Programming Fundamentals

Introduction to C++

Dr. Ramez Alkhatib

Topics

2.1 The Parts of a C++ Program

2.2 The cout Object

2.3 The **#include** Directive

2.4 Standard and Prestandard C++

2.5 Variables, Constants, and the Assignment Statement

2.6 Identifiers

2.7 Integer Data Types

2.8 The **char** Data Type

Topics Cont'd

2.9   The C++ **string** Class

2.10 Floating-Point Data Types

2.11 The **bool** Data Type

2.12 Determining the Size of a Data Type

2.13 More on Variable Assignments and Initialization

2.14 Scope

2.15 Arithmetic Operators

2.16 Comments

2.1 The Parts of a C++ Program

```
// sample C++ program          ←——————— comment
#include <iostream>            ←————— preprocessor directive
using namespace std;           ←————— which namespace to use
int main()      ←—————— beginning of function named main
{   ←———————————————— beginning of block for main
    cout << "Hello, there!";    ←— output statement
    return 0;←——————— send 0 back to operating system
}   ←———————————————— end of block for main
```

2.1 The Parts of a C++ Program Cont'd

Statement	Purpose
`// sample C++ program`	comment
`#include <iostream>`	preprocessor directive
`using namespace std;`	which namespace to use
`int main()`	beginning of function named `main`
`{`	beginning of block for `main`
`cout << "Hello, there!";`	output statement
`return 0;`	send 0 back to the operating system
`}`	end of block for `main`

Special Characters

Character	Name	Description
//	Double Slash	Begins a comment
#	Pound Sign	Begins preprocessor directive
< >	Open, Close Brackets	Encloses filename used in `#include` directive
( )	Open, Close Parentheses	Used when naming function
{ }	Open, Close Braces	Encloses a group of statements
" "	Open, Close Quote Marks	Encloses string of characters
;	Semicolon	Ends a programming statement

Important Details

- C++ is <u>case-sensitive</u>.
  - Uppercase and lowercase characters are different characters. '`Main`' is not the same as '`main`'.
- Every { must have a corresponding }, and vice-versa.

2.2 The *cout* Object

- Displays information on computer screen
- Use **<<** to send information to cout

```
cout << "Hello, there!";
```

- Can use **<<** to send multiple items to cout

```
cout << "Hello, " << "there!";
```
 Or
```
cout << "Hello, ";
cout << "there!";
```

Starting a New Line

- To get multiple lines of output on screen

  - Use **endl**

    ```
    cout << "Hello, there!" << endl;
    ```

  - Use **\n** in an output string

    ```
    cout << "Hello, there!\n";
    ```

2.3 The `#include` Directive

- Inserts the contents of another file into the program

- Is a preprocessor directive
  - Not part of the C++ language
  - Not seen by compiler

- Example:

  ```
  #include <iostream>
  ```

No ; goes here

2.4 Standard and Prestandard C++

Older-style C++ programs

- Use `.h` at end of header files

    `#include <iostream.h>`

- Do not use `using namespace` convention

- May not compile with a standard C++ compiler

- Variable

  - Has a name and a type of data it can hold

  data type → `char letter;` ← variable name

  - Is used to reference a location in memory where a value can be stored

  - Must be defined before it can be used

  - The value that is stored can be changed, *i.e.,* it can "vary"

Variables

- If a new value is stored in the variable, it replaces the previous value

- The previous value is overwritten and can no longer be retrieved

```cpp
int age;
age = 17;      // age is 17
cout << age;   // Displays 17
age = 18;      // Now age is 18
cout << age;   // Displays 18
```

Assignment Statement

- Uses the = operator

- Has a single variable on the left side and a value on the right side

- Copies the value on the right into the variable on the left

```
item = 12;
```

Constants

- Constant

  - Data item whose value does not change during program execution

  - Is also called a literal

```
'A'        // character constant
"Hello"    // string literal
12         // integer constant
3.14       // floating-point constant
```

2.6 Identifiers

- Programmer-chosen names to represent parts of the program, such as variables

- Name should indicate the use of the identifier

- Cannot use C++ key words as identifiers

- Must begin with alphabetic character or _, followed by alphabetic, numeric, or _ . Alpha may be upper- or lowercase

Valid and Invalid Identifiers

IDENTIFIER	VALID?	REASON IF INVALID
totalSales	Yes	
total_Sales	Yes	
total.Sales	No	Cannot contain period
4thQtrSales	No	Cannot begin with digit
totalSale$	No	Cannot contain $

2.7 Integer Data Types

- Designed to hold whole numbers
- Can be **signed** or **unsigned**

    12   −6   +3

- Available in different sizes (*i.e.*, number of bytes): **short**, **int**, and **long**
- Size of **short** $\leq$ size of **int** $\leq$ size of **long**

Defining Variables

- Variables of the same type can be defined
  - In separate statements

    ```
    int length;

    int width;
    ```

  - In the same statement

    ```
    int length,
         width;
    ```

- Variables of different types must be defined in separate statements

Integral Constants

- To store an integer constant in a long memory location, put '`L`' at the end of the number: `1234L`

- Constants that begin with '`0`' (zero) are octal, or base 8: `075`

- Constants that begin with '`0x`' are hexadecimal, or base 16: `0x75A`

2.8 The `char` Data Type

- Used to hold single characters or very small integer values
- Usually occupies 1 byte of memory
- A numeric code representing the character is stored in memory

SOURCE CODE   **MEMORY**

`char letter = 'C';` **letter**

67

String Constant

- Can be stored a series of characters in consecutive memory locations

$$\texttt{"Hello"}$$

- Stored with the null terminator, \0, at end

H	e	l	l	o	\0

- Is comprised of characters between the " "

A character or a string constant?

