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

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

- Simple program that prints a line of text.

```
1  // Fig. 2.1: fig02_01.cpp
2  // Text-printing program.
3  #include <iostream> // enables program to output data to the screen
4
5  // function main begins program execution
6  int main() {
7     std::cout << "Welcome to C++!\n"; // display message
8
9     return 0; // indicate that program ended successfully
10 } // end function main
```

```
Welcome to C++!
```

First Program in C++: Printing a Line of Text (cont.)

- // indicates that the remainder of each line is a comment.
  - You insert comments to document your programs and to help other people read and understand them.
  - Comments are ignored by the C++ compiler and do not cause any machine-language object code to be generated.
- A comment beginning with // is called a single-line comment because it terminates at the end of the current line.
- You also may use comments containing one or more lines enclosed in /* and */.
- Good Programming Practice
  - Every program should begin with a comment that describes the purpose of the program.

First Program in C++: Printing a Line of Text (cont.)

- A preprocessing directive is a message to the C++ preprocessor.
- Lines that begin with # are processed by the preprocessor before the program is compiled.
- #include <iostream> notifies the preprocessor to include in the program the contents of the input/output stream header file <iostream>.
  - This header is a file containing information used by the compiler when compiling any program that outputs data to the screen or inputs data from the keyboard using C++-style stream input/output.
- Common Programming Error
  - Forgetting to include the <iostream> header in a program that inputs data from the keyboard or outputs data to the screen causes the compiler to issue an error message.

First Program in C++: Printing a Line of Text (cont.)

- You use blank lines, *space characters,* and *tab characters* (i.e., “tabs”) to make programs easier to read.
  - Together, these characters are known as white space.
  - White-space characters are normally *ignored* by the compiler.

First Program in C++: Printing a Line of Text (cont.)

- main is a part of every C++ program.
- The parentheses after main indicate that main is a program building block called a function.
- C++ programs typically consist of one or more functions and classes.
- Exactly *one* function in every program *must* be named main.
- C++ programs begin executing at function main, even if main is *not* the first function defined in the program.
- The keyword int to the left of main indicates that main "returns" an integer (whole number) value.
  - A keyword is a word in code that is reserved by C++ for a specific use.
  - For now, simply include the keyword int to the left of main in each of your programs.

First Program in C++: Printing a Line of Text (cont.)

- A left brace, {, must *begin* the body of every function.
- A corresponding right brace, }, must *end* each function's body.
- A statement instructs the computer to perform an action.
- Together, the quotation marks and the characters between them are called a string, a character string or a string literal.
- We refer to characters between double quotation marks simply as strings.
  - White-space characters in strings are not ignored by the compiler.
- Most C++ statements end with a semicolon (;), also known as the statement terminator.
  - Preprocessing directives (like #include) do not end with a semicolon.

First Program in C++: Printing a Line of Text (cont.)

- Common Programming Errors
  - Omitting the semicolon at the end of a C++ statement is a syntax error. The **syntax** of a programming language specifies the rules for creating proper programs in that language.
  - A **syntax error** occurs when the compiler encounters code that violates C++'s language rules (i.e., its syntax).
  - The compiler normally issues an error message to help you locate and fix the incorrect code.
  - Syntax errors are also called **compiler errors**, **compile-time errors,** or **compilation errors**, because the compiler detects them during the compilation phase.
  - You cannot execute your program until you correct all the syntax errors in it.
  - Some compilation errors are not syntax errors.

First Program in C++: Printing a Line of Text (cont.)

- Good Programming Practice
  - Indent the body of each function one level within the braces that delimit the function's body. This makes a program's functional structure stand out, making the program easier to read.
  - Set a convention for the size of indent you prefer, then apply it uniformly. The tab key may be used to create indents, but tab stops may vary. We can prefer three spaces per level of indent.

First Program in C++: Printing a Line of Text (cont.)

