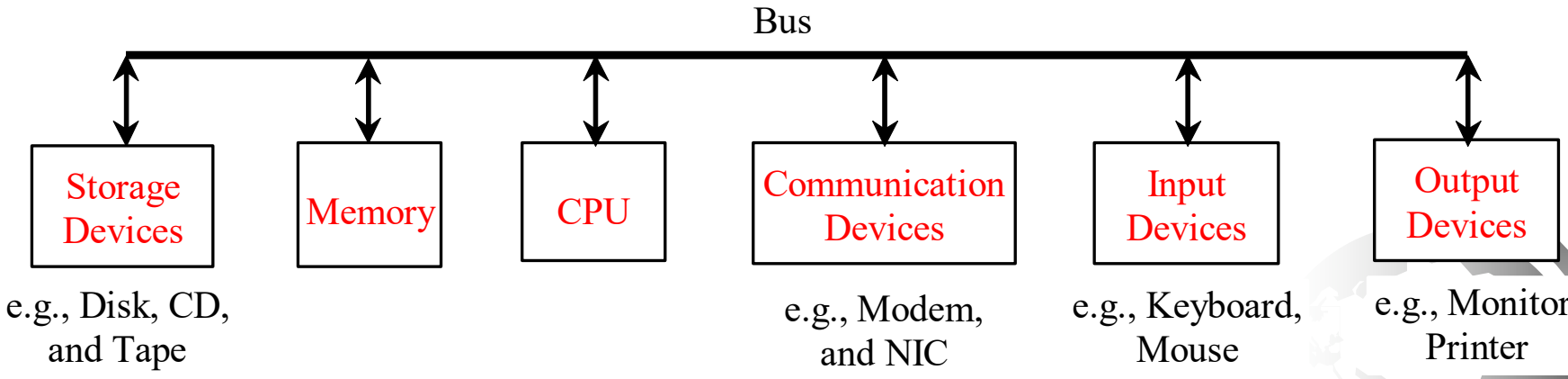
Chapter 1 Introduction to Computers, Programs, and C++

Objectives

- To understand computer basics, programs, and operating systems (§§1.2–1.4).
- To describe the history of C++ (§1.5).
- To write a simple C++ program for console output (§1.6).
- To understand the C++ program-development cycle (§1.7).
- To know programming style and documentation (§1.8).
- To explain the differences between syntax errors, runtime errors, and logic errors (§1.9).

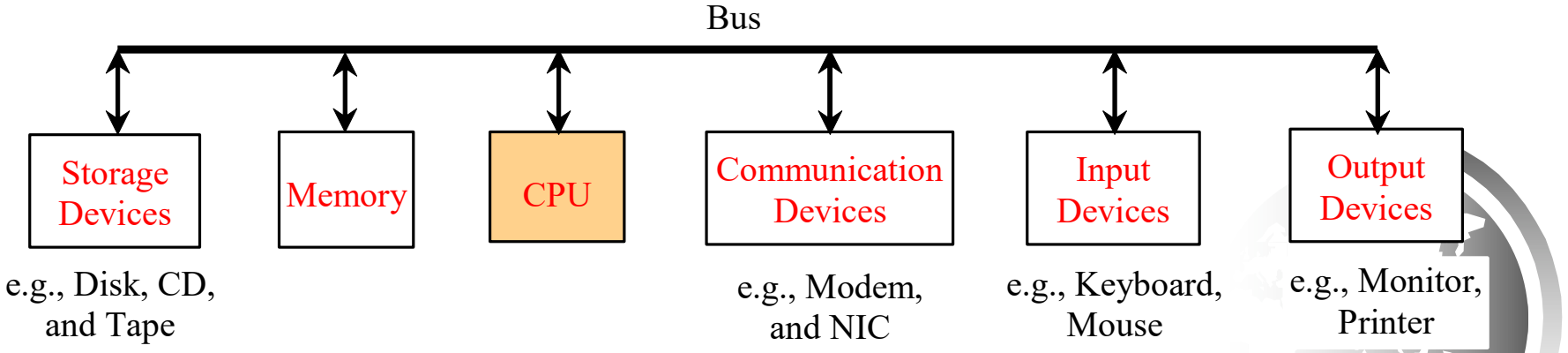
What is a Computer?

