CHAPTER 1
Introduction to Computers, Programs, and C++
Objectives
- To review computer basics, programs, and operating systems (§§1.2-1.4).
- (Optional) To represent numbers in binary, decimal, and hexadecimal (§1.5).
- To know the history of C++ (§1.6).
- To write a simple C++ program (§1.7).
- To understand the C++ program development cycle (§1.8).
- To develop C++ using C++Builder (§1.9).
- To develop C++ using command line tools on Windows (§1.10).
- To develop C++ using command line tools on Unix (§1.11).
1.1 Introduction

You use word processors to write documents, Web browsers to explore the Internet, and email programs to send email over the Internet. Word processors, browsers, and email programs are all examples of software that runs on computers. Software is developed using programming languages. C++ is a popular and powerful programming language. Most application software such as word processors, browsers, and email programs are developed using C++. This book will introduce you to developing programs using C++.

You are about to begin an exciting journey, learning a powerful programming language. Before the journey, it is helpful to review computer basics, programs, and operating systems, and to become familiar with number systems. You may skip the review in §§1.2–1.4 if you are familiar with such terms as CPU, memory, disks, operating systems, and programming languages. You may also skip §1.5 and use it as reference when you have questions regarding binary and hexadecimal numbers.

1.2 What Is a Computer?

A computer is an electronic device that stores and processes data. A computer includes both hardware and software. In general, hardware is the physical aspect of the computer that can be seen, and software is the invisible instructions that control the hardware and make it perform specific tasks. Computer programming consists of writing instructions for computers to perform. You can learn a programming language without knowing computer hardware, but you will be better able to understand the effect of the instructions in the program if you do. This section gives a brief introduction to computer hardware components and their functionality.

A computer consists of the following major hardware components, as shown in Figure 1.1.

- Central Processing Unit (CPU)
- Memory (main memory)
- Storage Devices (disks, CDs, tapes)
- Input and Output Devices (monitors, keyboards, mice, printers)
- Communication Devices (modems and network interface cards (NICs))
A computer consists of a CPU, memory, hard disk, floppy disk, monitor, printer, and communication devices.

The components are connected through a bus that is a subsystem for transferring data between the components.

1.2.1 Central Processing Unit
<Cside Remark: CPU>
The central processing unit (CPU) is the brain of a computer. It retrieves instructions from memory and executes them. The CPU usually has two components: a control unit and an arithmetic/logic unit. The control unit controls and coordinates the actions of the other components. The arithmetic and logic unit performs numeric operations (addition, subtraction, multiplication, division) and logical operations (comparisons).

<Cside Remark: speed>
<Cside Remark: hertz>
<Cside Remark: megahertz>
<Cside Remark: gigahertz>
Today's CPU is built on a small silicon semiconductor chip with millions of transistors. The speed of the CPU is mainly determined by clock speed. Every computer has an internal clock. The faster the clock speed, the more instructions are executed in a given period of time. The clock emits electronic pulses at a constant rate, and these are used to control and synchronize the pace of operations. The unit of measurement is called a hertz (Hz), with 1 hertz equaling 1 pulse per second. The clock speed of computers is usually measured in megahertz (MHz) (1 MHz is 1 million Hz). The speed of the CPU has been improved continuously. If you buy a PC now, you can get an Intel Pentium 4 Processor at 3 gigahertz (GHz) (1 GHz is 1000 MHz).

1.2.2 Memory
<Cside Remark: bit>
<Cside Remark: byte>
Computers use zeros and ones because digital devices have two stable states, referred to as zero and one by convention. Data of various kinds, such as numbers, characters, and strings, are encoded as a series of bits (binary digits: zeros and ones). Memory stores data and program instructions for the CPU to execute. A memory unit is an ordered sequence of bytes, each holding eight bits, as shown in Figure 1.2.

- 2000 01001010 Encoding for character 'J'
- 2001 01100001 Encoding for character 'a'
- 2002 01110110 Encoding for character 'v'
- 2003 01100001 Encoding for character 'a'
- 2004 00000011 Encoding for number 3

**Figure 1.2**
Memory stores data and program instructions.

The programmer needs not to be concerned about the encoding and decoding of data, which is performed automatically by the system based on the encoding scheme. The encoding scheme varies. For example, character 'J' is represented by 01001010 in one byte in the popular ASCII encoding. A small number such as 3 can be stored in a single byte. If a computer needs to store a large number that cannot fit into a single byte, it uses several adjacent bytes. No two data items can share or split the same byte. A byte is the minimum storage unit.

A program and its data must be brought to memory before they can be executed. A memory byte is never empty, but its initial content may be meaningless to your program. The current content of a memory byte is lost whenever new information is placed in it.

**<Side Remark: RAM>**
**<Side Remark: megabyte>**
**<Side Remark: megabyte URL>**
Every byte has a unique address. The address is used to locate the byte for storing and retrieving data. Since the bytes can be accessed in any order, the memory is also referred to as RAM (random-access memory). Today's personal computers usually have at least 128 megabytes of RAM. A megabyte (abbreviated MB) is about 1 million bytes. For a
precise definition of megabyte, please see http://en.wikipedia.org/wiki/Megabyte. Like the CPU, memory is built on silicon semiconductor chips containing thousands of transistors embedded on their surface. Compared to the CPU chips, memory chips are less complicated, slower, and less expensive.

1.2.3 Storage Devices

Memory is volatile, because information is lost when the power is turned off. Programs and data are permanently stored on storage devices and are moved to memory when the computer actually uses them. The reason for this is that memory is much faster than storage devices. There are four main types of storage devices:

- Disk drives (hard disks and floppy disks)
- CD drives (CD-R, CD-RW, and DVD)
- Tape drives
- USB flash drives

<Side Remark: drive>
Drives are devices for operating a medium, such as disks, CDs, and tapes.

