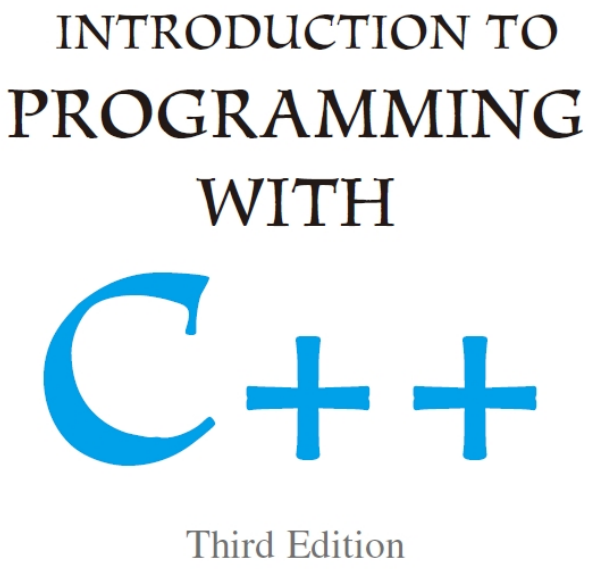
Introduction to Programming with C++

Chieh-Sen (Jason) Huang

Department of Applied Mathematics

National Sun Yat-sen University



Contents are based on book by Y. Daniel Liang

Topic Overview

- Part I: Fundamentals of Programming (Chapters 1-8)
- Part II: Object-Oriented Programming (Chapters 9-16)

Fundamentals of Programming

1. Introduction to computers, programs, and C++
2. Elementary programming techniques with primitive data types, expressions, and operators
3. Selection statements
4. Mathematical functions, characters, and strings
5. Loops
6. Functions
7. Arrays
8. Explore pointers and dynamic memory management

Object-Oriented Programming

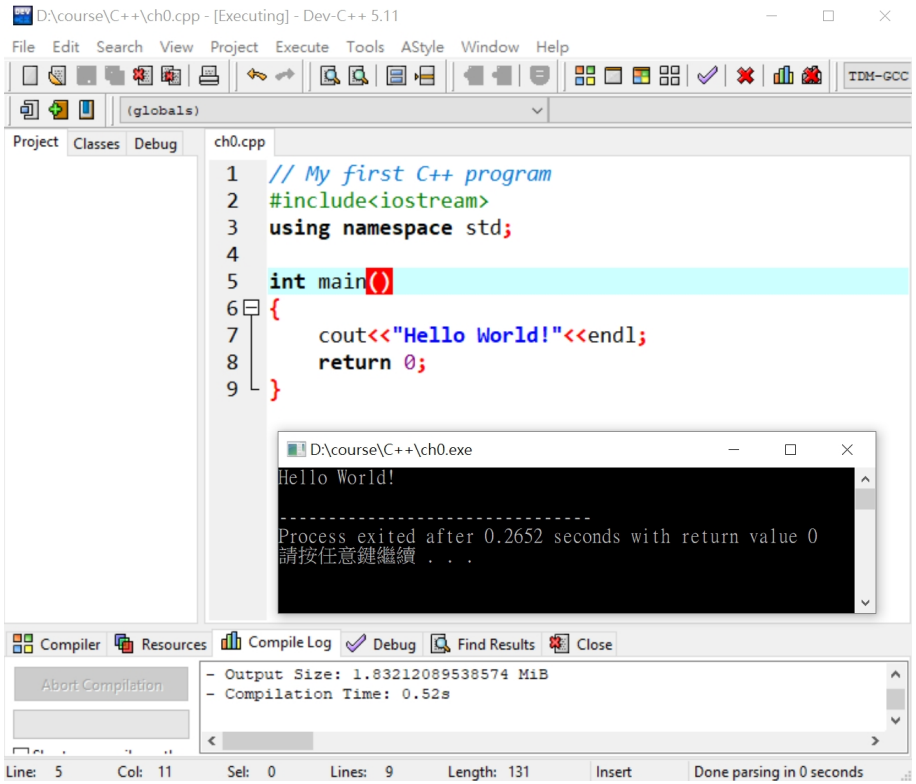
1. Programming with objects and classes
2. Design classes
3. Develop generic classes using templates
4. Use IO classes for file input and output
5. Use operators to simplify functions
6. Define classes from base classes
7. Design and implement linked lists, queues (optional)

C++ Development Tools

- You can use a C++ development tool, such as Visual C++ or Dev-C++ which we will use.