A computer consists of a CPU, memory, hard disk, monitor, printer, and communication devices.



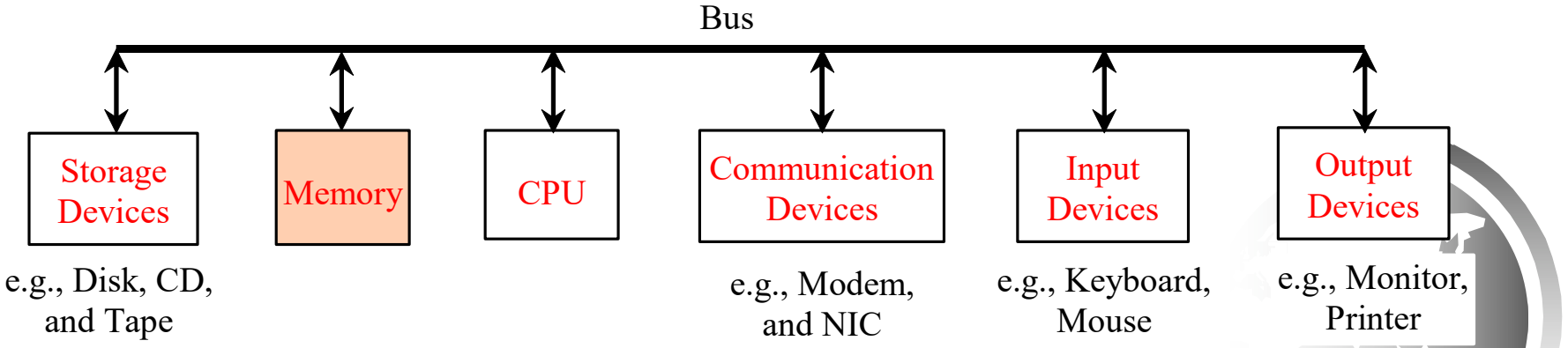
CPU

The central processing unit (CPU) is the brain of a computer. It retrieves instructions from memory and executes them. The CPU speed is measured in megahertz (MHz), with 1 megahertz equaling 1 million pulses per second. 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 (1 gigahertz is 1000 megahertz).



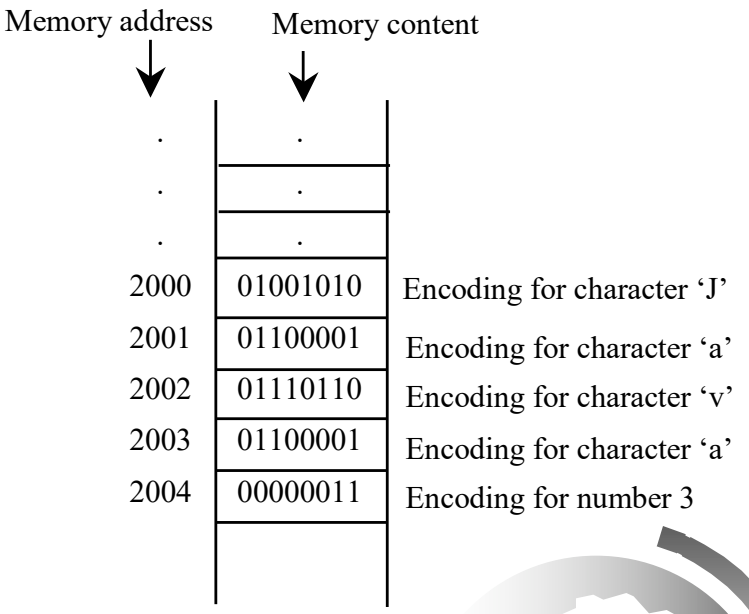
Memory

Memory is to store data and program instructions for CPU to execute. A memory unit is an ordered sequence of bytes, each holds eight bits. 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.