1.2.3.1 Disks
<Side Remark: hard disk>
<Side Remark: floppy disk>
There are two kinds of disks: hard disks and floppy disks. Personal computers usually have a 3.5-inch floppy disk drive and a hard drive. A floppy disk has a fixed capacity of about 1.44 MB. Hard disk capacities vary. The capacity of the hard disks of the latest PCs is in the range of 30 gigabytes to 160 gigabytes. Hard disks provide much faster performance and larger capacity than floppy disks. Both disk drives are often encased inside the computer. A floppy disk is removable. A hard disk is mounted inside the case of the computer. Removable hard disks are also available. Floppy disks will eventually be replaced by CD-RW and flash drives.

1.2.3.2 CDs and DVDs
<Side Remark: CD-R>
<Side Remark: CD-RW>
CD stands for compact disk. There are two types of CD drives: CD-R and CD-RW. A CD-R is for read-only permanent storage, and the user cannot modify its contents once they are recorded. A CD-RW can be used like a floppy disk, and thus can be read and rewritten. A single CD can hold up to 700MB. Most software is distributed through CD-ROMs. Most
new PCs are equipped with a CD-RW drive that can work with both CD-R and CD-RW.

DVD stands for digital versatile disc. DVDs and CDs look alike. You can store data using a CD or DVD. A DVD can hold more information than a CD. A standard DVD storage is 4.7GB in capacity.

1.2.3.3 Tapes
Tapes are mainly used for backup of data and programs. Unlike disks and CDs, tapes store information sequentially. The computer must retrieve information in the order it was stored. Tapes are very slow. It would take one to two hours to back up a 1-gigabyte hard disk.

1.2.3.4 USB Flash Drives
USB flash drives are popular new devices for storing and transporting data. They are small – about the size of a pack of gum. They act like a portable hard disk, which can be plugged into the USB port of your computer. USB flash drives are currently available with up to 2GB storage capacity.

1.2.4 Input and Output Devices

Input and output devices let the user communicate with the computer. The common input devices are keyboards and mice. The common output devices are monitors and printers.

1.2.4.1 The Keyboard
A computer keyboard resembles a typewriter keyboard except that it has extra keys for certain special functions.

Function keys are located at the top of the keyboard with prefix F. Their use depends on the software.

A modifier key is special key (e.g., Shift, Alt, Ctrl) that modifies the normal action of another key when the two are pressed in combination.

Numeric keypad, located on the right-hand corner of the keyboard, is a separate set of number keys for quick input of numbers.

Arrow keys, located between the main keypad and the numeric keypad, are used to move the cursor up, down, left, and right.
Insert, delete, page up, page down keys, located above the arrow keys, are used in word processing for performing insert, delete, page up, and page down.

1.2.4.2 The Mouse
A mouse is a pointing device. It is used to move an electronic pointer called a cursor around the screen or to click on an object on the screen to trigger it to respond.

1.2.4.3 The Monitor
The monitor displays information (text and graphics). The resolution and dot pitch determine the quality of the display.

<Side Remark: screen resolution>
The resolution specifies the number of pixels per square inch. Pixels (short for “picture elements”) are tiny dots that form an image on the screen. A common resolution for a 17-in screen, for example, is 1024 pixels wide and 768 pixels high. The resolution can be set manually. The higher the resolution, the sharper and clearer the image is.

<Side Remark: dot pitch>
The dot pitch is the amount of space between pixels. Typically, it has a range from 0.21 to 0.81 millimeters. The smaller the dot pitch, the better the display.

1.2.5 Communication Devices

<Side Remark: modem>
<Side Remark: DSL>
<Side Remark: NIC>
<Side Remark: LAN>
<Side Remark: mbps>
Computers can be networked through communication devices. The commonly used communication devices are the dialup modem, DSL, cable modem, and network interface card. A dialup modem uses a phone line and can transfer data at a speed up to 56,000 bps (bits per second). A DSL (digital subscriber line) also uses a phone line and can transfer data at a speed 20 times faster than a dialup modem. A cable modem uses the TV cable line maintained by the cable company. A cable modem is as fast as DSL. A network interface card (NIC) is a device that connects a computer to a local area network (LAN). The LAN is commonly used in business, universities, and government organizations. A typical NIC, called 10BaseT, can transfer data at 10 mbps (million bits per second).
1.3 Programs

<Side Remark: software>
Computer programs, known as software, are instructions to the computer. You tell a computer what to do through programs. Without programs, a computer is an empty machine. Computers do not understand human languages, so you need to use computer languages to communicate with them.

<Side Remark: machine language>
The language a computer speaks is the computer's native language or machine language. The machine language is a set of primitive instructions built into every computer. Machine languages are different for different types of computers. The instructions are in the form of binary code, so you have to enter binary codes for various instructions. Programming using a native machine language is a tedious process. Moreover, the programs are highly difficult to read and modify. For example, to add two numbers, you might have to write an instruction in binary like this:

1101101010011010

<Side Remark: assembly language>
Assembly language is a low-level programming language in which a mnemonic is used to represent each of the machine language instructions. For example, to add two numbers, you might write an instruction in assembly code like this:

ADDF3 R1, R2, R3

<Side Remark: assembler>
Assembly languages were developed to make programming easy. Since the computer cannot understand assembly language, however, a program called assembler is used to convert assembly language programs into machine code, as shown in Figure 1.3.

Figure 1.3
Assembler translates assembly language instructions to machine code.

Since assembly language is machine-dependent, an assembly
program can only be executed on a particular machine. Assembly programs are written in terms of machine instructions with easy-to-remember mnemonic names. The high-level languages were developed in order to overcome the platform-specific problem and make programming easier.

