Good Programming Practice 2.11

Refer to the operator precedence and associativity chart (Appendix A) when writing expressions containing many operators. Confirm that the operators in the expression are performed in the order you expect. If you're uncertain about the order of evaluation in a complex expression, break the expression into smaller statements or use parentheses to force the order of evaluation, exactly as you'd do in an algebraic expression. Be sure to observe that some operators such as assignment (=) associate right to left rather than left to right.

2.8 Wrap-Up

You learned many important basic features of C++ in this chapter, including displaying data on the screen, inputting data from the keyboard and declaring variables of fundamental types. In particular, you learned to use the output stream object `cout` and the input stream object `cin` to build simple interactive programs. We explained how variables are stored in and retrieved from memory. You also learned how to use arithmetic operators to perform calculations. We discussed the order in which C++ applies operators (i.e., the rules of operator precedence), as well as the associativity of the operators. You also learned how C++'s `if` statement allows a program to make decisions. Finally, we introduced the equality and relational operators, which you use to form conditions in `if` statements.

The non-object-oriented applications presented here introduced you to basic programming concepts. As you'll see in Chapter 3, C++ applications typically contain just a few lines of code in function `main`—these statements normally create the objects that perform the work of the application, then the objects "take over from there." In Chapter 3, you'll learn how to implement your own classes and use objects of those classes in applications.

Summary

Section 2.2 First Program in C++: Printing a Line of Text

- Single-line comments (p. 40) begin with `//`. You insert comments to document your programs and improve their readability.
- Comments do not cause the computer to perform any action (p. 41) when the program is run—they're ignored by the compiler and do not cause any machine-language object code to be generated.
- A preprocessing directive (p. 40) begins with `#` and is a message to the C++ preprocessor. Preprocessing directives are processed before the program is compiled.
- The line `#include <iostream>` (p. 40) tells the C++ preprocessor to include the contents of the input/output stream header, which contains information necessary to compile programs that use `std::cin` (p. 46) and `std::cout` (p. 41) and the stream insertion (<<, p. 42) and stream extraction (>>, p. 46) operators.
- White space (i.e., blank lines, space characters and tab characters, p. 40) makes programs easier to read. White-space characters outside of string literals are ignored by the compiler.
- C++ programs begin executing at `main` (p. 41), even if `main` does not appear first in the program.
- The keyword `int` to the left of `main` indicates that `main` "returns" an integer value.

- The body (p. 41) of every function must be contained in braces (`{` and `}`).
- A string (p. 41) in double quotes is sometimes referred to as a character string, message or string literal. White-space characters in strings are *not* ignored by the compiler.
- Most C++ statements (p. 41) end with a semicolon, also known as the statement terminator (we'll see some exceptions to this soon).
- Output and input in C++ are accomplished with streams (p. 41) of characters.
- The output stream object `std::cout`—normally connected to the screen—is used to output data. Multiple data items can be output by concatenating stream insertion (`<<`) operators.
- The input stream object `std::cin`—normally connected to the keyboard—is used to input data. Multiple data items can be input by concatenating stream extraction (`>>`) operators.
- The notation `std::cout` specifies that we are using `cout` from "namespace" `std`.
- When a backslash (i.e., an escape character) is encountered in a string of characters, the next character is combined with the backslash to form an escape sequence (p. 42).
- The newline escape sequence `\n` (p. 42) moves the cursor to the beginning of the next line on the screen.
- A message that directs the user to take a specific action is known as a prompt (p. 46).
- C++ keyword `return` (p. 42) is one of several means to exit a function.

Section 2.4 Another C++ Program: Adding Integers

- All variables (p. 45) in a C++ program must be declared before they can be used.
- A variable name is any valid identifier (p. 45) that is not a keyword. An identifier is a series of characters consisting of letters, digits and underscores (_). Identifiers cannot start with a digit. Identifiers can be any length, but some systems or C++ implementations may impose length restrictions.
- C++ is case sensitive (p. 45).
- Most calculations are performed in assignment statements (p. 47).
- A variable is a location in memory (p. 48) where a value can be stored for use by a program.
- Variables of type `int` (p. 45) hold integer values, i.e., whole numbers such as 7, –11, 0, 31914.

Section 2.5 Memory Concepts

- Every variable stored in the computer's memory has a name, a value, a type and a size.
- Whenever a new value is placed in a memory location, the process is destructive (p. 48); i.e., the new value replaces the previous value in that location. The previous value is lost.
- When a value is read from memory, the process is nondestructive (p. 49); i.e., a copy of the value is read, leaving the original value undisturbed in the memory location.
- The `std::endl` stream manipulator (p. 47) outputs a newline, then "flushes the output buffer."

Section 2.6 Arithmetic

- C++ evaluates arithmetic expressions (p. 49) in a precise sequence determined by the rules of operator precedence (p. 50) and associativity (p. 50).
- Parentheses may be used to group expressions.
- Integer division (p. 49) yields an integer quotient. Any fractional part in integer division is truncated.
- The modulus operator, `%` (p. 50), yields the remainder after integer division.

Section 2.7 Decision Making: Equality and Relational Operators

- The `if` statement (p. 53) allows a program to take alternative action based on whether a condition is met. The format for an `if` statement is

  ```
  if ( condition )
     statement;
  ```

  If the condition is true, the statement in the body of the `if` is executed. If the condition is not met, i.e., the condition is false, the body statement is skipped.
- Conditions in `if` statements are commonly formed by using equality and relational operators (p. 53). The result of using these operators is always the value true or false.
- The `using` declaration (p. 55)

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

  informs the compiler where to find `cout` (namespace `std`) and eliminates the need to repeat the `std::` prefix. The `using` directive (p. 55)

  ```
  using namespace std;
  ```

  enables the program to use all the names in any included C++ standard library header.

Self-Review Exercises

2.1 Fill in the blanks in each of the following.

a) Every C++ program begins execution at the function ________.
b) A(n) ________ begins the body of every function and a(n) ________ ends the body.
c) Most C++ statements end with a(n) ________.
d) The escape sequence `\n` represents the ________ character, which causes the cursor to position to the beginning of the next line on the screen.
e) The ________ statement is used to make decisions.

2.2 State whether each of the following is *true* or *false*. If *false*, explain why. Assume the statement `using std::cout;` is used.

a) Comments cause the computer to print the text after the `//` on the screen when the program is executed.
b) The escape sequence `\n`, when output with `cout` and the stream insertion operator, causes the cursor to position to the beginning of the next line on the screen.
c) All variables must be declared before they're used.
d) All variables must be given a type when they're declared.
e) C++ considers the variables `number` and `NuMbEr` to be identical.
f) Declarations can appear almost anywhere in the body of a C++ function.
g) The modulus operator (`%`) can be used only with integer operands.
h) The arithmetic operators `*`, `/`, `%`, `+` and `–` all have the same level of precedence.
i) A C++ program that prints three lines of output must contain three statements using `cout` and the stream insertion operator.

2.3 Write a single C++ statement to accomplish each of the following (assume that neither `using` declarations nor a `using` directive have been used):

a) Declare the variables `c`, `thisIsAVariable`, `q76354` and `number` to be of type `int` (in one statement).
b) Prompt the user to enter an integer. End your prompting message with a colon (`:`) followed by a space and leave the cursor positioned after the space.
c) Read an integer from the user at the keyboard and store it in integer variable `age`.
d) If the variable `number` is not equal to `7`, print `"The variable number is not equal to 7"`.