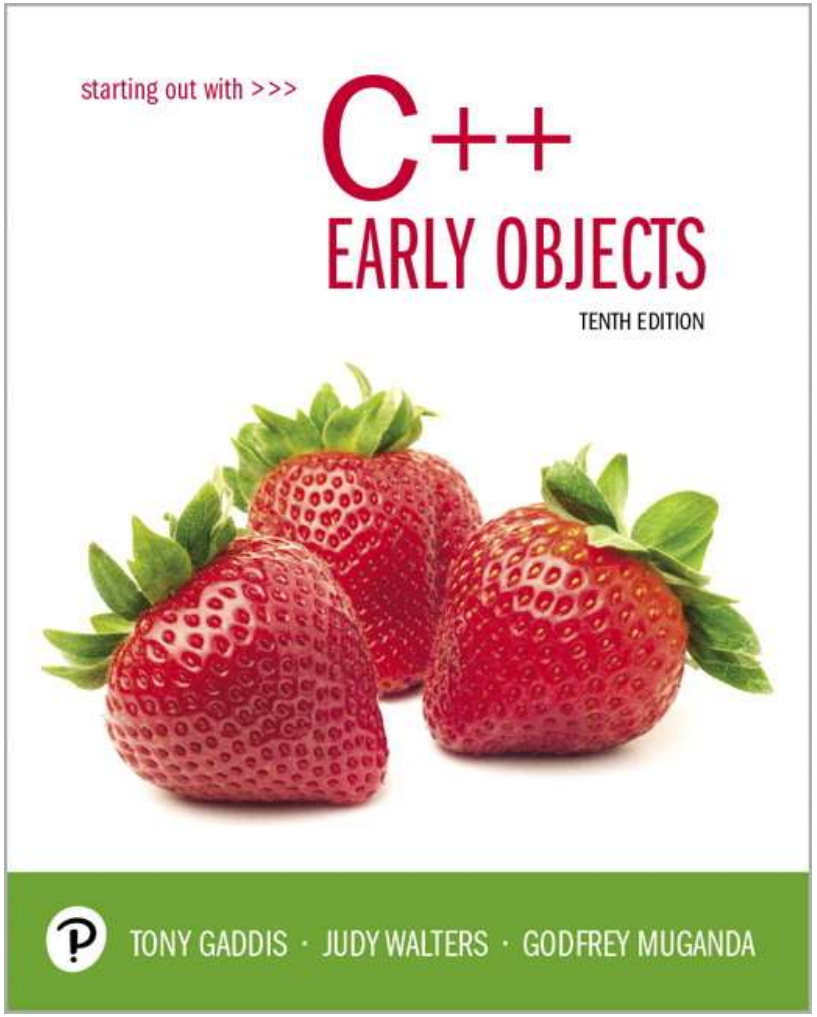
Starting Out with C++ Early Objects

Tenth Edition



Chapter 1

Introduction to Computers and Programming

Topics

1.1 Why Program?

1.2 Computer Systems: Hardware and Software

1.3 Programs and Programming Languages

1.4 What Is a Program Made of?

1.5 Input, Processing, and Output

1.6 The Programming Process

1.1 Why Program?

Computer – programmable machine designed to follow instructions

Program/Software – instructions that a computer follows to perform a task

Programmer – person with the right skills who designs, creates, and tests programs for computers

SO, without programmers, no programs; without programs, the computer cannot do anything

Programming – an Art and a Science

Artistry in programming:

- Organization of the tasks that the program performs
- How information is displayed
- How the user interacts with the program

Science in programming:

- Understanding of the language used to write the program
- Understanding how to test the program, and to change it if it does not work as intended