- These tools support an integrated development environment (IDE) for rapidly developing C++ programs.

- Editing, compiling, building, executing, and debugging programs are integrated in one graphical user interface.



Introduction to computers, programs, and C++

- The central theme of this course is to learn how to solve problems by writing a program.

- The term programming means to create (or develop) software, which is also called a program (code).

- In basic terms, software contains the instructions that tell a computer-or a computerized device-what to do.

- This course teaches you how to create programs by using the C++ programming language.

- There are many programming languages, some of which are decades old. Each language was invented for a specific purpose-to build on the strengths of a previous language, for example, or to give the programmer a new and unique set of tools.

- In truth, there is no "best" language. Each has its strengths and weaknesses.

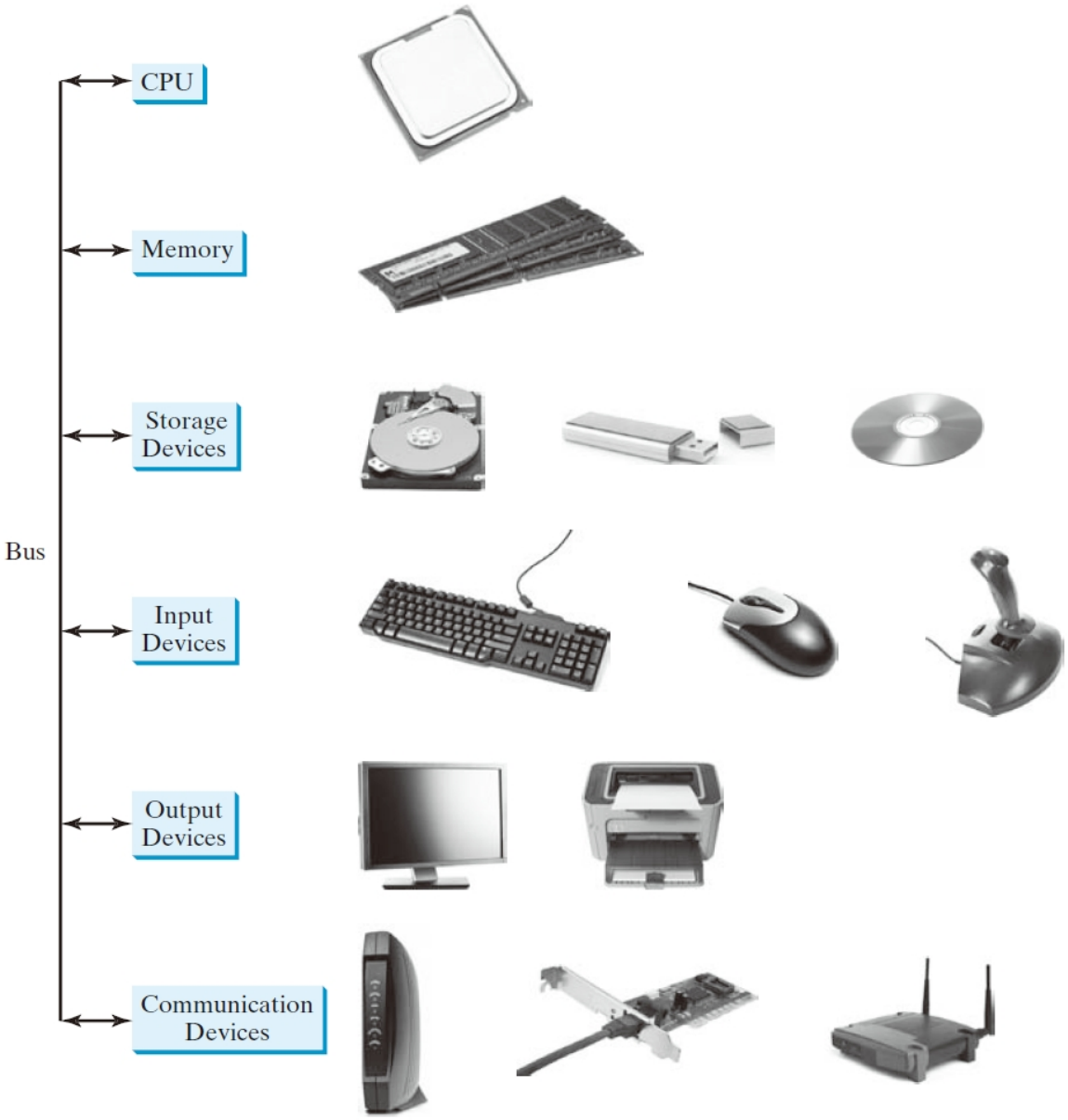
Introduction to computers, programs, and C++