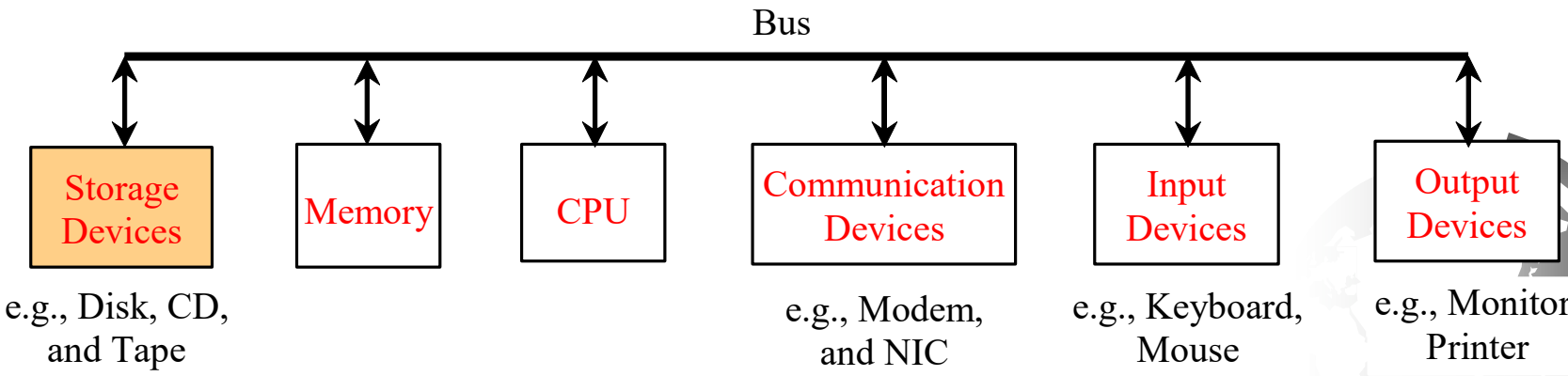
How Data is Stored?

Data of various kinds, such as numbers, characters, and strings, are encoded as a series of bits (zeros and ones). Computers use zeros and ones because digital devices have two stable states, which are referred to as *zero* and *one* by convention. The programmers need 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. A small number such as three can be stored in a single byte. If computer needs to store a large number that cannot fit into a single byte, it uses a number of adjacent bytes. No two data can share or split a same byte. A byte is the minimum storage unit.