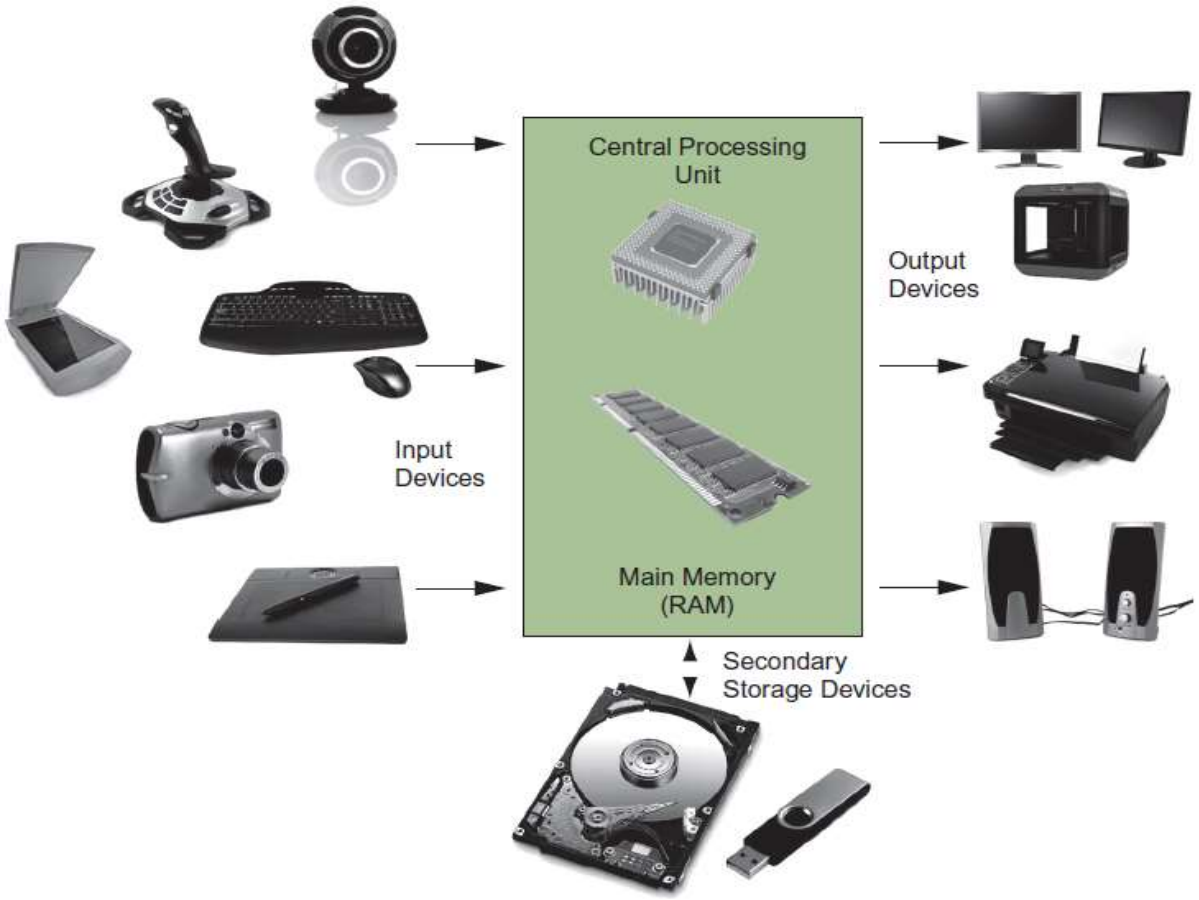
1.2 Computer Systems: Hardware and Software

Hardware – Physical components of a computer

Main Hardware Component Categories

1. Central Processing Unit (CPU)
2. Main memory (RAM)
3. Secondary storage devices
4. Input Devices
5. Output Devices

Main Hardware Component Categories

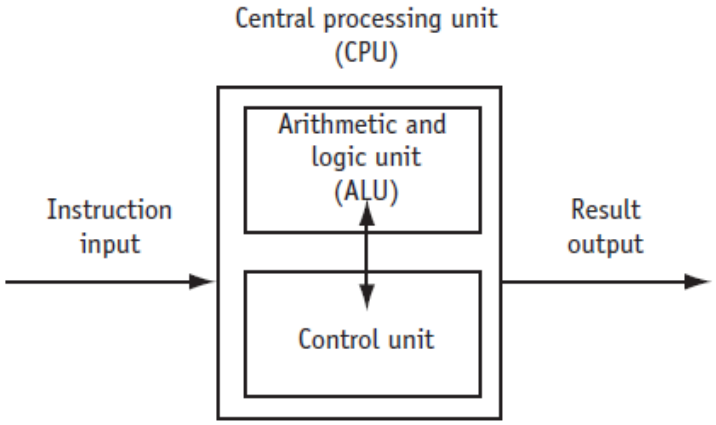


Central Processing Unit (CPU)

CPU – Hardware component that runs programs

Includes

- Control Unit
  - Retrieves and decodes program instructions
  - Coordinates computer operations
- Arithmetic & Logic Unit (ALU)
  - Performs mathematical operations



The CPU's Role in Running a Program

Cycle through:

- Fetch: get the next program instruction from main memory
- Decode: interpret the instruction and generate a signal
- Execute: route the signal to the appropriate component to perform an operation

Main Memory

- Holds both program instructions and data
- Volatile – erased when the program terminates or computer is turned off
- Also called Random Access Memory (RAM), because the CPU can access data and instructions from any memory location.