- Typically, output and input in C++ are accomplished with streams of data.
- When a cout statement executes, it sends a stream of characters to the standard output stream object—std::cout—which is normally "connected" to the screen.
- The std:: before cout is required when we use names that we've brought into the program by the preprocessing directive #include <iostream>.
  - The notation std::cout specifies that we are using a name, in this case cout, that belongs to "namespace" std.
  - The names cin (the standard input stream) and cerr (the standard error stream) also belong to namespace std.

First Program in C++: Printing a Line of Text (cont.)

- In the context of an output statement, the << operator is referred to as the stream insertion operator.
  - The value to the operator's right, the right operand, is inserted in the output stream.
- The characters \n are *not* printed on the screen.
- The backslash (\) is called an escape character.
  - It indicates that a "special" character is to be output.
- When a backslash is encountered in a string of characters, the next character is combined with the backslash to form an escape sequence.
- The escape sequence \n means newline.
  - Causes the cursor to move to the beginning of the next line on the screen.

Escape Sequences

Escape sequence	Description
`\n`	Newline. Position the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Used to print a backslash character.
`\'`	Single quote. Used to print a single-quote character.
`\"`	Double quote. Used to print a double-quote character.

First Program in C++: Printing a Line of Text (cont.)

- When the return statement is used at the end of main the value 0 indicates that the program has *terminated successfully*.
- According to the C++ standard, if program execution reaches the end of main without encountering a return statement, it's assumed that the program terminated successfully—exactly as when the last statement in main is a return statement with the value 0.

Modifying Our First C++ Program

- Welcome to C++! can be printed several ways.

```
1  // Fig. 2.3: fig02_03.cpp
2  // Printing a line of text with multiple statements.
3  #include <iostream> // enables program to output data to the screen
4
5  // function main begins program execution
6  int main() {
7     std::cout << "Welcome ";
8     std::cout << "to C++!\n";
9  } // end function main
```

```
Welcome to C++!
```

Modifying Our First C++ Program (cont.)

- A single statement can print multiple lines by using newline characters.
- Each time the \n (newline) escape sequence is encountered in the output stream, the screen cursor is positioned to the beginning of the next line.
- To get a blank line in your output, place two newline characters back to back.

```cpp
// Fig. 2.4: fig02_04.cpp
// Printing multiple lines of text with a single statement.
#include <iostream> // enables program to output data to the screen

// function main begins program execution
int main() {
   std::cout << "Welcome\nto\n\nC++!\n";
} // end function main
```

```
Welcome
to

C++!
```

Another C++ Program: Adding Integers

- The next program obtains two integers typed by a user at the keyboard, computes their sum and outputs the result using std::cout.
- The following program shows the program and sample inputs and outputs.

```
1  // Fig. 2.5: fig02_05.cpp
2  // Addition program that displays the sum of two integers.
3  #include <iostream> // enables program to perform input and output
4
5  // function main begins program execution
6  int main() {
7     // declaring and initializing variables
8     int number1{0}; // first integer to add (initialized to 0)
9     int number2{0}; // second integer to add (initialized to 0)
10    int sum{0}; // sum of number1 and number2 (initialized to 0)
11
12    std::cout << "Enter first integer: "; // prompt user for data
13    std::cin >> number1; // read first integer from user into number1
14
15    std::cout << "Enter second integer: "; // prompt user for data
16    std::cin >> number2; // read second integer from user into number2
17
18    sum = number1 + number2; // add the numbers; store result in sum
19
20    std::cout << "Sum is " << sum << std::endl; // display sum; end line
21 } // end function main
```

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```

Another C++ Program: Adding Integers (cont.)

- Declarations introduce identifiers into programs.
- The identifiers number1, number2 and sum are the names of variables.
- A variable is a location in the computer's memory where a value can be stored for use by a program.
- Variables number1, number2 and sum are data of type int, meaning that these variables will hold integers (whole numbers such as 7, –11, 0 and 31914).

Another C++ Program: Adding Integers (cont.)