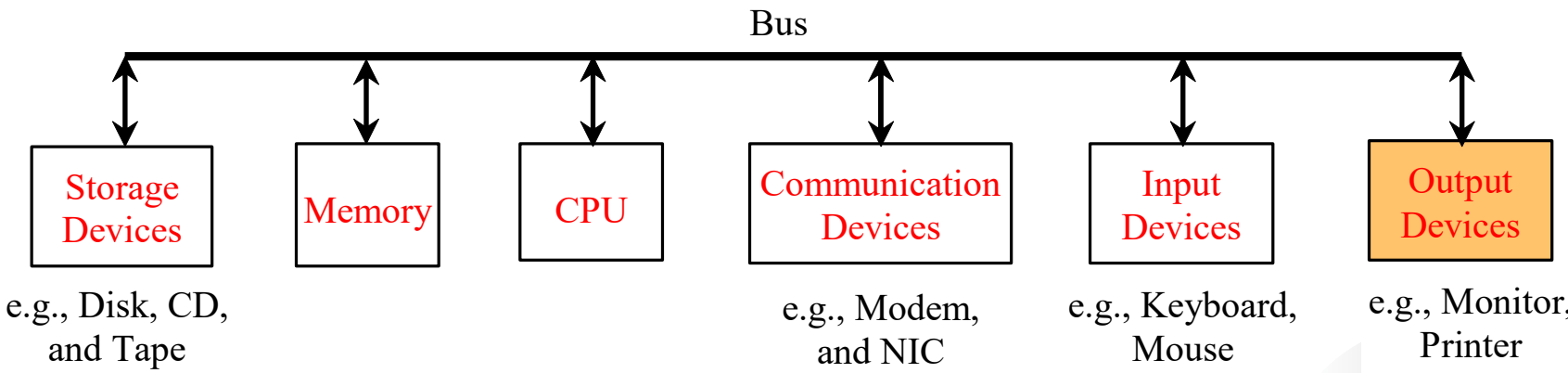
Storage Devices

Memory is volatile, because information is lost when the power is off. Programs and data are permanently stored on storage devices and are moved to memory when the computer actually uses them. There are three main types of storage devices:Disk drives (hard disks), CD drives (CD-R and CD-RW), and Tape drives.



Output Devices: Monitor

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



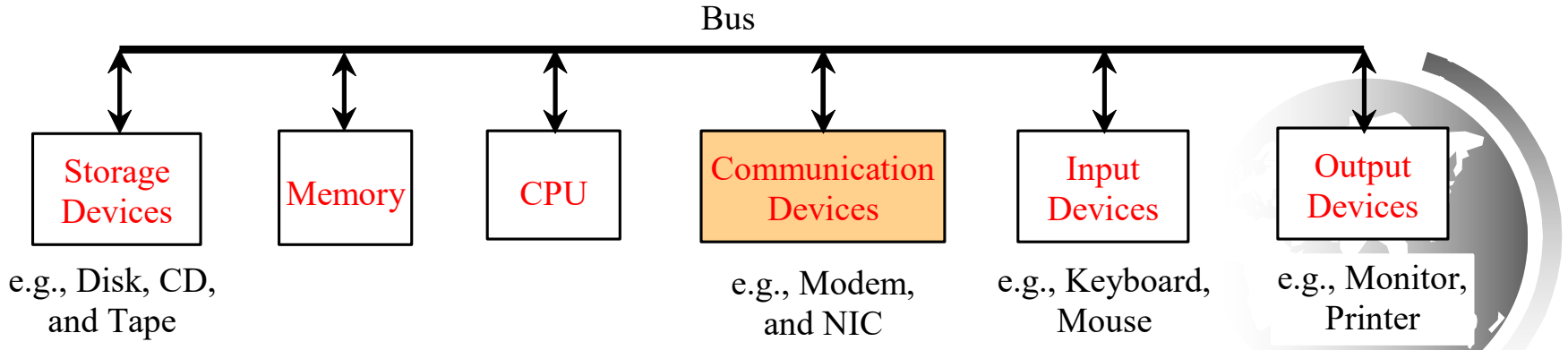
Monitor Resolution and Dot Pitch

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. The resolution can be set manually. The higher the resolution, the sharper and clearer the image is. However, the image may be very small if you set high resolution on a small screen monitor. PC monitors are usually 15-inch, 17-inch, 19-inch, or 21-inch. For a 15-inch monitor, a comfortable resolution setting would be 640×480 (307,200 pixels).

dot pitch The *dot pitch* is the amount of space between pixels. The smaller the dot pitch, the better the display.

Communication Devices

A *regular modem* uses a phone line and can transfer data in a speed up to 56,000 bps (bits per second). A *DSL* (digital subscriber line) also uses a phone line and can transfer data in a speed 20 times faster than a regular modem. A *cable modem* uses the TV cable line maintained by the cable company. A cable modem is as fast as a DSL. Network interface card (*NIC*) is a device to connect a computer to a local area network (LAN). The LAN is commonly used in business, universities, and government organizations. A typical type of NIC, called *10BaseT*, can transfer data at 10 mbps (million bits per second).



Programs

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.

Programs are written using programming languages.

