C++ Programming:
Basic Elements of C++

Some material taken from: C++ Programming: Program Design Including Data Structures

Objectives

- Become familiar with the basic components of a C++ program, including functions, special symbols, and identifiers
- Explore simple data types
- Discover how to use arithmetic operators and expressions
- Learn what an assignment statement is

Objectives

- Discover how to input data into memory
- Examine ways to display output results
- Understand preprocessor directives
- Learn how to structure a program and use comments

Introduction

- Computer program: sequence of statements designed to accomplish some task
- Programming: planning/creating a program
- Syntax: rules that specify which statements are legal
- Programming language: a set of rules, symbols, and special words

C++ Programs

- A C++ program is a collection of one or more subprograms, called functions
- A subprogram or a function is a sequence of statements that, when activated (executed), accomplishes something
- Every C++ program has a function called main
- The smallest individual unit of a program written in any language is called a **token**

Tokens

- Special symbols
- Keyword symbols
- Identifiers

Special Symbols:

Include:

- +
- -
- *
- /
- .
- ;
- ?
- ,
- <=
- !=
- ==
- >=

Keyword Symbols

- Reserved words, or keywords
- Include:
  - int
  - double
  - char
  - void
  - const
  - return

Identifiers

- Used to name user-defined objects
- Consist of letters, digits, and the underscore character (_)
- Must begin with a letter or underscore
- C++ is case sensitive
- Some identifiers are defined in user libraries, example: cout and cin

Legal and Illegal Identifiers

- The following are legal identifiers in C++:
  - first
  - sum
  - secondNumber

Table 2-1 Examples of Illegal Identifiers

Illegal Identifier	Description
`employee Salary`	There can be no space between `employee` and `Salary`.
`Hello!`	The exclamation mark cannot be used in an identifier.
`one+two`	The symbol + cannot be used in an identifier.
`2nd`	An identifier cannot begin with a digit.

Data Types

- Data Type: set of values together with a set of operations is called a data type
- C++ data can be classified into three categories:
  - Simple data type
  - Structured data type
  - Pointers (not covered in this class)

Simple Data Types

- Three categories of simple data
  - Integral: integers (numbers without a decimal)
  - Floating-point: decimal numbers

Integral Data Types

Table 2-2 Values and Memory Allocation for Three Simple Data Types

Data Type	Values	Storage (in bytes)
`int`	-2147483648 to 2147483647	4
`bool`	`true` and `false`	1
`char`	-128 to 127	1

int Data Type

- Examples:
  - -6728
  - 0
  - 78
- Positive integers do not have to have a + sign in front of them
- No commas are used within an integer

bool Data Type

- bool type
  - Has two values, true and false
  - Manipulate logical (Boolean) expressions
- true and false are called logical values
- bool, true, and false are reserved words

char Data Type

- The smallest integral data type
- Used for characters: letters, digits, and special symbols
- Each character is enclosed in single quotes
- Some of the values belonging to char data type are: 'A', 'a', '0', '*', '+', '$', '&'
- A blank space is a character and is written ' ', with a space left between the single quotes

Floating-Point Data Types

- C++ uses scientific notation to represent real numbers (floating-point notation)
- double: represents any real number
  - Range: -1.7E+308 to 1.7E+308
  - Takes 8 bytes of memory

Table 2-3 Examples of Real Numbers Printed in C++ Floating-Point Notation

Real Number	C++ Floating Point-Notation
75.924	7.592400E1
0.18	1.800000E-1
0.0000453	4.530000E-5
-1.482	-1.482000E0
7800.0	7.800000E3

Arithmetic Operators

- C++ Operators
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - % remainder (mod operator), specific for integral types
- Unary operator - has only one operand
- Binary Operator - has two operands
- **Examples:**

Arithmetic Operators Examples

Result

- 2+5
- 5.0/2
- -3*2
- 21%6
- 6%7
- -7%6

Order of Precedence

- All operations inside of ( ) are evaluated first
- *, /, and % are at the same level of precedence and are evaluated next
- + and – have the same level of precedence and are evaluated last
- When operators are on the same level
  - Performed from left to right
- Example: 2*3+7.0/4*3 means (2*3)+((7.0/4)*3)
- **You are encouraged to use parenthesis**