- Lines 8–10 initialize each variable to 0 by placing a value in braces ({ and }) immediately following the variable's name
  - Known as list initialization
  - Introduced in C++11
- Previously, these declarations would have been written as:
  - int number1 = 0;
  - int number2 = 0;
  - int sum = 0;

Another C++ Program: Adding Integers (cont.)

- All variables *must* be declared with a *name* and a *data type* before they can be used in a program.
- If more than one name is declared in a declaration (as shown here), the names are separated by commas (,); this is referred to as a comma-separated list.
- Error Prevention Tip
  - Although it's not always necessary to initialize every variable explicitly, doing so will help you avoid many kinds of problems.
- Good Programming Practice
  - Declare only one variable in each declaration and provide a comment that explains the variable's purpose in the program.

Another C++ Program: Adding Integers (cont.)

- Data type double is for specifying real numbers, and data type char for specifying *character data*.
- Real numbers are numbers with decimal points, such as 3.4, 0.0 and –11.19.
- A char variable may hold only a single lowercase letter, a single uppercase letter, a single digit or a single special character (e.g., $ or *).
- Types such as int, double and char are called fundamental types.
- Fundamental-type names are keywords and therefore *must* appear in all lowercase letters.

Another C++ Program: Adding Integers (cont.)

Integral types	Floating-point types
bool	float
char	double
signed char	long double
unsigned char	
short int	
unsigned short int	
int	
unsigned int	
long int	
unsigned long int	
long long int	
unsigned long long int	
char16_t	
char32_t	
wchar_t	

C++ fundamental types.

Another C++ Program: Adding Integers (cont.)

- A variable name is any valid identifier that is *not* a keyword.
- An identifier is a series of characters consisting of letters, digits and underscores (_) that does not begin with a digit.
- C++ is case sensitive—uppercase and lowercase letters are different, so a1 and A1 are *different* identifiers.
- Portability Tip
  - C++ allows identifiers of any length, but your C++ implementation may restrict identifier lengths. Use identifiers of 31 characters or fewer to ensure portability (and readability)

Another C++ Program: Adding Integers (cont.)

- Good Programming Practice
  - Choosing meaningful identifiers helps make a program self documenting—a person can understand the program simply by reading it rather than having to refer to program comments or documentation.
  - Avoid using abbreviations in identifiers. This improves program readability.
  - Do not use identifiers that begin with underscores and double underscores, because C++ compilers use names like that for their own purposes internally.

Another C++ Program: Adding Integers (cont.)

- Declarations of variables can be placed almost anywhere in a program, but they must appear *before* their corresponding variables are used in the program.

Another C++ Program: Adding Integers (cont.)

- A prompt it directs the user to take a specific action.
- A cin statement uses the input stream object cin (of namespace std) and the stream extraction operator, >>, to obtain a value from the keyboard.
- Using the stream extraction operator with std::cin takes character input from the standard input stream, which is usually the keyboard.

Another C++ Program: Adding Integers (cont.)

- When the computer executes an input statement that places a value in an int variable, it waits for the user to enter a value for variable number1.
- The user responds by typing the number (as characters) then pressing the *Enter* key (sometimes called the *Return* key) to send the characters to the computer.
- The computer converts the character representation of the number to an integer and assigns (i.e., copies) this number (or value) to the variable number1.
- Any subsequent references to number1 in this program will use this same value.
- Pressing *Enter* also causes the cursor to move to the beginning of the next line on the screen.

Another C++ Program: Adding Integers (cont.)

- In this program, an assignment statement adds the values of variables number1 and number2 and assigns the result to variable sum using the assignment operator =.
  - Most calculations are performed in assignment statements.
- The = operator and the + operator are called binary operators because each has two operands.
- Good Programming Practice
  - Place spaces on either side of a binary operator. This makes the operator stand out and makes the program more readable.

Another C++ Program: Adding Integers (cont.)