- A character constant is a single character, enclosed in single quotes:

    ```
    'C'
    ```

- A string constant is a sequence of characters enclosed in double quotes:

    ```
    "Hello, there!"
    ```

- A single character in double quotes is a string constant, not a character constant:

    ```
    "C"
    ```

2.9 The C++ `string` Class

- Must `#include <string>` to create and use string objects

- Can define `string` variables in programs

  `string name;`

- Can assign values to string variables with the assignment operator

  `name = "George";`

- Can display them with `cout`

  `cout << name;`

2.10 Floating-Point Data Types

- Designed to hold real numbers

  `12.45`   `-3.8`

- Stored in a form similar to scientific notation
- Numbers are all signed
- Available in different sizes (number of bytes): `float`, `double`, and `long double`
- Size of `float` $\leq$ size of `double`

  $\leq$ size of `long double`

Floating-point Constants

- Can be represented in
    - Fixed point (decimal) notation:

        `31.4159`   `0.0000625`

    - E notation:

        `3.14159E1`   `6.25e-5`

- Are **double** by default
- Can be forced to be float `3.14159F` or long double `0.0000625L`

Assigning Floating-point Values to Integer Variables

If a floating-point value is assigned to an integer variable

– The fractional part will be truncated (*i.e.*, "chopped off" and discarded)

– The value is not rounded

```
int rainfall = 3.88;
cout << rainfall;   // Displays 3
```

2.11 The `bool` Data Type

- Represents values that are **true** or **false**

- **bool** values are stored as short integers

- **false** is represented by 0, **true** by 1

```
bool allDone = true;
bool finished = false;
```

allDone   finished

1	0

2.12 Determining the Size of a Data Type

The **sizeof** operator gives the size of any data type or variable

```
double amount;
cout << "A float is stored in "
     << sizeof(float) << " bytes\n";
cout << "Variable amount is stored in "
     << sizeof(amount) << " bytes\n";
```

2.13 More on Variable Assignments and Initialization

- Assigning a value to a variable

    - Assigns a value to a previously created variable
    - A single variable name must appear on left side of the = symbol

```
int size;
size = 5;      // legal
5 = size;      // not legal
```

Variable Assignment vs. Initialization

- Initializing a variable

  - Gives an initial value to a variable at the time it is created

  - Can initialize some or all variables of definition

    ```
    int length = 12;
    int width = 7, height = 5, area;
    ```

2.14 Scope

- The scope of a variable is that part of the program where the variable may be used

- A variable cannot be used before it is defined

```
int a;
cin >> a;    // legal
cin >> b;    // illegal
int b;
```

2.15 Arithmetic Operators

- Used for performing numeric calculations

- C++ has unary, binary, and ternary operators

    - unary (1 operand)   `-5`

    - binary (2 operands)   `13  -  7`

    - ternary (3 operands) `exp1  ?  exp2  :  exp3`

Binary Arithmetic Operators

SYMBOL	OPERATION	EXAMPLE	ans
+	addition	ans = 7 + 3;	10
-	subtraction	ans = 7 - 3;	4
*	multiplication	ans = 7 * 3;	21
/	division	ans = 7 / 3;	2
%	modulus	ans = 7 % 3;	1

/ Operator

- C++ division operator (/) performs integer division if both operands are integers

```
cout << 13 / 5;     // displays 2
cout <<  2 / 4;     // displays 0
```

- If either operand is floating-point, the result is floating-point

```
cout << 13 / 5.0;   // displays 2.6
cout << 2.0 / 4;    // displays 0.5
```

% Operator

- C++ modulus operator (%) computes the remainder resulting from integer division

  ```
  cout << 9 % 2;    // displays 1
  ```

- % requires integers for both operands

  ```
  cout << 9 % 2.0; // error
  ```

2.16 Comments

- Are used to document parts of a program
- Are written for persons reading the source code of the program
  - Indicate the purpose of the program
  - Describe the use of variables
  - Explain complex sections of code
- Are ignored by the compiler

Single-Line Comments

- Begin with **//** through to the end of line

```
int length = 12; // length in inches
int width = 15;  // width in inches
int area;        // calculated area

// Calculate rectangle area
area = length * width;
```

Multi-Line Comments

- Begin with **/*** and end with ***/**

- Can span multiple lines

```
/*----------------------------
    Here's a multi-line comment
    ----------------------------*/
```

- Can also be used as single-line comments
```
int area;   /* Calculated area */
```

Questions?

Your time to ask questions..

References

Chapter 2

Starting Out with C++ , Early Objects, Seventh Edition

by Tony Gaddis, Judy Walters, and Godfrey Muganda