Expressions

- If all operands are integers
  - Expression is called an integral expression
- If all operands are floating-point
  - Expression is called a floating-point expression
- An integral expression yields integral result
- A floating-point expression yields a floating-point result

Mixed Expressions

- Mixed expression:
  - Has operands of different data types
  - Contains integers and floating-point
- Examples of mixed expressions:
  - 2 + 3.5
  - 6 / 4 + 3.9
  - 5.4 * 2 – 13.6 + 18 / 2

Evaluating Mixed Expressions

- If operator has same types of operands
  - Evaluated according to the type of the operands
  - Example: (12+1)/5 evaluates to 2 !
- If operator has both types of operands
  - Integer is promoted to floating-point
  - Operator is evaluated
  - Result is floating-point
  - Example: (12+1.0)/5 evaluates 2.6

Memory Allocation

- Memory reservation
- Two types:
  - Constant: stored in a memory location, its content can't change
  - Variable: value stored in a memory location whose content may change during execution
- The syntax to declare a constant is:

```
const dataType identifier = value;
```

- In C++, const is a reserved word

Memory Allocation (continued)

- **Examples:**

```
const int  n = 37;

const double x = 2.5;
```

- We must initialize a constant

Allocating Memory (continued)

- The syntax to declare one variable:

  dataType identifier;

- The syntax to declare more than one variable of the same type:

  dataType identifier1, indentifier2;

  or equivalently

  dataType identifier1;

  dataType identifier2;

- **Examples:**

Allocating Memory (continued)

- **Examples:**

```
int x;

double y, z;

int num = 1;
```

- We can initialize a variable at declaration

Assignment Statement

- The assignment statement takes the form:

  variable = expression;

- Expression is evaluated and its value is assigned to the variable on the left side
- In C++ = is called the assignment operator
- A C++ statement such as:

  x = x + 2;

  evaluates whatever is in x, adds two to it, and assigns the new value to the memory location x
- **Examples:**

Assignment Statement Examples

value

```
const int y = 2;
int x=3;
double v;
int z = x+ y;
x = 1;
//  y = 3;  error!
v = x/y;
v = (x+0.0)/y;
v = v*y+x;
```

Input

- Data must be loaded into main memory before it can be manipulated
- Storing data in memory is a two-step process:
  1. Instruct the computer to allocate memory
  2. Include statements to put data into allocated memory

Input (Read) Statement

- cin is used with >> to gather input

  cin>>variable>>variable. . .;

- The extraction operator is >>
- For example, if miles is a double variable

  cin>>miles;

  - Causes computer to get a value of type double
  - Places it in the memory cell miles

Input Statement (continued)

- Using more than one variable in cin allows more than one value to be read at a time
- For example, if feet and inch are variables of the type int a statement such as:

```
cin>>feet>>inch;
```

  - Inputs two integers from the keyboard
  - Places them in locations feet and inch respectively

Output

- The syntax of cout and << is:

  cout<< expression or manipulator

  << expression or manipulator << ...;
- Called an output (cout) statement
- The << operator is called the insertion operator or the stream insertion operator
- Expression evaluated and its value is printed at the current cursor position on the screen

Output (continued)

- Manipulator: alters output
- endl: the simplest manipulator
  - Causes cursor to move to beginning of the next line

Output Example

- Output of the C++ statement cout<<a; is meaningful if a has a value
  - For example, the sequence of C++ statements,

```
int a = 45;

cout<<a;
```

produces an output of 45

The New Line Character

- The new line character is ‘\n’
- Without this character the output is printed on one line
- Tells the output to go to the next line
- When \n is encountered in a string
  - Cursor is positioned at the beginning of next line
- A \n may appear anywhere in the string

Examples

- Without the new line character:

  cout<<"Hello there.";

  cout<<"My name is Goofy.";

  - Would output:

    Hello there.My name is Goofy.