- std::endl is a so-called stream manipulator.
- The name endl is an abbreviation for “end line” and belongs to namespace std.
- The std::endl stream manipulator outputs a newline, then “flushes the output buffer.”
  - This simply means that, on some systems where outputs accumulate in the machine until there are enough to “make it worthwhile” to display them on the screen, std::endl forces any accumulated outputs to be displayed at that moment.
  - This can be important when the outputs are prompting the user for an action, such as entering data.

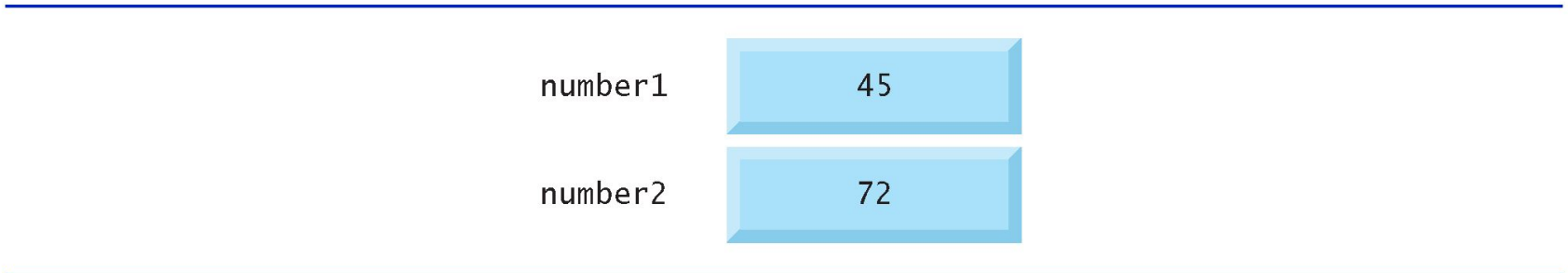
Another C++ Program: Adding Integers (cont.)

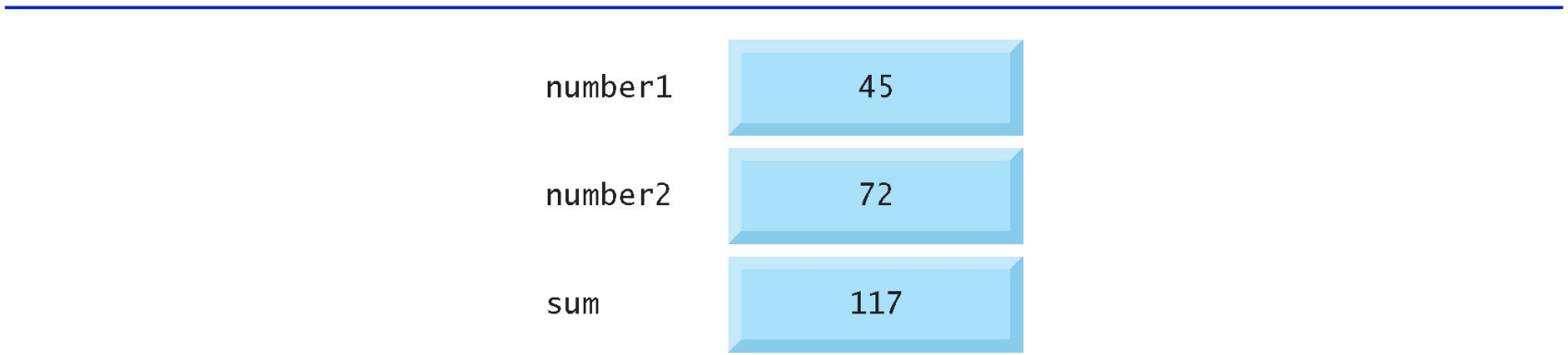
- Using multiple stream insertion operators (<<) in a single statement is referred to as concatenating, chaining or cascading stream insertion operations.
- Calculations can also be performed in output statements.