<Side Remark: high-level language>
The high-level languages are English-like and easy to learn and program. Here, for example, is a high-level language statement that computes the area of a circle with radius 5:

\[
\text{area} = 5 \times 5 \times 3.1415;
\]

There are over one hundred high-level languages. The popular languages used today are:

- COBOL (COmmon Business Oriented Language)
- FORTRAN (FORmula TRANslation)
- BASIC (Beginner All-purpose Symbolic Instructional Code)
- Pascal (named for Blaise Pascal)
- Ada (named for Ada Lovelace)
- Visual Basic (Basic-like visual language developed by Microsoft)
- Delphi (Pascal-like visual language developed by Borland)
- C (whose developer designed B first)
- **C++ (an object-oriented language, based on C)**
- Java
- C# (a Java-like language developed by Microsoft)

Each of these languages was designed for a specific purpose. COBOL was designed for business applications and now is used primarily for business data processing. FORTRAN was designed for mathematical computations and is used mainly for numeric computations. BASIC, as its name suggests, was designed to be learned and used easily. Ada was developed for the Department of Defense and is mainly used in defense projects. Visual Basic and Delphi are used in developing graphical user interfaces and in rapid application development. C combines the power of an assembly language with the ease of use and portability of a high-level language. C++ is popular for system software projects like writing compilers and operating systems. The Microsoft Windows operating system was coded using C++. Java, developed by Sun Microsystems, is widely used for developing Internet applications. C# (pronounced C sharp) is a new language developed by Microsoft for developing applications based on Microsoft .NET platform.
A program written in a high-level language is called a **source program**. Since a computer cannot understand a source program, a program called a **compiler** is used to translate the source program into a machine-language program. The machine-language program is often then linked with other supporting library code to form an executable file. The executable file can be executed on the machine, as shown in Figure 1.4. On Windows, executable files have extension .exe.

**Figure 1.4**
A source program is compiled into a machine-language file, which is then linked with the system library to form an executable file.

### 1.4 Operating Systems

**<Side Remark: OS>**
The operating system (OS) is the most important program that runs on a computer to manage and control its activities. You are probably using Windows (98, NT, 2000, XP, or ME), Mac OS, or Linux. Windows is currently the most popular PC operating system. Application programs, such as a Web browser or a word processor, cannot run without an operating system. The interrelationship of hardware, operating system, application software, and the user is shown in Figure 1.5.

**Figure 1.5**
The operating system is the software that controls and manages the system.

The major tasks of the operating systems are:
• Controlling and monitoring system activities
• Allocating and assigning system resources
• Scheduling operations

1.4.1 Controlling and Monitoring System Activities

Operating systems are responsible for security, ensuring that unauthorized users do not access the system. Operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the monitor, keeping track of files and directories on the disk, and controlling peripheral devices, such as disk drives and printers. Operating systems also make sure that different programs and users running at the same time do not interfere with one another.

1.4.2 Allocating and Assigning System Resources

The OS is responsible for determining what computer resources (CPU, memory, disks, input and output devices) a program needs and for allocating and assigning them to run the program.

1.4.3 Scheduling Operations

The OS is responsible for scheduling programs to use the system resources efficiently. Many of today's operating systems support such techniques as multiprogramming, multithreading, or multiprocessing to increase system performance.

.Side Remark: multiprogramming>

Multiprogramming allows multiple programs to run simultaneously by sharing of the CPU. The CPU is much faster than the other components. As a result, it is idle most of the time; for example, while waiting for data to be transferred from the disk or from other sources. A multiprogramming OS takes advantage of this by allowing multiple programs to use the CPU when it would otherwise be idle. For example, you may use a word processor to edit a file while the Web browser is downloading a file at the same time.

.Side Remark: multithreading>

Multithreading allows concurrency within a program, so that its subunits can run at the same time. For example, a word-processing program allows users to edit text and save it to a file at the same time. In this example, editing and saving are two tasks within the same application. These two tasks may run on separate threads concurrently.
Multiprocessing, or parallel processing, uses two or more processors together to perform a task. It is like a surgical operation where several doctors work together on one patient.

NOTE: You can skip this section and use it as reference when you have questions regarding binary and hexadecimal numbers.

Computers use binary numbers internally because storage devices like memory and disk are made to store 0s and 1s. A number or a character inside a computer is stored as a sequence of 0s and 1s. Each 0 or 1 is called a bit. The binary number system has two digits, 0 and 1.

Since we use decimal numbers in our daily life, binary numbers are not intuitive. When you write a number like 20 in a program, it is assumed to be a decimal number. Internally, computer software is used to convert decimal numbers into binary numbers, and vice versa.

Most of time, you write programs using decimal number systems. However, if you write programs to deal with a system like an operating system, you need to use binary numbers to reach down to the "machine-level." Binary numbers tend to be very long and cumbersome. Hexadecimal numbers are often used to abbreviate binary numbers, in which each hexadecimal digit represents exactly four binary digits. The hexadecimal number system has sixteen digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The letters A, B, C, D, E, and F correspond to the decimal numbers 10, 11, 12, 13, 14, and 15.

The digits in the decimal number system are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. A decimal number is represented using a sequence of one or more of these digits. The value that each digit in the sequence represents depends on its position. A position in a sequence has a value that is an integral power of 10. For example, the digits 7, 4, 2, and 3 in decimal number 7423 represent 7000, 400, 20, and 3, respectively, as shown below:
The decimal number system has ten digits and the position values are integral powers of 10. We say that 10 is the base or radix of the decimal number system. Similarly, the base of the binary number system is 2 since the binary number system has two digits and the base of the hex number system is 16 since the hex number system has sixteen digits.

1.5.1 Conversions between Binary Numbers and Decimal Numbers

Given a binary number \( b.b_2...b_b b_n \), the equivalent decimal value is

\[
b \times 2^n + b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \ldots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0
\]

The following are examples of converting binary numbers to decimals:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Conversion Formula</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>( 1 \times 2^1 + 0 \times 2^0 )</td>
<td>2</td>
</tr>
<tr>
<td>1000</td>
<td>( 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 )</td>
<td>8</td>
</tr>
<tr>
<td>10101011</td>
<td>( 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 )</td>
<td>171</td>
</tr>
</tbody>
</table>

To convert a decimal number \( d \) to a binary number is to find the bits \( b_n, b_{n-1}, b_{n-2}, \ldots, b_2, b_1, \) and \( b_0 \) such that

\[
d = b_n \times 2^n + b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \ldots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0
\]

These bits can be found by successively dividing \( d \) by 2 until the quotient is 0. The remainders are \( b_0, b_1, b_2, \ldots, b_{n-2}, b_{n-1}, \) and \( b_n \). For example, the decimal number 123 is 111011 in binary. The conversion is done as follows:
TIP
The Windows Calculator, as shown in Figure 1.6, is a useful tool for performing number conversions. To run it, choose Programs, Accessories, and Calculator from the Start button.