- Experienced programmers know that one language might work well in some situations, and another language may be more appropriate in others.

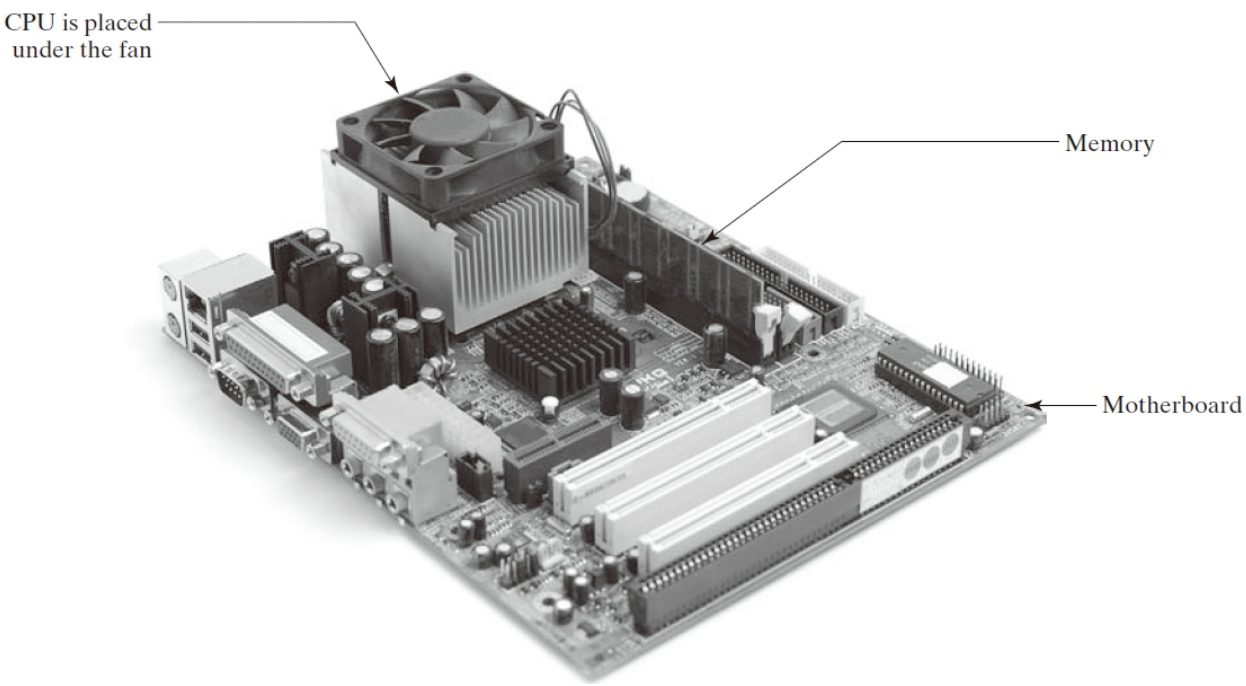
- Therefore, seasoned programmers try to master many different programming languages, giving them access to a vast arsenal of software development tools.

- If you learn to program using one language, it will be easy to pick up other languages.

What Is a Computer?