Memory Concepts

- Variable names such as number1, number2, and sum actually correspond to locations in the computer's memory.
- Every variable has a name, a type, a size, and a value.
- When a value is placed in a memory location, the value overwrites the previous value in that location; thus, placing a new value into a memory location is said to be destructive.
- When a value is read out of a memory location, the process is nondestructive.


number1
45


number1
45
number2
72


number1
45
number2
72
sum
117

Arithmetic

- Most programs perform arithmetic calculations.
- Figure 2.9 summarizes the C++ arithmetic operators.
- The asterisk (*) indicates multiplication.
- The percent sign (%) is the remainder operator.
  - Yields the remainder after integer division.
  - Can be used only with integer operands.
- The arithmetic operators are all binary operators.
- Integer division (i.e., where both the numerator and the denominator are integers) yields an integer quotient.
  - Any fractional part in integer division is discarded (i.e., truncated)—no rounding occurs.

Operation	Arithmetic operator	Algebraic expression	C++ expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm or $b \cdot m$	`b * m`
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`

Arithmetic (cont.)

- Arithmetic expressions in C++ must be entered into the computer in straight-line form.
- Expressions such as “a divided by b” must be written as a / b, so that all constants, variables and operators appear in a straight line.
- Parentheses are used in C++ expressions in the same manner as in algebraic expressions.
- For example, to multiply a times the quantity b + c we write
  - a * (b + c)

Arithmetic (cont.)

- C++ applies the operators in arithmetic expressions in a precise sequence determined by the following rules of operator precedence, which are generally the same as those followed in algebra.

Operator(s)	Operation(s)	Order of evaluation (precedence)
`( )`	Parentheses	Evaluated first. For *nested* parentheses, such as in the expression `a * (b + c / (d + e))`, the expression in the *innermost* pair evaluates first. [*Caution:* If you have an expression such as `(a + b) * (c - d)` in which two sets of parentheses are not nested, but appear "on the same level," the C++ Standard does *not* specify the order in which these parenthesized subexpressions will evaluate.]

