Introduction to C++ Programming, Input/Output, and Operators

Chapter 2

TOPICS

Parts of a C++ Program

Input and Output in C++

Arithmetic and Arithmetic Operators

Using declarations and using directives

PARTS OF A C++ PROGRAM

First Program in C++: Printing a Line of Text

Simple program that prints a line of text

Fig02_01.cpp

Parts of a C++ Program – Comments, Directives, White Space

Comments

- // – the rest of the line is a comment
- /* and */ – anything in between is a comment (can span multiple lines)

Preprocessor directive

- processed by the preprocessor before compilation
- #include <iostream> What do you think this library is for?

White space

- blank lines, spaces and indenting make code easier to read

PARTS OF A C++ PROGRAM – MAIN

main

- Function, and a part of every C++ program
- Where the program begins executing

C++ programs typically have one or more functions and classes

- One function in every program must be named main

What does the **int** to the left of main refer to?

Parts of a C++ Program – Braces, Statements

Matching braces

- Surround body of every function
- Surround other code blocks

Statements

- instruct computer to perform an action.
- Most C++ statements end with a semicolon

INPUT AND OUTPUT IN C++

INPUT / OUTPUT IN C++ – CIN, COUT

Input & output accomplished with **streams of characters**

cin

- Pronounced “see-in”
- **Standard input stream** – keyboard

cout

- Pronounced “see-out”
- **Standard output stream** – screen

INPUT / OUTPUT IN C++ – CERR

Data may be output to other devices, such as files, disks and printers

cerr

- Pronounced “see-air”
- **Standard error stream**
- For displaying error messages.

OUTPUT STATEMENTS – SYNTAX

cout sends stream of characters to standard output stream object (usually connected to the screen)

std::cout << “Welcome to C++!”;

Output Statements – Parts of the Statement

std::cout << “Welcome to C++!”;

std is a namespace

- A grouping of various entities like classes, objects and functions
- std is the namespace referred to by <iostream>

:: is an operator

- a way to reference a function from a namespace

cout

- The particular function we want to use from this namespace

OUTPUT STATEMENTS

```
std::cout << "Welcome to C++!";
```

<< is the **stream insertion operator**

- It means "Put this thing that follows me into the stream"
- It points in the direction where data flows

"Welcome to C++!"

- The text to send through the stream
- Will display on the screen

Another C++ Program: Adding Integers

Gets two integers entered by user, computes the sum of these values and outputs the result using std::cout

Fig02_05.cpp

Another C++ Program: Adding Integers – Note the Parts

Declarations

- introduce identifiers into programs
- For variables, functions, etc.

Variable

- Name for a memory location where a value can be stored
- Must be declared with a data type

int, **double** and **char** are **fundamental types**

- Aka primitive types
- Fundamental-type names are keywords and all lowercase

C++ is case sensitive!

INPUT STATEMENTS

cin takes input from the standard input stream object (usually connected to the keyboard)

Prompts

- Direct the user to take a specific action
- Meaningful prompts → user-friendliness, fewer input errors

INPUT STATEMENTS – SYNTAX

std::cin >> number1;

std is a namespace

- A grouping of various entities like classes, objects and functions
- std is the namespace referred to by <iostream>

:: is an operator

- a way to reference a function from a namespace

cin

- The particular function we want to use from this namespace

INPUT STATEMENTS – SYNTAX, CONTINUED

```
std::cin >> number1;
```

>> is the **stream extraction operator**

- It means “Read the stream and put its contents in this variable” OR,
- “cin gives number1”
- It points in the direction where data flows

number1

- Where to store the value read from the stream

More on Displaying Output

On some systems, outputs accumulate until there are enough to "make it worthwhile" to display them on the screen

- Called **buffering**
- So we may need to "**flush**" the stream

std::endl is a **stream manipulator**

- **endl** means "end line" and belongs to namespace **std**
- Outputs a **newline**, then "flushes the output buffer"
- Forces any accumulated outputs to be displayed immediately
- Can be important when outputs are prompting user for action, like entering data

MULTIPLE STREAM INSERTION OPERATORS

Called **concatenating**, **chaining** or **cascading** stream insertion operations

Calculations can also be performed in output statements

```
std::cout << “Sum is “ << num1 + num2 << std::endl;
```

Arithmetic and Arithmetic Operators

ASSIGNMENT STATEMENT EXAMPLE

sum = number1 + number2;

Assignment statement that calculates the sum of the variables **number1** and **number2** then assigns the result to variable **sum** by using assignment operator **=**

- "sum gets the value of number1 + number2."