- With the new line character:

  cout<<"Hello there.\n";

  cout<<"My name is Goofy.";

  - Would output

    Hello there.

    My name is Goofy.

Escape Sequences

Table 2-4 Commonly Used Escape Sequences

	Escape Sequence	Description
\n	Newline	Cursor moves to the beginning of the next line
\t	Tab	Cursor moves to the next tab stop
\ b	Backspace	Cursor moves one space to the left
\r	Return	Cursor moves to the beginning of the current line (not the next line)
\\	Backslash	Backslash is printed
\'	Single quotation	Single quotation mark is printed
\"	Double quotation	Double quotation mark is printed

- Examples:

Examples

```
int a, b;
a = 65;
b = 78;
cout << 29+2<<endl;
cout<<“Hi\n”;
cout<<  “a = ” << a << “ and\n  b=  ” << b;
cout<< “\n \t  \“ :) \” ”;
```

Program flow

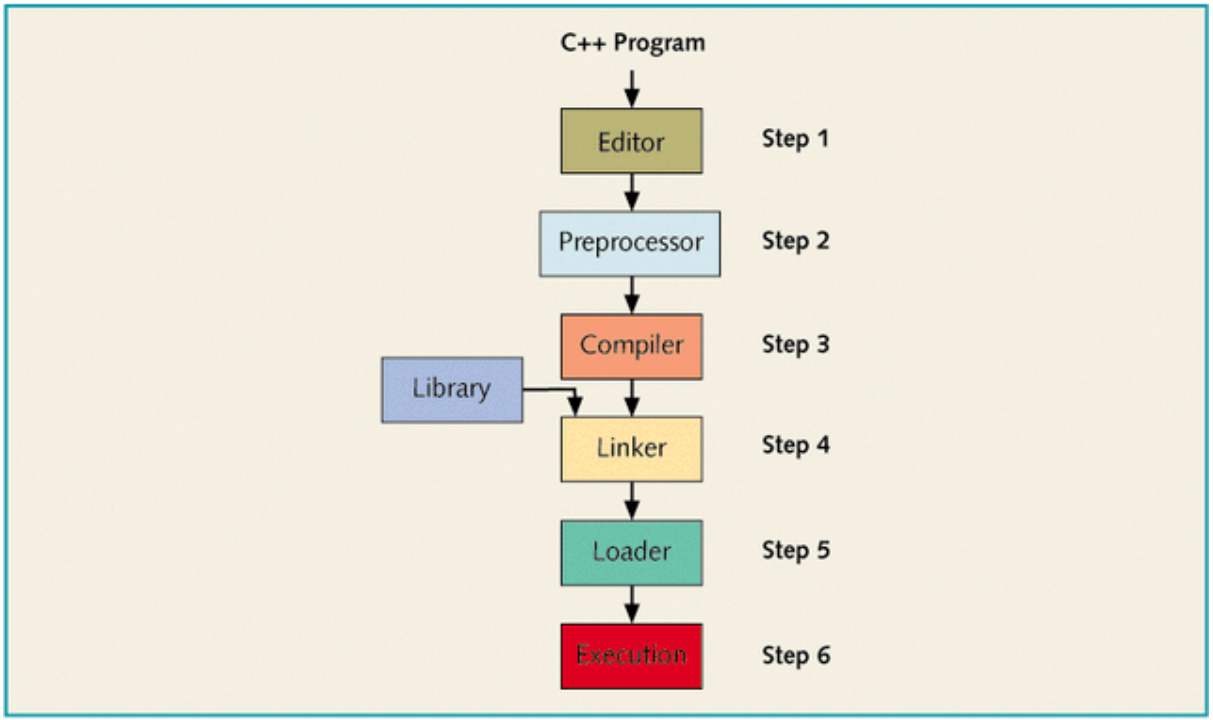

Figure 2-18 Processing a C++ program

Libraries and Preprocessor Directives

- C++ has a small number of operations
- Many functions and symbols needed to run a C++ program are provided as collection of libraries
- Every library has a name and is referred to by a header file
- Preprocessor directives are commands supplied to the preprocessor
- All preprocessor commands begin with #
- No semicolon at the end of these commands