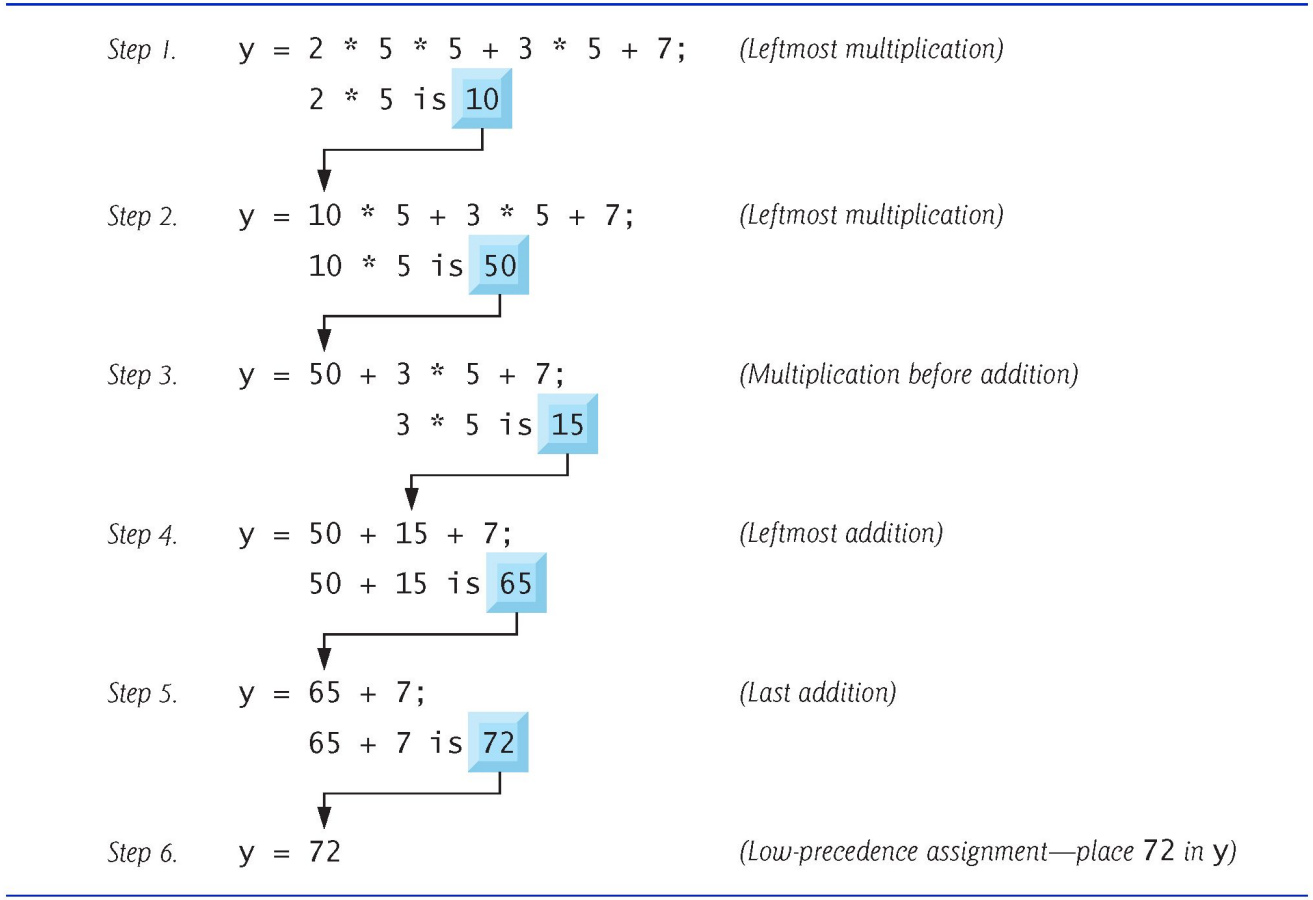
`*`
`/`
`%`
	Multiplication
	Division
	Remainder
		Evaluated second. If there are several, they're evaluated left to right.

`+` `-`	Addition Subtraction	Evaluated last. If there are several, they're evaluated left to right.

Arithmetic (cont.)

- There is no arithmetic operator for exponentiation in C++, so x^2 is represented as x * x.
- Figure 2.11 illustrates the order in which the operators in a second-degree polynomial are applied.
- As in algebra, it's acceptable to place redundant parentheses in an expression to make the expression clearer.

Second-Degree Polynomial Evaluation Order



Decision Making: Equality and Relational Operators

- The if statement allows a program to take alternative action based on whether a condition is true or false.
  - If the condition is true, the statement in the body of the if statement is executed.
  - If the condition is false, the body statement is not executed.
- Conditions in if statements can be formed by using the equality operators and relational operators (Fig. 2.12).
- The relational operators all have the same level of precedence and associate left to right.
- The equality operators both have the same level of precedence, which is lower than that of the relational operators, and associate left to right.

Decision Making: Equality and Relational Operators (cont.)

Algebraic relational or equality operator	C++ relational or equality operator	Sample C++ condition	Meaning of C++ condition
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

Decision Making: Equality and Relational Operators (cont.)

- Common Programming Error
  - Reversing the order of the pair of symbols in the operators !=, >= and <= (by writing them as =!, => and =<, respectively) is normally a syntax error. In some cases, writing != as =! will not be a syntax error, but almost certainly will be a logic error that has an effect at execution time. A fatal logic error causes a program to fail and terminate prematurely. A nonfatal logic error allows a program to continue executing, but usually produces incorrect results.
  - Confusing the equality operator == with the assignment operator = results in logic errors. We like to read the equality operator as “is equal to” or “double equals,” and the assignment operator as “gets” or “gets the value of” or “is assigned the value of.” Confusing these operators may not necessarily cause an easy-to-recognize syntax error, but may cause subtle logic errors.

