occurs, you'd have to return to the edit phase, make the necessary corrections and proceed through the remaining phases again to determine that the corrections fix the problem(s).

1.14 Typical C++ Development Environment (cont.)

- Most programs in C++ input and/or output data.
- Certain C++ functions take their input from `cin` (the standard input stream; pronounced "see-in"), which is normally the keyboard, but `cin` can be redirected to another device.
- Data is often output to `cout` (the standard output stream; pronounced "see-out"), which is normally the computer screen, but `cout` can be redirected to another device.
- When we say that a program prints a result, we normally mean that the result is displayed on a screen.
  - Data may be output to other devices, such as disks and hardcopy printers.
- There is also a standard error stream referred to as `cerr` that is used for displaying error messages.

Common Programming Error 1.1

Errors such as division by zero occur as a program runs, so they're called ***runtime errors*** *or* ***execution-time errors****.* ***Fatal runtime errors*** *cause programs to terminate immediately without having successfully performed their jobs.* ***Nonfatal runtime errors*** *allow programs to run to completion, often producing incorrect results. [*Note: *On some systems, divide-by-zero is* not *a fatal error. Please see your system documentation.]*

1.15 Notes About C++ and *C++ How to Program, Late objects Version 7/e*

- This book is geared for novice programmers, so we stress program *clarity*.
- If you need additional technical details on C++, you may want to read the C++ standard document, which can be ordered from ANSI at
  - `webstore.ansi.org`
- The title of the document is "Information Technology – Programming Languages – C++" and its document number is INCITS/ISO/IEC 14882-2003.
- We list many websites relating to C++ and object-oriented programming in our C++ Resource Center at `www.deitel.com/cplusplus/`

Good Programming Practice 1.1

Write your C++ programs in a simple and straightforward manner. This is sometimes referred to as KIS ("keep it simple"). Do not "stretch" the language by trying bizarre usages.

Portability Tip 1.3

Although it's possible to write portable programs, there are many problems among different C and C++ compilers and different computers that can make portability difficult to achieve. Writing programs in C and C++ does not guarantee portability. You'll often need to deal directly with compiler and computer variations. As a group, these are sometimes called ***platform*** *variations.*

Good Programming Practice 1.2

Read the documentation for the version of C++ you're using to keep aware of the rich collection of C++ features and that you're using them correctly.

Good Programming Practice 1.3

If, after reading your C++ language documentation, you still are not sure how a feature of C++ works, experiment using a small test program and see what happens. Set your compiler options for "maximum warnings." Study each message that the compiler generates and correct the program to eliminate the messages.

1.16 Test-Driving a C++ Application

- 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.
- [*Note:* For this test drive only, we've modified this application from the exercise you'll be asked to create in Chapter 5, Functions and an Introduction to Recursion. Normally this application randomly selects the correct answer as you execute the program. The modified application uses the same correct answer every time the program executes (though this may vary by compiler), so you can use the same guesses we use in this section and see the same results as we walk you through interacting with your first C++ application.]

1.16 Test-Driving a C++ Application (cont.)

- Test-drive in a Windows Command Prompt

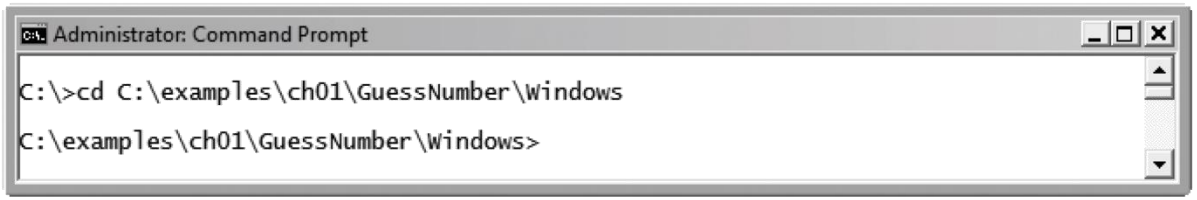

Fig. 1.2 | Opening a **Command Prompt** window and changing the directory.

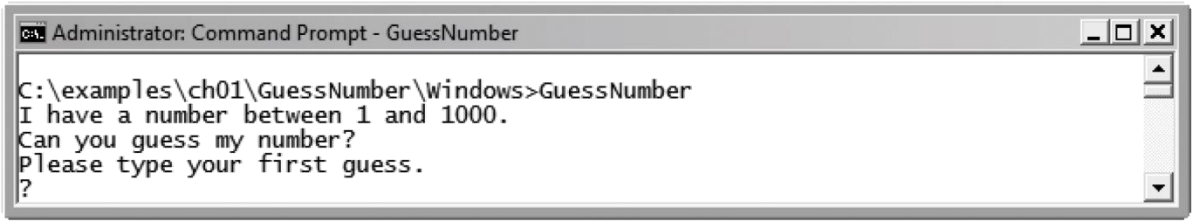

Fig. 1.3 | Running the **GuessNumber** application.