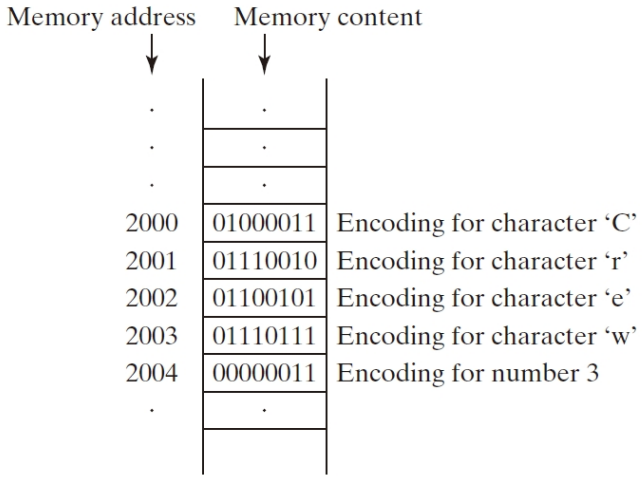


What Is a Computer?



Memory

- A computer's memory consists of an ordered sequence of bytes for storing programs as well as data that the program is working with.
- A program and its data must be moved into the computer's memory before they can be executed by the CPU.
- Every byte in the memory has a unique address.
- The address is used to locate the byte for storing and retrieving the data.



Programming Languages

- Computer programs, known as software, are instructions that tell a computer what to do.
- Computers do not understand human languages; so programs must be written in languages computers can understand.
- There are hundreds of programming languages, developed to make the programming process easier for people.
- However, all programs must be converted into the instructions the computer can execute.

Machine Language

- A computer's native language, which differs among different types of computers, is its machine language-a set of built-in primitive instructions.

- These instructions are in the form of binary code, so if you want to give a computer an instruction in its native language, you have to enter the instruction as binary code.

- For example, to add two numbers, you might have to write an instruction in binary code, as follows:

```
1101101010011010
```

Assembly Language

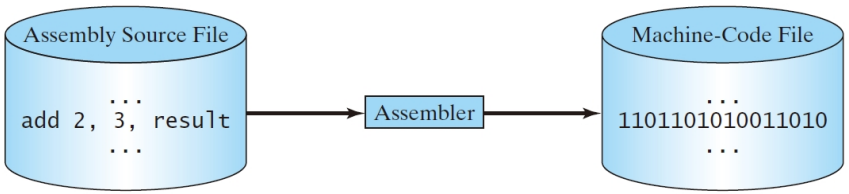
- Programming in machine language is a tedious process. Moreover, programs written in machine language are very difficult to read and modify.

- For this reason, assembly language was created in the early days of computing as an alternative to machine languages.

- Assembly language uses a short descriptive word, known as a mnemonic (silent "M"), to represent each of the machine-language instructions.

- For example, to add the numbers 2 and 3 and get the result, you might write an instruction in assembly code like this:

```
add 2, 3, result
```



High-Level Language

- In the 1950s, a new generation of programming languages known as high-level languages emerged.

- They are platform-independent, which means that you can write a program in a highlevel language and run it in different types of machines.

- High-level languages are English-like and easy to learn and use.

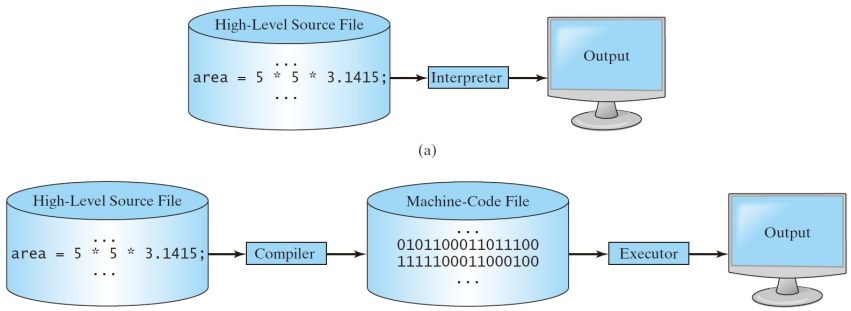
- The instructions in a high-level programming language are called statements.

- Here, for example, is a high-level language statement that computes the area of a circle with a radius of 5:

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

High-Level Language

- A program written in a high-level language is called a source program or source code.
- Because a computer cannot execute a source program, a source program must be translated into machine code for execution.
- The translation can be done using another programming tool called an interpreter or compiler.
- An interpreter reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it immediately
- A compiler translates the entire source code into a machine-code file, and the machinecode file is then executed.