Programming Languages

Machine Language Assembly Language High-Level Language

Machine language is a set of primitive instructions built into every computer. The instructions are in the form of binary code, so you have to enter binary codes for various instructions. Program with 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 write an instruction in binary like this:

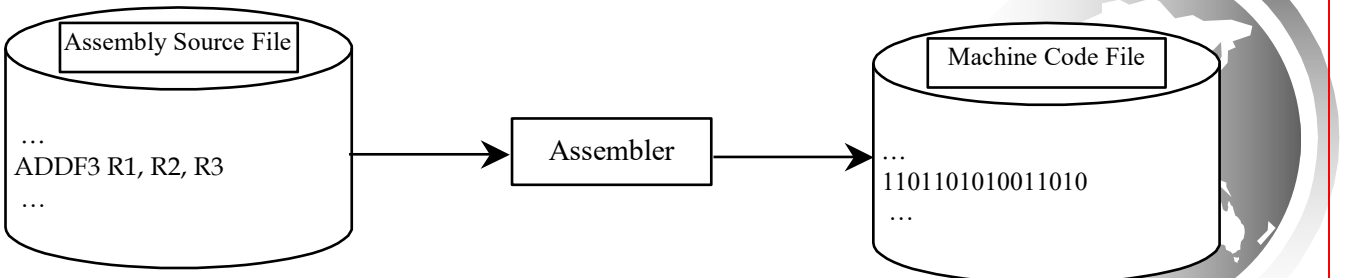
```
1101101010011010
```

Programming Languages

Machine Language Assembly Language High-Level Language

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. For example, to add two numbers, you might write an instruction in assembly code like this:

ADDF3 R1, R2, R3



Programming Languages

Machine Language Assembly Language High-Level Language

The high-level languages are English-like and easy to learn and program. For example, the following is a high-level language statement that computes the area of a circle with radius 5:

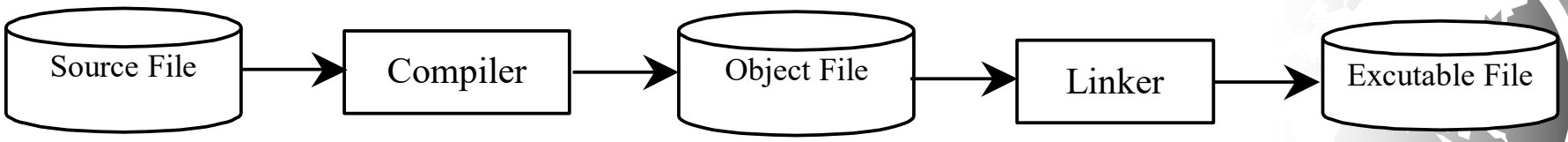
```
area = 5 * 5 * 3.1415;
```

Popular High-Level Languages

- ✦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)
- ✦C (whose developer designed B first)
- ✦Visual Basic (Basic-like visual language developed by Microsoft)
- ✦Delphi (Pascal-like visual language developed by Borland)
- ✦C++ (an object-oriented language, based on C)
- ✦Java (a popular object-oriented language, similar to C++)
- ✦C# (a Java-like developed my Microsoft)