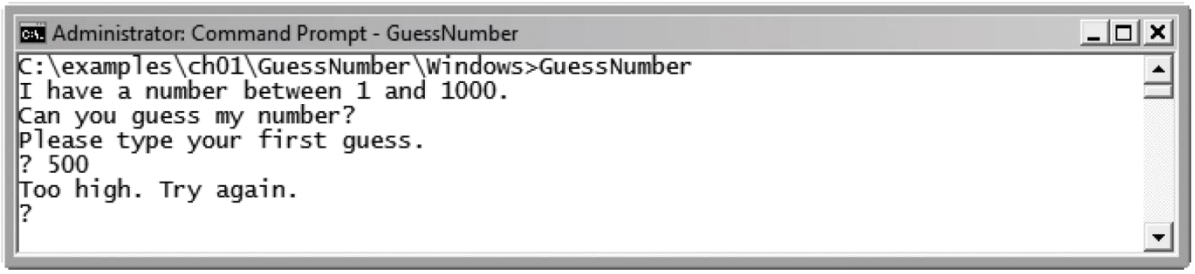

Fig. 1.4 | Entering your first guess.

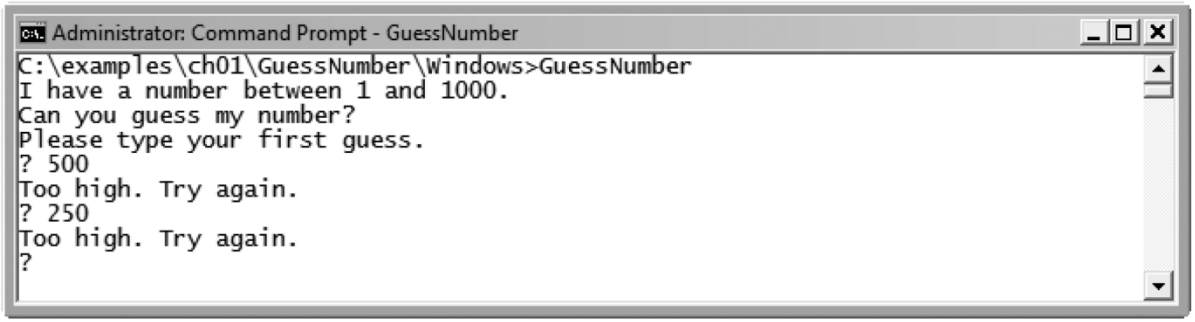

Fig. 1.5 | Entering a second guess and receiving feedback.

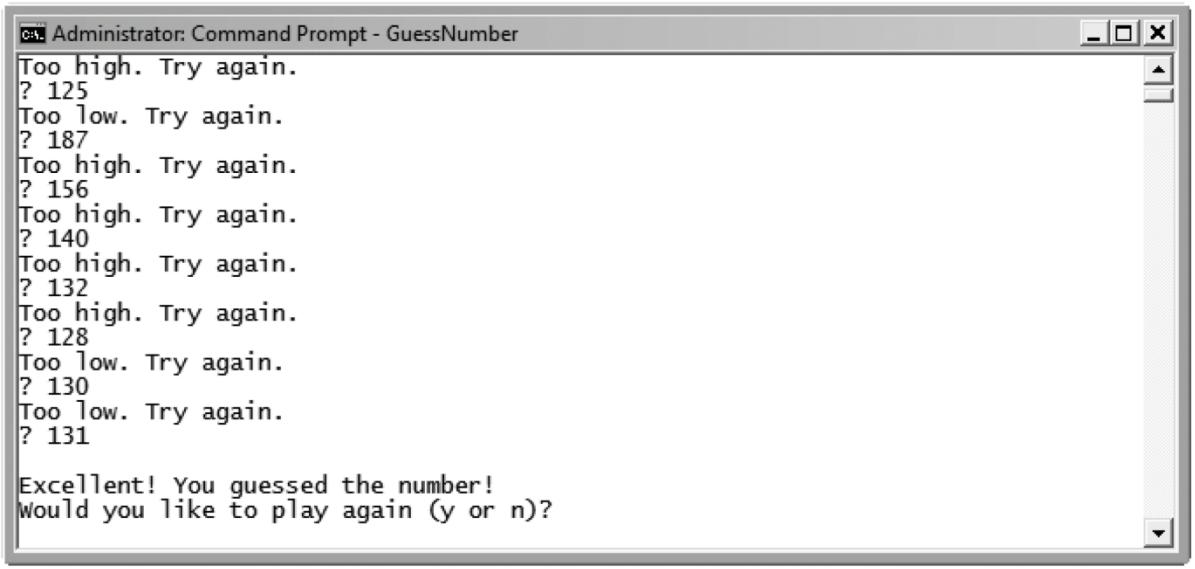

Fig. 1.6 | Entering additional guesses and guessing the correct number.