![Windows Calculator](image)

**Figure 1.6**
You can perform number conversions using the Windows Calculator.

***End of TIP***

1.5.2 Conversions Between Hexadecimal Numbers and Decimal Numbers

<Side Remark: hex to decimal>

Given a hexadecimal number \( h_nh_{n-1}h_{n-2} \ldots h_2h_1h_0 \), the equivalent decimal value is

\[
h_n \times 16^n + h_{n-1} \times 16^{n-1} + h_{n-2} \times 16^{n-2} + \ldots + h_2 \times 16^2 + h_1 \times 16^1 + h_0 \times 16^0
\]

The following are examples of converting hexadecimal numbers to decimals:

***Same table as in Intro5E p12***

<table>
<thead>
<tr>
<th>Hexadecimal</th>
<th>Conversion Formula</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>7F</td>
<td>(7 \times 16^1 + 15 \times 16^0)</td>
<td>127</td>
</tr>
<tr>
<td>FFFF</td>
<td>(15 \times 16^3 + 15 \times 16^2 + 15 \times 16^1 + 15 \times 16^0)</td>
<td>65535</td>
</tr>
</tbody>
</table>
**<Side Remark: decimal to hex>**

To convert a decimal number $d$ to a hexadecimal number is to find the hexadecimal digits $h_n$, $h_{n-1}$, $h_{n-2}$, ..., $h_2$, $h_1$, and $h_0$ such that

$$d = h_n \times 16^n + h_{n-1} \times 16^{n-1} + h_{n-2} \times 16^{n-2} + \ldots + h_2 \times 16^2 + h_1 \times 16^1 + h_0 \times 16^0$$

These numbers can be found by successively dividing $d$ by 16 until the quotient is 0. The remainders are $h_0$, $h_1$, $h_2$, ..., $h_{n-2}$, $h_{n-1}$, and $h_n$.

For example, the decimal number 123 is 7B in hexadecimal. The conversion is done as follows:

![Conversion Diagram](https://via.placeholder.com/150)

1.5.3 Conversions Between Binary Numbers and Hexadecimal Numbers

**<Side Remark: hex to binary>**

To convert a hexadecimal number to a binary number, simply convert each digit in the hexadecimal number into a four-digit binary number using Table 1.1.

***Same table as in Intro5E p13***

**Table 1.1**

Converting Hexadecimal to Binary

<table>
<thead>
<tr>
<th>Hexadecimal</th>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>A</td>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>1100</td>
<td>12</td>
</tr>
</tbody>
</table>
For example, the hexadecimal number 7B is 1111011, where 7 is 111 in binary, and B is 1011 in binary.

.Side Remark: binary to hex
To convert a binary number to a hexadecimal, convert every four binary digits from right to left in the binary number into a hexadecimal number.

For example, the binary number 1110001101 is 38D, since 1101 is D, 1000 is 8, and 11 is 3, as shown below.

```
1 1 1 0 0 0 1 1 0 1
   \   \   \   \  
    3   8   D
```

NOTE: Octal numbers are also useful. The octal number system has eight digits, 0 to 7. A decimal number 8 is represented as 10 in the octal system.

1.6 History of C++
C, C++, Java, and C# are very similar. C++ evolved from C. Java was modeled after C++. C# is a subset of C++ with some features similar to Java. If you know one of these languages, it is easy to learn the others.

.Side Remark: BCPL
.Side Remark: B
.Side Remark: C
C evolved from the B language and the B language evolved from the BCPL language. BCPL was developed by Martin Richards in the mid-1960s for writing operating systems and compilers. Ken Thompson incorporated many features from BCPL in his B language and used it to create early versions of the UNIX operating system at Bell Laboratories in 1970 on a DEC PDP-7 computer. Both BCPL and B were typeless, i.e., every data item occupies a fixed-length “word” or “cell” in memory. How a data item is treated, for example, as a number or a string, is the responsibility of the programmer. Dennis Ritchie extended the B language by adding types and other
features in 1971 to develop the UNIX operating system on a
DEC PDP-11 computer. Today, C is portable and hardware
independent. C is widely used for developing operating
systems.

**Side Remark: C++**

C++ is an extension of C, developed by Bjarne Stroustrup at
Bell Labs during 1983-1985. C++ added a number of features
that improved the C language. Most importantly, it added the
support of using classes for object-oriented programming.
Object-oriented programming can make programs easy to
develop and easy to maintain. C++ could be considered a
superset of C. The features of C are supported by C++. C
programs can be compiled using C++ compilers. After learning
C++, you will be able to read and understand C programs as
well.

**Side Remark: ANSI standard**

An international standard for C++ was created by American
National Standards Institute (ANSI) in 1998. The ANSI
standard is an attempt to ensure that C++ is portable, i.e.,
your programs compiled using one vendor’s compiler can be
compiled without errors from any other vendors on any
platform. Since the standard has been around for a while,
all the major vendors now support the ANSI standard.
Nevertheless, the C++ compiler vendors may add proprietary
features into the compiler. So, it is possible that your
program may compile fine by one compiler, but have to modify
the code in order to be compiled by a different compiler.

1.7 A Simple C++ Program

Let us begin with a simple C++ program that displays the
message “Welcome to C++!” on the console. The program is
shown in Listing 1.1.

Listing 1.1 Welcome.cpp (Displaying a Message)

```cpp
#include <iostream>

int main()
```

```cpp
```
```cpp
// Display Welcome to C++ to the console
std::cout << "Welcome to C++!" << std::endl;
return 0;
```

**<Side Remark: line number>**
The line numbers are not part of the program, but are displayed for reference purposes. So, don’t type line numbers in your program.

**<Side Remark: directive>**
**<Side Remark: header file>**
Line 1 is a compiler directive that tells the compiler to include the iostream library in this program, which is needed to support console input and output. The iostream is also called a header file in C++, because it is included at the header or beginning of a program.