Popular High-Level Programming Languages

Language	*Description*
Ada	Named for Ada Lovelace, who worked on mechanical general-purpose computers. The Ada language was developed for the Department of Defense and is used mainly in defense projects.
BASIC	Beginner's All-purpose Symbolic Instruction Code. It was designed to be learned and used easily by beginners.
C	Developed at Bell Laboratories. C combines the power of an assembly language with the ease of use and portability of a high-level language.
C++	C++ is an object-oriented language, based on C.
C#	Pronounced "C Sharp." It is a hybrid of Java and C++ and was developed by Microsoft.
COBOL	COmmon Business Oriented Language. It is used for business applications.
FORTRAN	FORmula TRANslation. It is popular for scientific and mathematical applications.
Java	Developed by Sun Microsystems, now part of Oracle. It is widely used for developing platform-independent Internet applications.
Pascal	Named for Blaise Pascal, who pioneered calculating machines in the seventeenth century. It is a simple, structured, general-purpose language primarily used for teaching programming.
Python	A simple general-purpose scripting language good for writing short programs.
Visual Basic	Visual Basic was developed by Microsoft and it enables the programmers to rapidly develop graphical user interfaces.

C family

- C++ is a general-purpose, object-oriented programming language.
- C, C++, Java, and C# are related.
- C++ evolved from C.
- Java was modeled after C++.
- C# is a subset of C++, with some features similar to Java.

History of C++

- C evolved from the B language, which evolved from the BCPL (Basic Combined Programming Language).

- Martin Richards developed BCPL 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.

- Dennis Ritchie extended the B language by adding types and other features in 1971 to develop the UNIX operating system.

- Today, C is portable and hardware independent. It is widely used for developing operating systems.

History of C++

- C++ is an extension of C, developed by Bjarne Stroustrup at Bell Laboratories during 1983-1985.

- C++ added a number of features that improved the C language. Most important, it added the support of using classes for object-oriented programming.

- Object-oriented programming can make programs easy to reuse 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.

- An international standard for C++, known as C++98, was created by the International Standard Organization (ISO) in 1998.

- A new standard, known as C++11, was approved by ISO in 2011. C++11 added new features into the core language and standard library.

Welcome.cpp

LISTING 1.1 Welcome.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6    // Display Welcome to C++ to the console
7    cout << "Welcome to C++!" << endl;
8
9    return 0;
10 }
```

```
Welcome to C++!
```

- C++ is a case-sensitive language. This means that uppercase and lowercase letters are treated as different characters.

- The line numbers are not part of the program, but are displayed for reference purposes.

  `#include <iostream>`

  is a compiler **preprocessor directive** that tells the compiler to include the **iostream library** in this program, which is needed to support console input and output.

Welcome.cpp

- C++ library contains predefined code for developing C++ programs.

- The library like **iostream** is called a **header file** in C++, because it is usually included at the head of a program.

  ```
  using namespace std;
  ```

  tells the compiler to use the **standard namespace**. **std** is an abbreviation of standard.

- **Namespace** is a mechanism to avoid naming conflicts in a large program.

- The names **cout** and **endl** in line 7 are defined in the iostream library in the standard namespace.

- For the compiler to find these names, the statement in line 2 must be used. For now, all you need to do is to write line 2 in your program for performing any input and output operations.

Welcome.cpp

- Every C++ program is executed from a **main** function.
- A function is a construct that contains statements.
- The main function defined in lines 4-10 contains two statements. They are enclosed in a block that starts with a left brace, {, (line 5) and ends with a right brace, } (line 10).

```
 4   int main()
 5   {
 6     //Display Welcome to C++ to the console
 7     cout<<"Welcome to C++"<< endl;
 8
 9    return 0;
10
11   }
```

- Every statement in C++ must end with a **semicolon** (;), known as the statement **terminator**.

Welcome.cpp

- **cout** (pronounced see-out) stands for console output.
- The **<<** (angle brackets or less than) operator, referred to as the **stream insertion operator**, sends a string to the console.
- A string must be enclosed in **quotation marks**. The statement in line 7 first outputs the string "Welcome to C++!" to the **console**, then outputs **endl**.
- Note that **endl** stands for **end line**.
- Sending endl to the console ends a line and flushes the output buffer to ensure that the output is displayed immediately.

Welcome.cpp

- The statement (line 9)