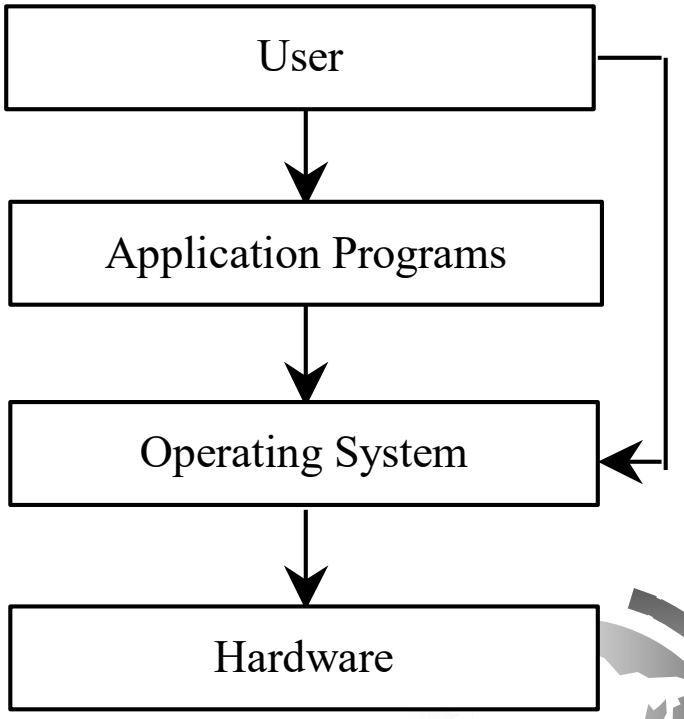
Compiling Source Code

A program written in a high-level language is called a s*ource program*. Since a computer cannot understand a source program. Program called a *compiler* is used to translate the source program into a machine language program called an *object program*. The object program is often then linked with other supporting library code before the object can be executed on the machine.



Operating Systems

The *operating system* (OS) is a program that manages and controls a computer's activities. You are probably using Windows 98, NT, 2000, XP, or ME. Windows is currently the most popular PC operating system. Application programs such as an Internet browser and a word processor cannot run without an operating system.



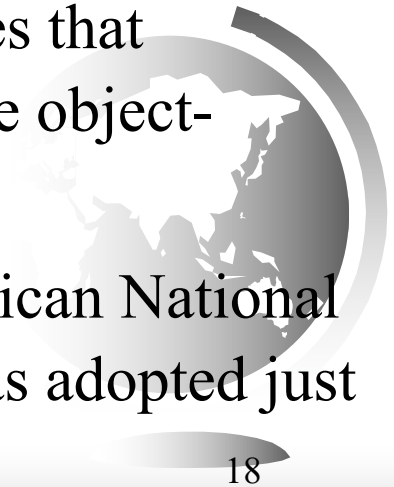
History of C++

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.

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.

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 object-oriented programming.

An international standard for C++ was created by American National Standards Institute (ANSI) in 1998 (C++98). C++11 was adopted just recently.



A Simple C++ Program

Let us begin with a simple C++ program that displays the message "Welcome to C++!" on the console.

```
#include <iostream>
using namespace std;
int main()
{
  // Display Welcome to C++ to the console
  cout << "Welcome to C++!" << endl;
  return 0;
}
```

Welcome

Note: Clicking the green button displays the source code with interactive animation and live run. Internet connection is needed for this button.

C++ IDE Tutorial

You can develop a C++ program from a command window or from an IDE. An IDE is software 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 or open an existing file in a window, then click a button, menu item, or function key to compile and run the program. Examples of popular IDEs are Microsoft Visual C++, Dev-C++, Eclipse, and NetBeans. All these IDEs can be downloaded free.

Extending the Simple C++ Program

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.

```
#include <iostream>
using namespace std;
int main()
{
  cout << "Programming is fun!" << endl;
  cout << "Fundamentals First" << endl;
  cout << "Problem Driven" << endl;
  return 0;
}
```