Preprocessor Directive Syntax

- Syntax to include a header file

```
#include <headerFileName>
```

- Causes the preprocessor to include the header file iostream in the program
- The syntax is:

```
#include <iostream>
```

Header Files

- The descriptions of the functions needed to perform I/O are contained in iostream
- The syntax is:
  - #include <iostream>

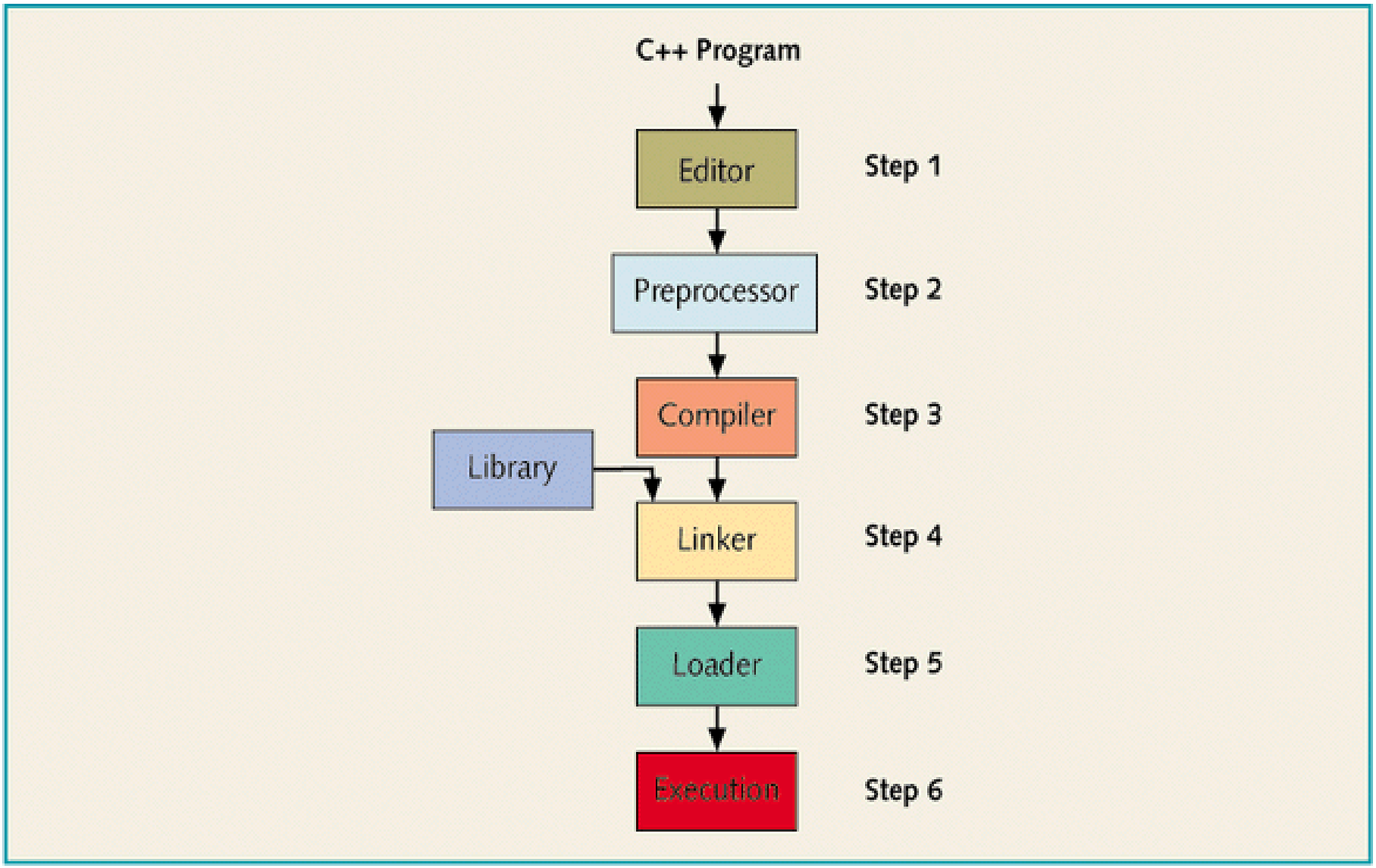

Figure 2-18 Processing a C++ program

Using cin and cout in a Program and namespace

- cin and cout are declared in the header file iostream, but within a namespace named std
- To use cin and cout in a program, use the following two statements:

```
#include <iostream>

using namespace std;
```