```
9    return 0;
```

  is placed at the end of every main function to exit the program.

- The value 0 indicates that the program has terminated with a successful exit.

- Some compilers will work fine if this statement is omitted; however, other compilers will not.

- It is a good practice always to include this statement for your program to work with all C++ compilers.

Welcome.cpp

- Line 6 is a comment that documents what the program is and how it is constructed.

- Comments help programmers to communicate and understand the program. They are not programming statements and thus are ignored by the compiler.

- In C++, a comment is preceded by two slashes (//) on a line, called a line comment, or

- enclosed between /* (asterisk) and */ on one or several lines, called a block comment or 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 */.

```
6     // I am a line comment.
7     /* I am a
8           block comment or
9           paragraph comment. */
```

Special Characters

Character	Name	Description
#	Pound sign	Used in #include to denote a preprocessor directive.
<>	Opening and closing angle brackets	Encloses a library name when used with #include.
( )	Opening and closing parentheses	Used with functions such as main().
{ }	Opening and closing braces	Denotes a block to enclose statements.
//	Double slashes	Precedes a comment line.
<<	Stream insertion op.	Outputs to the console.
>>	Stream extraction operator	Assigns an input to a variable.
" "	Opening and closing quotation marks	Wraps a string (i.e., sequence of characters).
;	Semicolon	Marks the end of a statement.

Perform mathematical computations