**<Side Remark: main function>**
Every C++ program is executed from a main function. A function is a construct that contains statements. The main function defined in lines 3-7 contains two statements. The statements are enclosed in a block that starts with a left brace, `{`, (line 4) and ends with a right brace, `}` (line 7). Every statement in the block must end with a semicolon (`;`).

**<Side Remark: console output>**
The statement in line 5 displays a message to the console. std::cout represents the console. The `<<` operator, referred to as the stream insertion operator, sends a string to the console. A string must be enclosed in quotation marks. std::endl outputs a new line and flushes the output buffer to ensure that the output is displayed immediately. Note that `endl` stands for `end line`.

**<Side Remark: successful exit>**
The statement
```
return 0;
```
is placed at the end of every main function to exit the program. The value 0 indicates that the program has terminated successfully.

**<Side Remark: comment>**
**<Side Remark: line comment>**
**<Side Remark: paragraph comment>**
Line 5 is a comment that documents what the program is and how the program is constructed. Comments help programmers to communicate and understand the program. Comments are not
programming statements and thus are ignored by the compiler. In C++, comments are preceded by two slashes (\//\) on a line, called a line comment, or enclosed between /* and */ on one or several lines, called a paragraph comment. When the compiler sees //, it ignores all text after // on the same line. When it sees /*, it scans for the next */ and ignores any text between /* and */.

Here are examples of the two types of comments:

```
// This application program prints Welcome to C++!
/* This application program prints Welcome to C++! */
/* This application program
prints Welcome to C++! */
```

### <Side Remark: keyword>

Keywords, or Reserved words, are words that have a specific meaning to the compiler and cannot be used for other purposes in the program. There are two keywords: int and return in this program.

#### NOTE

You are probably wondering about such points as why the main function is declared this way and why `std::cout << "Welcome to C++!" << std::endl` is used to display a message to the console.
Your questions cannot be fully answered yet. For the time being, you will just have to accept that this is how things are done. You will find the answers in the coming chapters.

#### NOTE

### <Side Remark: syntax rules>

Like any other programming language, C++ has its own syntax, and you need to write code that obeys the syntax rules. The C++ compiler will report syntax errors if your program violates the syntax rules. Pay close attention to the punctuation. The redirection symbol << is two <>’s. There are two colons (:) between std and cout. Every statement in the function ends with a semicolon (;).

#### CAUTION:

### <Side Remark: case-sensitive>

C++ source programs are case-sensitive. It would be wrong, for example, to replace `main` in the program with `Main`.

The program in Listing 1.1 displays one message. Once you understand the program, it is easy to extend it to display more messages. For example, you can rewrite the program to
display three messages, as shown in Listing 1.2.

Listing 1.2 Welcome1.cpp (Three Messages)
***PD: Please add line numbers (including space lines, true for all line numbers in the book) in the following code***
***Layout: Please layout exactly. Don’t skip the space. This is true for all source code in the book. Thanks, AU.

.Side Remark line 1: include library>
.Side Remark line 3: main function>
.Side Remark line 5: output>
.Side Remark line 9: successful return>
# include <iostream>

int main()
{
  std::cout << "Welcome to C++!" << std::endl;
  std::cout << "Welcome to C++Builder!" << std::endl;
  std::cout << "Welcome to C++ Compiler!" << std::endl;
  return 0;
}

1.8 C++ Program Development Cycle

You have to create your program and compile it before it can be executed. This process is repetitive, as shown in Figure 1.7. If your program has compilation errors, you have to fix them by modifying the program, and then recompile it. If your program has runtime errors or does not produce the correct result, you have to modify the program, recompile it, and execute it again.
Figure 1.7
The C++ programming-development process consists of creating/modifying source code, compiling, linking and executing programs.

NOTE
.Side Remark: .cpp source file>
A C++ source file typically ends with the extension .cpp. Some compilers may accept other file name extensions (e.g., .c, .cp, or .c), but you should stick with the .cpp extension to be compliant with all ANSI C++ compilers.

NOTE
.Side Remark: compiler command>
A C++ compiler command performs three tasks in sequence: preprocessing, compiling, and linking. The compiler first processes the directives. The directives start with the # sign. For example, the include statement in line 1 of Listing 1.1 is a directive to tell the compiler to include a library. The compiler then translates the source code into a machine code file called object

```cpp
#include <iostream>

int main()
{
    // Display Welcome to C++ to the console
    std::cout << "Welcome to C++!" << std::endl;
    return 0;
}
```
file, and finally it links the object file with supporting library files to form an executable file. On Windows, the object file is stored on disk with an .obj extension and the executable files are stored with an .exe extension. On UNIX, the object file has an .o extension and the executable files do not have file extensions.

You can develop a C++ program from a command line or from an IDE such as C++Builder and Microsoft Visual C++.NET. The following sections introduce how to develop a C++ program from C++Builder, Windows’s command line, UNIX console, and Visual C++.NET.

1.9 Developing C++ Programs Using C++Builder

<Side Remark: IDE>

C++Builder is a C++ development tool that provides an integrated development environment (IDE) for rapidly developing C++ programs. Editing, compiling, building, debugging, and online help are integrated in one graphical user interface. Just enter source code in one window or open an existing file in a window, then click a button, menu item, or function key to compile and run the program.

<Side Remark: C++BuilderX>

C++Builder is a product of Borland. The company is well-known for developing innovative IDE tools. You can download a free copy of the personal edition of C++BuilderX (X means version 10).

1.9.1 Downloading and Installing C++Builder

Here are the steps to download and install C++BuilderX:

1. Point your Web browser to http://www.borland.com/downloads/download_cbuilderx.html. You will see a list of products available for download, as shown in Figure 1.8.
2. Click the Personal link. You will need to register and fill out a short survey. After this, an activation file has been emailed to you. You will see the download options, as shown in Figure 1.9.
3. Download one that matches your operating system and language preference. Save the file on your disk. For the English version on Windows, the file is cbx1_personal_windows.zip.
4. Unzip the file into a temporary folder named c:\temp.
5. Click install.exe in c:\temp\c++builder\Windows (see Figure 1.10) to install C++Builder. Accept the license
term and choose all default setting to complete the installation. The default installation installs all files in c:\CBuilderX.