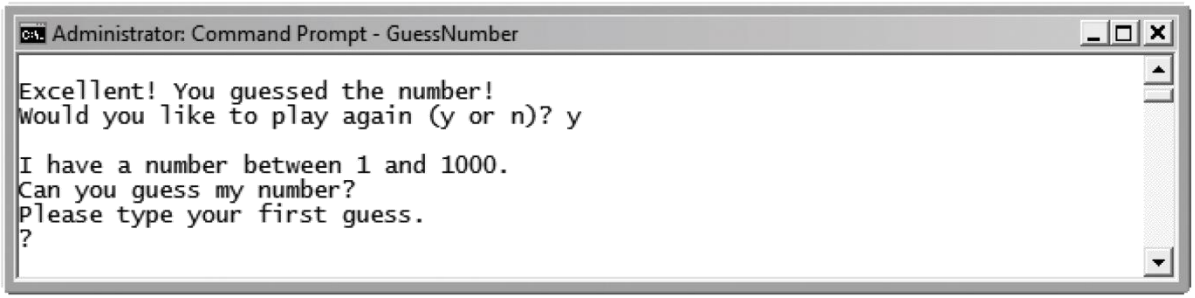

Fig. 1.7 | Playing the game again.

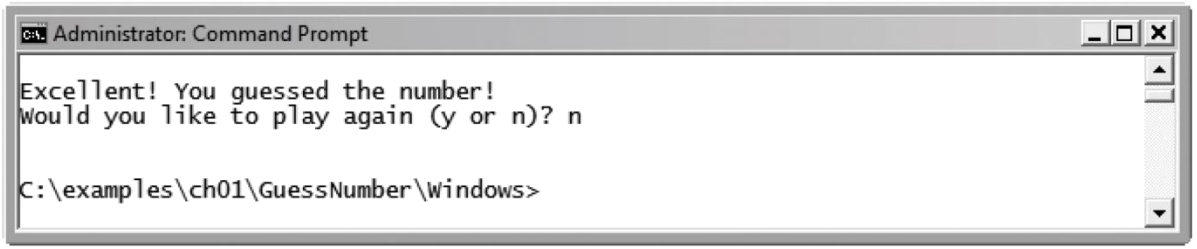

Fig. 1.8 | Exiting the game.

1.16 Test-Driving a C++ Application (cont.)

- Test-drive in a Linux shell

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

Fig. 1.9 | Changing to the **GuessNumber** application's directory.

```
~/examples/ch01/GuessNumber/GNU_Linux$ g++ GuessNumber.cpp -o GuessNumber
~/examples/ch01/GuessNumber/GNU_Linux$
```

Fig. 1.10 | Compiling the **GuessNumber** application using the g++ command.

```
~/examples/ch01/GuessNumber/GNU_Linux$ ./GuessNumber
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
?
```

Fig. 1.11 | Running the **GuessNumber** application.

```
~/examples/ch01/GuessNumber/GNU_Linux$ ./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. 1.12 | Entering an initial guess.

```
~/examples/ch01/GuessNumber/GNU_Linux$ ./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. 1.13 | 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 (y or n)?
```

Fig. 1.14 | Entering additional guesses and guessing the correct number.

```
Excellent! You guessed the number.
Would you like to play again (y or n)? y

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

Fig. 1.15 | Playing the game again.

```
Excellent! You guessed the number.
Would you like to play again (y or n)? n

~/examples/ch01/GuessNumber/GNU_Linux$
```

Fig. 1.16 | Exiting the game.

1.17 Software Technologies

- Agile Software Development is a set of methodologies that try to get software implemented quickly with fewer resources than previous methodologies.
- Refactoring involves reworking code to make it clearer and easier to maintain while preserving its functionality.
- Design patterns are proven architectures for constructing flexible and maintainable object-oriented software.
- Game programming. The computer game business is larger than the first-run movie business.
- Open source software is a style of developing software in whuixh 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.

1.17 Software Technologies (cont.)

- Linux is an open source operating system and one of the greatest successes of the open source movement.
- MySQL is an open source database management system.
- PHP is the most popular open source server-side "scripting" language for developing Internet-based applications.
- LAMP is an acronym for the set of open source technologies that many developers used to build web applications—it stands for Linux, Apache, MySQL and PHP (or Perl or Python—two other popular languages used for similar purposes).
- Ruby on Rails combines the scripting language Ruby with the Rails web application framework developed by the company 37Signals.
- With Software as a Service (SaaS) the software runs on servers elsewhere on the Internet.
  - You typically access the service through a browser.