```
#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;
}
```

```
(10.5 + 2 * 3) / (45 - 3.5) = 0.39759036144578314
```

Exercise

In-Class Exercise: Write a C++ code to print out your personal information, e.g. name, gender, etc.
In-Class Exercise: Write a C++ code to compute the area of the disk with radius 3, where $\pi \sim 3.14$.

C++ Program-Development Cycle

- The C++ program-development process consists of creating/modifying source code, compiling, linking and executing programs.

- A C++ compiler contains three separate programs: preprocessor, compiler, and linker.

- For simplicity, we refer to all three programs together as a C++ compiler.

  1. A preprocessor is a program that processes a source file before it is passed down to the compiler.

     - The preprocessor processes the directives. The directives start with the # sign.

     - For example, #include in line 1 of Listing 1.1 is a directive to tell the compiler to include a library.

     - The preprocessor produces an intermediate file.

C++ Program-Development Cycle

2. The compiler then translates the intermediate file into a machine-code file.
   - The machine-code file is also known as an object file.
   - This is not the "C++ objects".
3. The linker links the machine-code file with supporting library files to form an executable file.

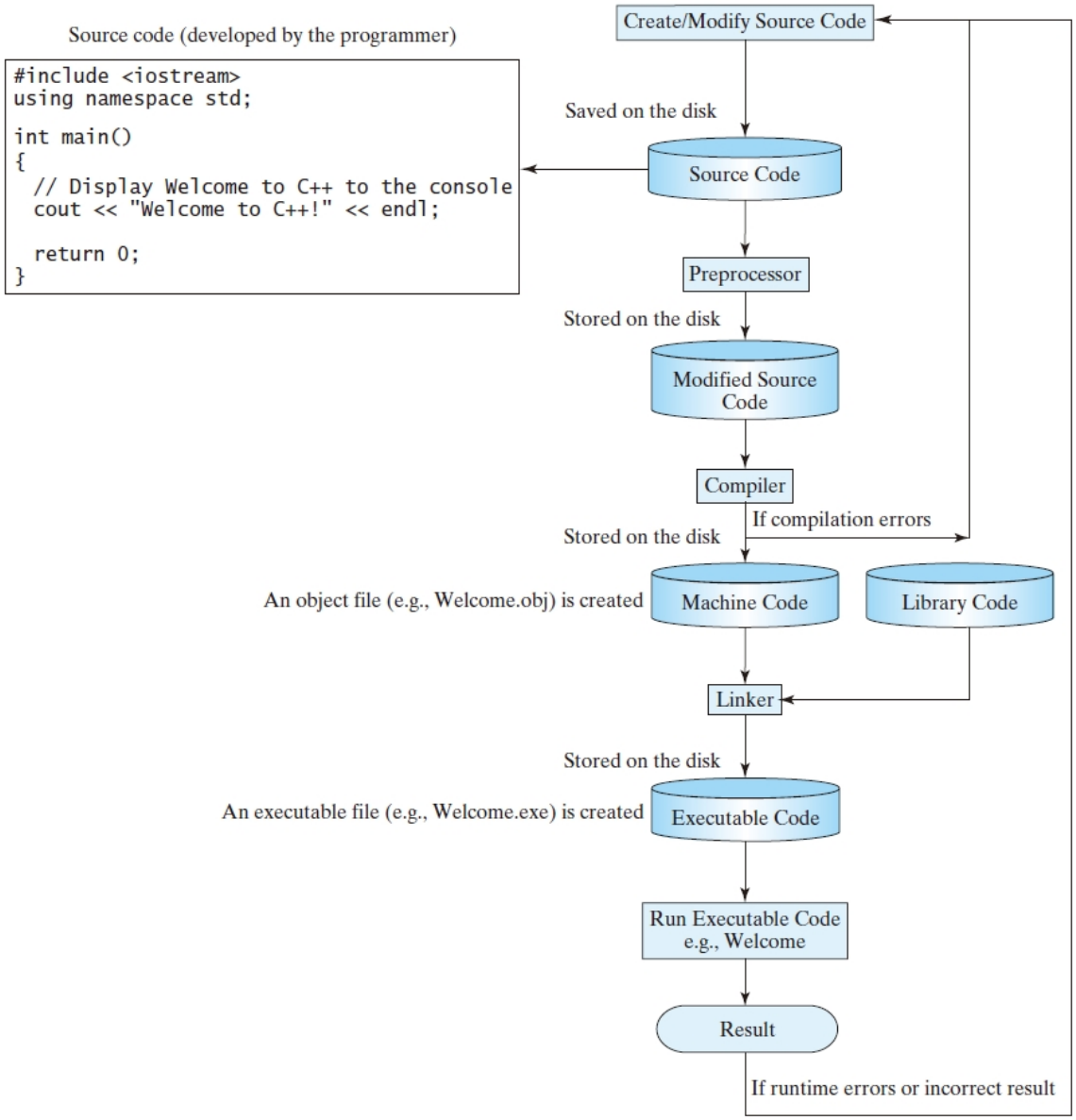
- On Windows, the machine-code file is stored on disk with an .obj extension
- The executable files are stored with an .exe extension.
- On UNIX, the machinecode file has an .o extension and the executable files do not have file extensions.
- A C++ source file typically ends with the extension .cpp.

C++ Program-Development Cycle

Preprocessing

Compiling

Linking



Programming Style and Documentation

- Good programming style and proper documentation make a program easy to read and help programmers prevent errors.

- Programming style deals with what programs look like.

- A program could compile and run properly even if you wrote it on one line only, but writing it that way would be bad programming style because it would be hard to read.

- Documentation is the body of explanatory remarks and comments pertaining to a program.

Programming Style and Documentation

- Programming style and documentation are as important as coding. Good programming style and appropriate documentation reduce the chance of errors and make programs easy to read.

- Include a summary at the beginning of the program to explain what the program does.

- A consistent indentation style makes programs clear and easy to read, debug, and maintain.

- Indentation is used to illustrate the structural relationships between a program's components or statements.

- Properly aligned code is easier to read and maintain.

Programming Errors

- Programming errors can be categorized into three types: syntax errors, runtime errors, and logic errors.

- Errors that are detected by the compiler are called syntax errors or compile errors.

- Syntax Errors example:

```
Compile

1>Test.cpp(4): error C2144: syntax error : 'int' should be preceded by ';'
1>Test.cpp(6): error C2001: newline in constant
1>Test.cpp(8): error C2143: syntax error : missing ';' before 'return'
```

- Runtime errors cause a program to terminate abnormally. They occur while an application is running if the environment detects an operation that is impossible to carry out.

- For instance, if the program expects to read in a number, but instead the user enters a string, this causes data-type errors to occur.

Programming Errors

- Logic errors occur when a program does not perform the way it was intended.
- For instance, 9/0.
- Common Error 1: Missing Braces
- Common Error 2: Missing Semicolons
- Common Error 3: Missing Quotation Marks
- Common Error 4: Misspelling Names

  Homework 1.1: Programming exercise, 1.3, 1.5, 1.7.