Main Memory Organization

- Bit
  - Smallest piece of memory
  - Stands for **b**inary dig**it**
  - Hold an electrical charge
    - A positive charge is "on"
    - A negative charge is "off"
- Byte
  - Is 8 consecutive bits
  - Has an address in memory
  - There are millions (or even billions) of bytes of memory in a computer

Secondary Storage

- Holds data when the program is not running or when the computer is turned off
- Several forms of secondary storage
  - disk drive: can be mounted inside the computer or connected to an external port. Data is stored magnetically or in solid-state memory.
  - flash: SD memory card, USB flash drive
  - optical: CD or DVD drive

Input Devices

- Used to send information to the computer from outside
- Many devices can provide input
  - keyboard, mouse, touch screen, microphone, scanner, digital camera, disk drive, CD/DVD drive, USB flash drive

Output Devices

- Used to send information from the computer to the outside
- Many devices can be used for output
  - Computer screen, printer, speakers, disk drive, CD/DVD recorder, USB flash drive

Software Programs That Run on a Computer

- System software
  - programs that manage the computer hardware and the programs that run on the computer
  - Operating Systems
    - Controls the operation of the computer
    - Manages connected devices and access to storage devices
    - Allows programs to run
  - Utility Programs
    - Support programs that enhance computer operations
    - Examples: virus scanners, data backup, file compression
  - Software development tools
    - Used by programmers to create software
    - Examples: compilers, integrated development environments (IDEs)

1.3 Programs and Programming Languages

- Program
  a set of instructions directing a computer to perform a task
- Programming Language
  a special language used to write programs

Algorithm

Algorithm: a set of well-defined steps to perform a task or to solve a problem

Order is important. Steps must be performed sequentially

Programs and Programming Languages

Types of languages

- Low-level: used for communication with computer hardware directly. Not very easy for a person to read.
- High-level: closer to human language

From a High-level Program to an Executable File: 1of 3

a) Create a file containing the program with a text editor.

- program statements: source code
- file: source file

b) Run the preprocessor to convert source file directives to source code program statements.

c) Run the compiler to convert source program statements into machine instructions (machine code), which is stored in an object file.

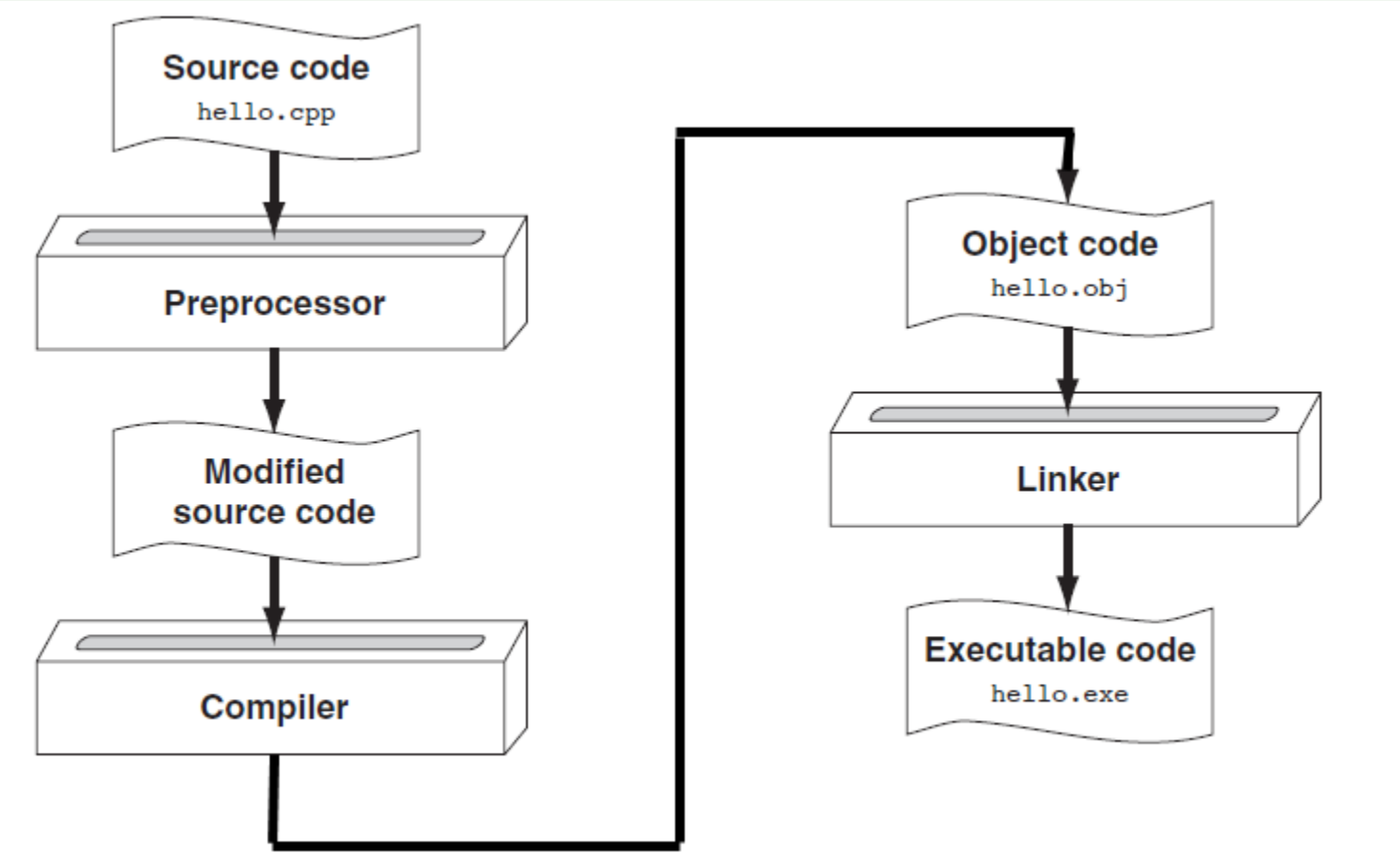
From a High-level Program to an Executable File 2 of 3