WelcomeWithThreeMessages

Computing with Numbers

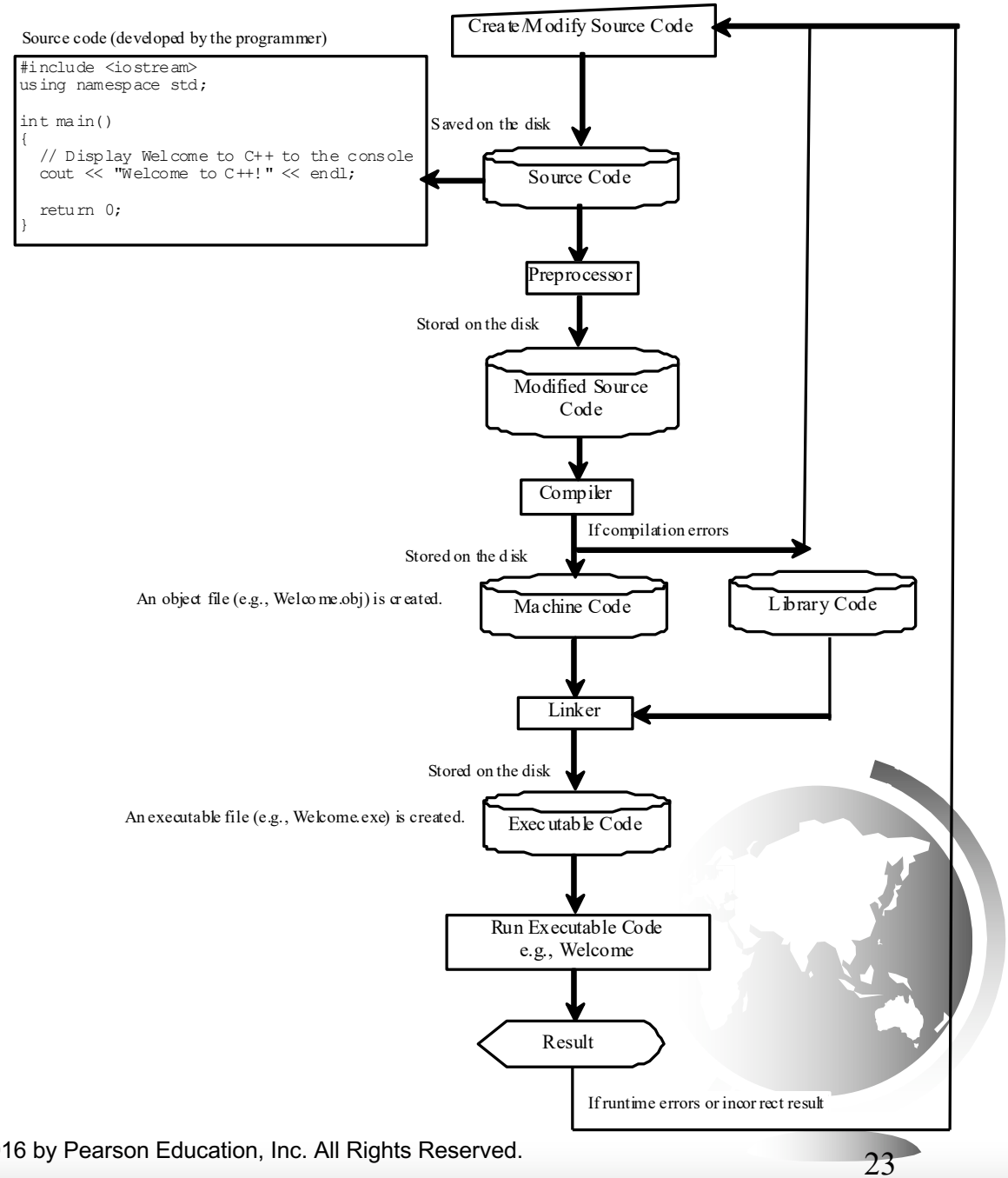
Further, you can perform mathematical computations and displays the result to the console. Listing 1.3 gives such an example.

```
#include <iostream>
using namespace std;
int main()
{
  cout << "(10.5 + 2 * 3) / (45 - 3.5) = ";
  cout << (10.5 + 2 * 3) / (45 - 3.5) << endl;

  return 0;
}
```

ComputeExpression

Creating, Compiling, and Running Programs



Programming Style and Documentation

- Appropriate Comments
- Proper Indentation and Spacing Lines
- Block Styles

Programming Errors

- Syntax Errors
- Runtime Errors
- Logic Errors

Syntax Errors

ShowSyntaxErrors

Runtime Errors

ShowRuntimeErrors

Logic Errors

ShowLogicErrors

Common Errors

- Common Error 1: Missing Braces
- Common Error 2: Missing Semicolons
- Common Error 3: Missing Quotation Marks
- Common Error 4: Misspelling Names