Other libraries

- Example: the string type, you need to access its definition from the header file string

- Include the following preprocessor directive:

```
#include <iostream>
using namespace std;
#include  <string>
//….

string str = “hello”;
cout<< str;
```

Creating a C++ Program

- C++ program has two parts:
  1. Preprocessor directives
  2. The program
- Preprocessor directives and program statements constitute C++ source code
- Source code must be saved in a file with the file extension .cpp

Creating a C++ Program (continued)

- Compiler generates the object code
  - Saved in a file with file extension .obj
- Executable code is produced and saved in a file with the file extension .exe.

Program Style and Form

- The Program Part
  - Every C++ program has a function main
  - Basic parts of function main are:
    - The heading
    - The body of the function
- The heading part has the following form

  typeOfFunction main(argument list)

Body and Syntax

- The body of the function is enclosed between the braces { and }
- Has two types of statements
  - Declaration statements
  - Executable statements
- Errors in syntax are found in compilation

```
int x;              //Line 1
int y               //Line 2: syntax error
double z;           //Line 3
y = w + x;          //Line 4: syntax error
```

Syntax

- Declaration Statements

```
int  a, b, c;
double  x, y;
```

  - Variables can be declared anywhere in the program, but they must be declared before they can be used
- Executable Statements have three forms:

```
a = 4;                                  //assignment statement
cin>>b;                                 //input statement
cout<<a<<endl<<b<<endl; //output statement
```

Use of Blanks

- Use of Blanks
  - One or more blanks separate input numbers
  - Blanks are also used to separate reserved words and identifiers from each other and other symbols
- Blanks between identifiers in the second statement are meaningless:
  - int a,b,c;
  - int a, b, c;
- In the statement: inta,b,c;

  no blank between the t and a changes the reserved word int and the identifier a into a new identifier, inta.

Semicolons, Brackets, & Commas

- Commas separate items in a list
- C++ statements end with a semicolon
- { and } are not C++ statements

Form and Style

- Consider two ways of declaring variables:
  - Method 1

    ```
    int feet, inch;

    double x, y;
    ```

  - Method 2

    ```
    int a,b;double x,y;
    ```

- Both are correct, however, the second is hard to read

Documentation

- Comments can be used to document code
  - Single line comments begin with // anywhere in the line
  - Multiple line comments are enclosed between /* and */
- Name identifiers with meaningful names
- Handle Run-together-words using CAPS for the beginning of each new word or an underscore before the new word. Examples:
  - int numOfStudents = 30;
  - int num_of_courses = 5;

Body of the Function

- The body of the function main has the following form:

```
int main ()
{
declare variables
statements
return 0;
}
```

Writing a Complete Program

- Begin the program with comments for documentation
- Include header files
- Declare named constants, if any
- Write the definition of the function main

Complete Examples

- Run Example 1
- Run Example 2
- Run Example 3

Summary

- C++ program: collection of functions where each program has a function called main
- Identifier consists of letters, digits, and underscores, and begins with letter or underscore
- The arithmetic operators in C++ are addition (+), subtraction (-),multiplication (*), division (/), and modulus (%)
- Arithmetic expressions are evaluated using the precedence associativity rules

Summary

- All operands in an integral expression are integers and all operands in a floating-point expression are decimal numbers
- Mixed expression: contains both integers and decimal numbers
- A named constant is initialized when declared
- All variables must be declared before used

Summary

- Use cin and stream extraction operator >> to input from the standard input device
- Use cout and stream insertion operator << to output to the standard output device
- Preprocessor commands are processed before the program goes through the compiler
- A file containing a C++ program usually ends with the extension .cpp