6. Recall that an activation file named reg389.txt was sent to you upon registration in Step 2. Before you can use the product, you have to place the activation file to c:\Documents and Settings\<username> if you are using Windows 2000 or Windows XP. For example, on my machine, I put the file into c:\Documents and Settings\administrator.

![Figure 1.8](image)

**Figure 1.8**

You can download C++Builder from this Web site.
Figure 1.9
You can click freecommmandLinetoools.exe to start downloading.

Figure 1.10
Clicking install.exe will install C++Builder.

1.9.2 Getting Started with C++Builder
The default installation automatically creates a desktop icon for C++Builder. You can launch C++Builder by double-clicking this icon. The C++Builder user interface appears, as shown in Figure 1.11.
The C++ user interface is a single window that performs functions for editing, compiling, debugging, and running programs.

The user interface primarily consists of the main menu, main toolbar, status bar, project pane, structure pane, and content pane.

**<Side Remark: main menu>**
The main menu is similar to that of other Windows applications and provides most of the commands you need to use C++Builder, including those for creating, editing, compiling, running, and debugging programs. The menu items are enabled and disabled in response to the current context.

**<Side Remark: toolbar>**
The toolbar provides buttons for several frequently used commands on the menu bar. Clicking a toolbar is faster than using the menu bar. For some commands, you also can use function keys or keyboard shortcuts. For example, you can save a file in three ways:

- Select File, Save from the menu bar.
- Click the "save" toolbar button.
- Use the keyboard shortcut Ctrl+S.

**TIP:**

**<Side Remark: tooltip>**
You can display a label, known as ToolTip, for a button by pointing the mouse to the button without clicking.

**<Side Remark: status bar>**
The status bar displays a message that alerts the user to the operation status, such as file saved for the Save file command and compilation successful for the Compilation command.

**<Side Remark: project pane>**
The project pane displays the files in the project. It consists of the following items, as shown in Figure 1.11.

- A small toolbar with four buttons (Close Project, Add To Project, Remove From Project, and Refresh).
- A drop-down list of all opened projects.
A tree view of all the files that make up the active project.

The project pane shows a list of one or more files. The project (.cbx) file appears first. Attached to it is a list of the files in the project. The list can include .exe, .cpp, .h, .html, text, or image files. You select a file in the project pane by clicking it. The content pane and the structure pane display information about the selected file. As you select different files in the project pane, each one will be represented in the content and structure panes.

The project pane shown in Figure 1.11 contains four files. The Add button is used to add new files to the project, and the Remove button to remove files from the project. For example, you can remove Welcome.html by selecting the file in the project pane and clicking the Remove button. You can then add the file back to the project as follows:

1. Click the Add button to display the Open dialog box, as shown in Figure 1.12.

2. Open Welcome.html. You will see Welcome.html displayed in the project pane.

![Add to “Welcome.cbx”](image)

Figure 1.12

The Open dialog box enables you to open an existing file.

**TIP:** You can select multiple files by clicking the files with the CTRL key pressed, or select consecutive files with the SHIFT key pressed.

*Side Remark: content pane*

The content pane displays all the opened files as a set of tabs. To open a file in the content pane, double-click it in the project pane. The content pane displays the detailed
content of the selected file. The editor or viewer used is determined by the file's extension. If you click the Welcome.html file in the project pane, for example, you will see three tabs (View, Source, and History) at the bottom of the content pane (see Figure 1.13(a)). If you select the Source tab, you will see the HTML editor. You can view and edit the HTML code in the content pane, as shown in Figure 1.13(b).

(a)                           (b)
Figure 1.13
(a) C++Builder renders HTML files in the content pane. (b) C++Builder can edit HTML files.

*Side Remark: structure pane*
The structure pane displays the structural information about the files you selected in the project pane. All the items displayed in the structure pane are in the form of a hierarchical indexed list. The expand symbol in front of an item indicates that it contains subitems. You can see the subitems by clicking on the expand symbol.

1.9.3 Creating a Project
A project is like a holder that ties all the files together. The information about each project is stored in a project file with a .cbx file extension. The project file contains a list of all the files and project settings and properties. C++Builder uses this information to load and save all the files in the project and compile and run the programs. To create and run a program, you have to first create a project.

Here are the steps to create a project:

1. Choose File, New to display Object Gallery, as shown in Figure 1.14(a).
2. In the Project tab, click New Console to display the New Console Application Wizard, as shown in Figure 1.14(b). Enter bookexample in the Name field and c:/example in the Directory field. Click Finish to create a project, as shown in Figure 1.15.

![Figure 1.14](image1.png)

(a) Object Gallery enables you to choose appropriate templates for projects and files. (b) The application wizard creates a project.

![Figure 1.15](image2.png)

A new project is created.

1.9.4 Creating, Compiling, and Running a C++ Program

.Side Remark: create files

After you created a project, you can create programs in the project. Here are the steps to create a C++ program for Listing 1.1:

1. Choose File, New File to display the Create New File window, as shown in Figure 1.16(a).
2. Enter Welcome in the Name field, select cpp in the type field, enter c:/example in the Directory field, and check the Add saved file to project option. Click OK to create Welcome.cpp, as shown in Figure 1.16(b).

3. Type in the code exactly from Listing 1.1, as shown in Figure 1.17.

![Figure 1.16](image)

(a) You can create a file using the Create New File window.  
(b) A new file is created and added in the project.

![Figure 1.17](image)

The source code for Welcome.cpp is typed in the content pane.

<Side Remark: compile>
To compile Welcome.cpp, use one of the following methods.  
(Be sure that Welcome.cpp is selected in the project pane.)

- Select Project, Make "Welcome.cpp" from the main menu.
• Click the Make toolbar button (\ discriminate button).

• Point to Welcome.cpp in the project pane, right-click the mouse button to display a popup menu, and choose Make from the menu. (I find this method most convenient.)

The compilation status is displayed on the status bar. If there are no syntax errors, the compiler generates several files, including an executable file named Welcome.exe.