d) Run the linker to connect hardware-specific library code to machine instructions, producing an executable file.

Steps b) through d) are often performed by a single command or button click.

Errors that occur at any step will prevent the execution of the following steps.

From a High-level Program to an Executable File 3 of 3



Putting It All Together

There are many software development systems that provide an integrated development environment (IDE). An IDE has the following:

- text editor
- compiler
- debugger
- other supporting utilities
- a user interface that ties them all together

1.4 What Is a Program Made Of?

There are common elements in most programming languages

1) Language elements:
   - Key Words
   - Programmer-Defined Identifiers
   - Operators
   - Punctuation
   - Syntax

Example Program

```
#include <iostream>
using namespace std;

int main()
{
  double num1 = 5,
         num2, sum;
   num2 = 12;

   sum = num1 + num2;
   cout << "The sum is " << sum;
   return 0;
}
```

Key Words

- Also known as reserved words
- Have a special meaning in C++
- Can not be used for another purpose
- Written using lowercase letters
- Examples in program (shown in green):

```
using namespace std;
int main()
```

Programmer-Defined Identifiers

- Names made up by the programmer
- Not part of the C++ language
- Used to represent various things, such as variables (memory locations)
- Example in program (shown in green):

```
double num1
```

Operators

- Used to perform operations on data
- Many types of operators
  - Arithmetic: `+, -, *, /`
  - Assignment: `=`
- Examples in program (shown in green):

```
num2 = 12;
sum = num1 + num2;
```

Punctuation

- Characters that mark the end of a statement, or that separate items in a list
- Example in program (shown in green):

```
double num1 = 5,
       num2, sum;
num2 = 12;
```

Lines vs. Statements 1 of 2

In a source file,

A line is all of the characters entered before a carriage return.

Blank lines improve the readability of a program.

Here are four sample lines. Line 3 is blank:

```
1. double num1 = 5, num2, sum;
2. num2 = 12;
3.
4. sum = num1 + num2;
```

Lines vs. Statements 2 of 2

In a source file,

A statement is an instruction to the computer to perform an action.

A statement may contain keywords, operators, programmer-defined identifiers, and punctuation.

A statement may fit on one line, or it may occupy multiple lines.

Here is a single statement that uses two lines:

```
double num1 = 5,
       num2, sum;
```

Variables

- A variable is a named location in computer memory.
- It holds a piece of data. The data that it holds may change while the program is running.
- The name of the variable should reflect its purpose
- A variable must be *defined* before it can be used. Variable definitions indicate the variable name and the type of data that it can hold.
- Example variable definition:

```
double num1;
```

1.5 Input, Processing, and Output

Three steps that many programs perform

1) Gather input data
   - from keyboard or mouse
   - from files on disk drives

2) Process the input data

1) Output the results of the processing
   - send it to the screen or a printer
   - write it to a file

1.6 The Programming Process 1 of 2

1. Define what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools to create a model of the program.

   Hierarchy charts, flowcharts, pseudocode, etc.
4. Check the model for logical errors.
5. Write the program source code.
6. Compile the source code.

The Programming Process 2 of 2

7. Correct any errors found during compilation.
8. Link the program to create an executable file.
9. Run the program using test data for input.
10. Correct any errors found while running the program.

Repeat steps 4 - 10 as many times as necessary.

11. Validate the results of the program.
    Does the program do what was defined in step 1?

Copyright