Operator = is a **binary** operator—it has two operands.

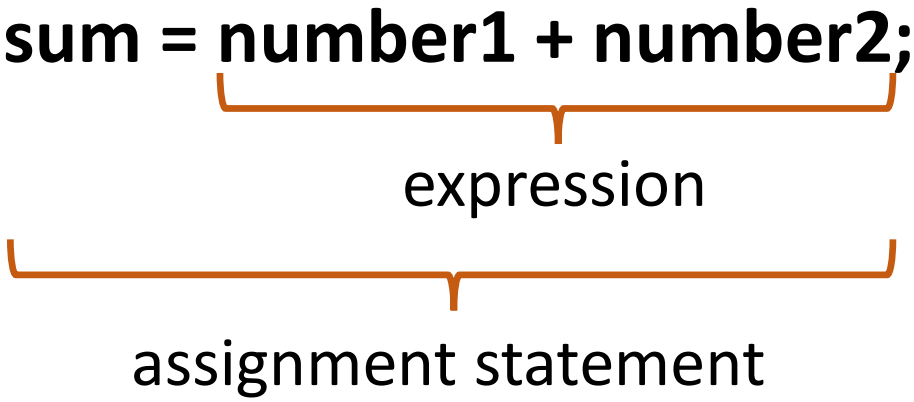
Everything to right of assignment operator is always evaluated before the assignment is performed

ASSIGNMENT STATEMENT – PARTS

In general, calculations are performed in assignment statements

Portions of statements that contain calculations are called **expressions**

An expression has a **value** associated with it



VARIABLES

Remember that every variable has:

- name
- type
- size (in bytes)
- value
- location (or address)

When a new value is placed into a variable, the new value replaces the previous value (if any)

The previous value is lost.

Arithmetic Operators

Addition – plus sign **+**

Subtraction – minus sign **-**

Multiplication – asterisk ***

Division – forward slash **/**

Remainder operator – percent sign **%**

These are **binary operators** because they each operate on **two operands**

INTEGER DIVISION

Integer division ALWAYS yields an integer quotient.

Any fractional part in integer division is discarded (truncated)

No rounding occurs.

Example:

17 / 5

evaluates to **3**

REMAINDER OPERATOR

The remainder operator yields the remainder after division

Example:

17 % 5

evaluates to **2**

PARENTHESES

Parentheses may be used to group terms in expressions in the same manner as in algebraic expressions.

If an expression contains nested parentheses, the expression in the innermost set of parentheses is evaluated first.

Precedence and Associativity of Operators

Precedence	Operator	Name	Evaluated
Highest	( )	P	Inner to outer
	* / %	MD (and R)	Left to right
	+ —	AS	Left to right
	< <= > >=	relational	Left to right
	== !=	equality	Left to right
Lowest	=	assignment	Right to left

Note that C++ doesn't have an exponent operator, so there's no E

EQUALITY AND RELATIONAL OPERATORS

fig02_13.cpp (Note some slight differences)

Condition – an expression that can be true or false

if selection statement

- Allows a program to make a decision based on a condition's value

USING **DECLARATIONS AND** USING **DIRECTIVES**

USING DECLARATIONS

Eliminate the need to repeat the **std::** prefix

A **using declaration** at the top of the file, like this:

```
using std::cout;
```

Replaces the **std::** in a statement like this:

```
std::cout << "Welcome to C++!\n";
```

Making the code a little less cluttered, like this:

```
cout << "Welcome to C++!\n";
```

USING DIRECTIVES

Can also eliminate a series of **using declarations**:

```
using std::cout;
using std::cin;
using std::cerr;
```

With a single **using directive**:

```
using namespace std;
```

IF STATEMENTS

Keyword **if** followed by a condition in parentheses

No curly braces needed if only one statement in body

Curly braces needed if more than one statement in body

Indenting body not required, but improves readability

Note that there is no semicolon **;** at the end of the first line of an if statement

- Such a semicolon would compile but result in a logic error at execution time.
- Treated as the empty statement—semicolon by itself