**Side Remark: run**

To run the program, simply choose Run, Run Project from the main menu or click the Run toolbar button (\ discriminate). The output is displayed in the message pane, as shown in Figure 1.17.

**NOTE:**

**Side Remark: compile and run**

The Run command invokes the Compile command if the program is not compiled or was modified after the last compilation.

**TIP:**

**Side Remark: message pane**

If the message pane is not displayed, choose View, Messages to display it.

Suppose you wish to create another program named Welcome1.cpp that displays three messages as shown in Listing 1.2. Here are the steps to accomplish the task:

1. Choose File, New File to display the Create New File window, as shown in Figure 1.18(a).
2. Enter Welcome1 in the Name field, select cpp in the type field, enter c:/example in the Directory field, and check the Add saved file to project option. Click OK to create Welcome1.cpp. Type in the code exactly from Listing 1.2, as shown in Figure 1.18(b).
3. In order to run Welcome1.cpp, you need to remove Welcome.cpp from the project. To remove it, choose Welcome.cpp in the project pane and click the Remove button, as shown in Figure 1.18(b).

4. Click the Run toolbar button to run the program. The output is displayed in the message pane, as shown in Figure 1.19.

Output for Welcome1.cpp is displayed in the message pane.
NOTE:

.Side Remark: DOS commands>
To develop programs from Window’s command line, you need to know how to use DOS commands. Please see Supplement I.A, “Basic DOS Commands,” on how to use basic DOS commands. All the supplements are accessible from the Companion Website.

.Side Remark: compilers>
When you install C++Builder in the previous section, two popular compilers (Borland C++ compiler and GNU C++ compiler) are automatically installed. GNU is an organization devoted to develop open source software (see www.gnu.org). The Borland C++ compiler is in c:\CBUILDERX\bin\bcc32.exe and the GNU C++ compiler is in c:\CBUILDERX\bin\mingw\bin\g++.exe. To use them directly from the command line, you have to add c:\CBUILDERX\bin and c:\CBUILDERX\mingw\bin into PATH environment variable. Here are the steps to add the new paths in Windows 2000 and Windows XP:

1. Choose Systems from the Window’s Control Panel to display the Systems Properties dialog, as shown in Figure 1.20(a).
2. Choose Advanced tab and click Environments Variables to display the Environment Variables dialog as shown in Figure 1.20(b).
3. Choose Path in the System variable section and click Edit to add the paths.

Figure 1.20
You need to add compilers in the environment path.

<Side Remark: text editor>
You can use any text editor to create and edit a C++ source code file. Figure 1.21 shows how to use the NotePad to create and edit the source code file.

![Figure 1.21](image)
You can create a C++ source file using Windows NotePad.

<Side Remark: bcc32 Borland compiler>
To compile Welcome.cpp using the Borland compiler, type the command **bcc32 Welcome.cpp**, as shown in Figure 1.22. If no syntax errors, an executable file named Welcome.exe is created. You can run it by typing **Welcome**.

![Figure 1.22](image)
You can compile using the Borland C++ compiler.

<Side Remark: g++ GNU compiler>
To compile Welcome.cpp using the GNU compiler, type the command **g++ Welcome.cpp -o Main**, as shown in Figure 1.23. If no syntax errors, an executable file named Main.exe is created. You can run it by typing **Main**.

![Figure 1.23](image)
You can compile using the GNU C++ compiler.

1.11 Developing C++ Programs on UNIX

**NOTE:**

<Side Remark: DOS commands>
To develop programs on UNIX, you need to know how to use UNIX commands. Please see Supplement I.B, “Basic UNIX Commands,” on how to use basic UNIX commands.

<Side Remark: compilers>
By default, a GNU C++ compiler is automatically installed on UNIX. You can use the vi or emacs editor to create a C++ source code file. Figure 1.24(a) shows how to use the vi editor to create and edit the source code file.

![Figure 1.24](a) You can create a C++ source file using the vi editor. (b) You can compile a C++ source file using the GNU compiler.

<Side Remark: GNU compiler>
To compile Welcome.cpp using the GNU compiler, type the command `g++ Welcome.cpp -o Main`, as shown in Figure 1.24(b). If no syntax errors, an executable file named Main is created. You can run it by typing `.Main`, as shown in Figure 1.24(b).

1.12 Developing C++ Programs Using Visual C++.NET

<Side Remark: IDE>
Visual C++.NET is a component of Microsoft Visual Studio .NET for developing C++ programs. Like Borland C++Builder, Visual C++.NET provides an integrated development environment for rapidly developing C++ programs. This section introduces how to create a project, create a program, compile and run the program.

1.12.1 Creating a Project
To create C++ programs in C++.NET, you have to first create a project. A project is like a holder that ties all the files together. Here are the steps to create a project:
1. Click the Start button, choose All Programs, Microsoft Visual Studio .NET to launch Microsoft Visual Studio .NET.

2. Choose File, New, Project (see Figure 1.25) to display the New Project dialog window, as shown in Figure 1.26(a).

3. Choose Visual C++ in the project type column and Win32 Console Project in the Templates column. Type bookexample in the Name field and c:\smith in the Location field. Click OK to display The Win32 Application Wizard window, as shown in Figure 1.26(b).

4. Choose Console application and check the Empty project box. Click Finish to create a project. You will see the project named bookexample in the solution explorer, as shown in Figure 1.27.

Figure 1.25
You need to create a project before creating programs.

Figure 1.26
(a) Choose Win32 Console Project to create test project. (b) The Application Wizard creates a project.
Solution explorer shows the files in the project

Figure 1.27
A project is created for C++ console applications.

1.12.2 Creating a C++ Program

After you created a project, you can create programs in the project. Here are the steps to create a C++ program for Listing 1.1:

1. Choose File, Add New Item from the main menu, or choose Add, Add New Item from the context menu of the bookexample project (see Figure 1.28(a)) to display the Add New Item window, as shown in Figure 1.28(b).

2. Choose C++ File (.cpp) in the Templates column. Enter Enter Welcome in the Name field and c:\smith\bookexample\ in the Location field. Click Open to create the file, as shown in Figure 1.29(a).