1.18 Future of C++: Open Source Boost Libraries, TR1 and C++0x

- The main goals for the new standard are to make C++ easier to learn, improve library building capabilities, and increase compatibility with the C programming language.
- Chapter 23 considers the future of C++—we introduce the Boost C++ Libraries, Technical Report 1 (TR1) and C++0x.
- The Boost C++ Libraries are free, open source libraries created by members of the C++ community.
- Boost has grown to over 80 libraries, with more being added regularly.

1.19 Basic Object Technology Concepts

- We begin our introduction to object orientation with some key terminology.
- Humans think in terms of objects.
- We sometimes divide objects into two categories: animate and inanimate.
  - Animate objects are "alive" in some sense—they move around and do things.
  - Inanimate objects do not move on their own.
- Objects all have attributes (e.g., size, shape, color and weight), and they all exhibit behaviors (e.g., a ball rolls, bounces, inflates and deflates; a baby cries, sleeps, crawls, walks and blinks; a car accelerates, brakes and turns; a towel absorbs water).

1.19 Basic Object Technology Concepts (cont.)

- Object-oriented design (OOD) models software in terms similar to those that people use to describe real-world objects.
  - Takes advantage of class relationships, where objects of a certain class, have the same characteristics.
  - Takes advantage of inheritance relationships, where new classes of objects are derived by absorbing characteristics of existing classes and adding unique characteristics of their own.
- Object-oriented design provides a natural and intuitive way to view the software design process.
- OOD also models communication between objects.

1.19 Basic Object Technology Concepts (cont.)

- OOD encapsulates attributes and operations into objects.
- Objects have the property of information hiding.
  - Objects may know how to communicate with one another across well-defined interfaces, but normally they're not allowed to know how other objects are implemented.
  - Information hiding is crucial to good software engineering.

1.19 Basic Object Technology Concepts (cont.)

- Languages like C++ are object oriented.
- Programming in such a language is called object-oriented programming (OOP), and it allows computer programmers to implement object-oriented designs as working software systems.
- C++ classes contain functions that implement operations and data that implements attributes.

1.19 Basic Object Technology Concepts (cont.)

- C++ programmers concentrate on creating their own user-defined types called classes.
- Each class contains data as well as the set of functions that manipulate that data and provide services to clients (i.e., other classes or functions that use the class).
- The data components of a class are called data members.
- The function components of a class are called member functions.
- The nouns in a system specification help the C++ programmer determine the set of classes from which objects are created that work together to implement the system.
  - Classes are to objects as blueprints are to houses.
  - Just as we can build many houses from one blueprint, we can instantiate (create) many objects from one class.

1.19 Basic Object Technology Concepts (cont.)

- Classes can have relationships—called associations—with other classes.
- Packaging software as classes makes it possible for future software systems to reuse the classes.
- Groups of related classes are often packaged as reusable components.
- Each new class you create will have the potential to become a valuable *software asset* that you and other programmers can reuse to speed and enhance the quality of future software development efforts.

Software Engineering Observation 1.4

Reuse of existing classes when building new classes and programs saves time and money. 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.

1.19 Basic Object Technology Concepts (cont.)

- To create the best solutions, you should follow a process for analyzing your project's requirements (i.e., determining what the system should do) and developing a design that satisfies them (i.e., deciding *how* the system should do it).
- If this process involves analyzing and designing your system from an object-oriented point of view, it's called an object-oriented analysis and design (OOAD) process.
- OOAD is the generic term for the process of analyzing a problem and developing an approach for solving it.
- Although many different OOAD processes exist, a single graphical language for communicating the results of any OOAD process has come into wide use.
- This language, known as the Unified Modeling Language (UML), was developed in the mid-1990s. You'll learn a simple subset of the UML in this course.

1.19 Basic Object Technology Concepts (cont.)

- In 1994, James Rumbaugh joined Grady Booch at Rational Software Corporation (now a division of IBM), and they began working to unify their design processes.
- They soon were joined by Ivar Jacobson.
- In 1996, the group released early versions of the UML to the software engineering community and requested feedback.
- Around the same time, an organization known as the Object Management Group™ (OMG™) invited submissions for a common modeling language.
  - The OMG (`www.omg.org`) is a nonprofit organization that promotes the standardization of object-oriented technologies by issuing guidelines and specifications, such as the UML.
- The UML is now the most widely used graphical representation scheme for modeling object-oriented systems.