3. Enter the code for Welcome.cpp exactly from Listing 1.1, as shown in Figure 1.29(b).
(a) You can open the Add New Item window from the project’s context menu. (b) You can specify the file type, name, and location to create a file.

Figure 1.29
(a) Welcome.cpp is created in the project. (b) The source code for Welcome.cpp is entered.

1.12.3 Compiling a C++ Program

After you created a program, you can compile it. You may compile it by choosing Build, Compile, or press Ctrl+F7, or choose Compile in the context menu for Welcome.cpp, as shown in Figure 1.30.

Figure 1.30
Choose the Compile command to compile the program.

1.12.4 Running a C++ Program

To run the program, choose Debug, Start Without Debugging, or press Ctrl+F5. You will see a dialog box, as shown in Figure 1.31(a). Click Yes to continue. You will see the output is displayed in a DOS window, as shown in Figure 1.31(b).
Figure 1.31
The output is displayed in a DOS window.

NOTE:
<Side Remark: compile and run>
The Run command invokes the Compile command if the program is not compiled or was modified after the last compilation.

NOTE:
<Side Remark: one main function>
Each project can have only one file that contains a main function. If you need to create another file with a main function, you have two options:

- Remove the current file that contains a main function from the project by choosing Remove from the context menu of the program, as shown in Figure 1.32. (Note that you can add a file to the project by choosing File, Add Existing Item.)

- Create a new project for the new program.

Figure 1.32
You can remove a file from a project.

***END NOTE***
Key Terms

***PD: Please place terms in two columns same as in intro5e.

- assembly language 7
- bcc32 command 21
- binary numbers 11
- bit 39
- block 25
- byte 5
- C++Builder 20
- cable modem 7
- central processing unit (CPU) 5
- comment 24
- compiler 9
- dot pitch 7
- DSL (digital subscriber line) 7
- g++ command 23
- hardware 4
- hexadecimal numbers 11
- high-level programming language 8
- Integrated Development Environment (IDE) 20
- keyword (or reserved word) 24
- machine language 7
- main function 21
- memory 5
- modem 7
- network interface card (NIC) 7
- operating system (OS) 9
- pixel 7
- resolution 7
- software 4
- source code 8
- source file 8
- storage devices 6
- statement 24
- stream insertion operator
- Visual C++ .NET

Chapter Summary
A computer is an electronic device that stores and processes data. A computer includes both hardware and software. In general, hardware is the physical aspect of the computer that can be seen, and software is the invisible instructions that control the hardware and make it perform tasks.

Computer programs, known as software, are instructions to the computer. You tell a computer what to do through programs. Computer programming consists of writing instructions for computers to perform.

The machine language is a set of primitive instructions built into every computer. Assembly language is a low-level programming language in which a mnemonic is used to represent each of the machine-language instructions.

High-level languages are English-like and easy to learn and program. There are over one hundred high-level languages. A program written in a high-level language is called a source program. Since a computer cannot understand a source program, a program called a compiler is used to translate the source program into a machine language program, which is then linked with other supporting library code to form an executable file.

The operating system (OS) is a program that manages and controls a computer’s activities. Application programs, such as Web browsers and word processors, cannot run without an operating system.

C++ is an extension of C, developed by Bjarne Stroustrup at Bell Labs during 1983-1985. C++ added a number of features that improved the C language. Most importantly, it added the support of using classes for object-oriented programming.

C++ source files end with the .cpp extension. You can develop C++ applications from the command line or using an IDE such as Borland C++Builder and Visual C++.NET.

Every C++ program is executed from a main function. A function is a construct that contains statements.

Review Questions

NOTE: Answers to review questions are on the Companion Website.

Sections 1.2 – 1.4

1.1 Define hardware and software.
1.2
Define machine language, assembly language, and high-level programming language.

1.3
What is an operating system?

**Section 1.5**

1.4
Convert the following decimal numbers into hexadecimal and binary numbers.
   100; 4340; 2000

1.5
Convert the following binary numbers into hexadecimal numbers and decimal numbers.
   1000011001; 100000000; 100111

1.6
Convert the following hexadecimal numbers into binary and decimal numbers.
   FEFA9; 93; 2000

**Sections 1.6 – 1.8**

1.7
Describe the history of C++. Can C++ run on any machine?
What is needed to compile and run C++ programs?

1.8
What are the input and output of a C++ compiler?

1.9
List some C++ development tools. Are tools like C++Builder and Visual C++.NET different languages from C++, or are they dialects or extensions of C++?

1.10
What is the relationship between C, C++, Java, and C#?

**Sections 1.9 – 1.11**

1.11
Explain the C++ keywords. List some C++ keywords you learned in this chapter.

1.12
Is C++ case-sensitive? What is the case for C++ keywords?

1.13
What is the C++ source filename extension, and what is the C++ executable filename extension on Windows?

1.14
What is a comment? What is the syntax for a comment in C++? Is the comment ignored by the compiler?

1.15
What is the statement to display a string on the console?

1.16
Identify and fix the errors in the following code:

***PD: Please add line numbers (including space lines) in the following code***
```cpp
#include <iostream>

int main
{
   // Display Welcome to C++ to the console
   std::cout << 'Welcome to C++!' << std::endl;
   return 0;
}
```

1.17
What is the command to compile a C++ program using the GNU
compiler? What is the command to run a C++ application on
Windows and on Unix?

**Programming Exercises**

1.1
*(Creating, compiling, and running a C++ program)* Create a
source file containing a C++ program. Perform the following
steps to compile the program and run it:

1. Create a file named `Welcome.cpp` for Listing 1.1.
   You can use an IDE or any editor that will save
   your file in text format.
2. Compile the source file.
3. Run the program.
4. Replace "Welcome to C++" with "My first program"
   in the program; save, compile, and run the
   program. You will see the message "My first
   program" displayed.
5. Replace `main` with `Main`, and recompile the source
   code. The compiler returns an error message
   because the C++ program is case-sensitive.
6. Change it back, and compile the program again.