Decision Making: Equality and Relational Operators (cont.)

- Fig. 2.12 uses six if statements to compare two numbers input by the user.
- If the condition in any of these if statements is satisfied, the output statement associated with that if statement executes.
- Figure 2.13 shows the program and the input/output dialogs of three sample executions.

```cpp
// Fig. 2.13: fig02_13.cpp
// Comparing integers using if statements, relational operators
// and equality operators.
#include <iostream> // enables program to perform input and output

using std::cout; // program uses cout
using std::cin; // program uses cin
using std::endl; // program uses endl

// function main begins program execution
int main() {
   int number1{0}; // first integer to compare (initialized to 0)
   int number2{0}; // second integer to compare (initialized to 0)

   cout << "Enter two integers to compare: "; // prompt user for data
   cin >> number1 >> number2; // read two integers from user

   if (number1 == number2) {
      cout << number1 << " == " << number2 << endl;
   }

```

```
22      if (number1 != number2) {
23         cout << number1 << " != " << number2 << endl;
24      }
25
26      if (number1 < number2) {
27         cout << number1 << " < " << number2 << endl;
28      }
29
30      if (number1 > number2) {
31         cout << number1 << " > " << number2 << endl;
32      }
33
34      if (number1 <= number2) {
35         cout << number1 << " <= " << number2 << endl;
36      }
37
38      if (number1 >= number2) {
39         cout << number1 << " >= " << number2 << endl;
40      }
41   } // end function main
```

```
Enter two integers to compare: 3 7
3 != 7
3 < 7
3 <= 7
```

```
Enter two integers to compare: 22 12
22 != 12
22 > 12
22 >= 12
```

```
Enter two integers to compare: 7 7
7 == 7
7 <= 7
7 >= 7
```

Decision Making: Equality and Relational Operators (cont.)

- using declarations eliminate the need to repeat the std:: prefix.
  - Can write cout instead of std::cout, cin instead of std::cin and endl instead of std::endl, respectively, in the remainder of the program.
- Many programmers prefer to use the declaration using namespace std; which enables a program to use all the names in any standard C++ header file (such as <iostream>) that a program might include.

Decision Making: Equality and Relational Operators (cont.)

- Each if statement in Fig. 2.13 has a single statement in its body and each body statement is indented.
- Each if statement's body is enclosed in a pair of braces, { }, creating what's called a compound statement or a block that may contain multiple statements.

Decision Making: Equality and Relational Operators (cont.)

- Good Programming Practice
  - Indent the statement(s) in the body of an if statement to enhance readability.
- Error Prevention Tip
  - You don’t need to use braces, { }, around single-statement bodies, but you must include the braces around multiple-statement bodies. You’ll see later that forgetting to enclose multiple-statement bodies in braces leads to errors. To avoid errors, as a rule, always enclose an if statement’s body statement(s) in braces.
- Common Programming Error
  - Placing a semicolon immediately after the right parenthesis after the condition in an if statement is often a logic error (although not a syntax error). The semicolon causes the body of the if statement to be empty, so the if statement performs no action, regardless of whether or not its condition is true. Worse yet, the original body statement of the if statement now becomes a statement in sequence with the if statement and always executes, often causing the program to produce incorrect results.

Decision Making: Equality and Relational Operators (cont.)

- Statements may be split over several lines and may be spaced according to your preferences.
- It’s a syntax error to split identifiers, strings (such as "hello") and constants (such as the number 1000) over several lines.
- Good Programming Practice
  - A lengthy statement may be spread over several lines. If a statement must be split across lines, choose meaningful breaking points, such as after a comma in a comma-separated list, or after an operator in a lengthy expression. If a statement is split across two or more lines, indent all subsequent lines and left-align the group of indented lines

Decision Making: Equality and Relational Operators (cont.)

- Figure 2.14 shows the precedence and associativity of the operators introduced in this chapter.
- The operators are shown top to bottom in decreasing order of precedence.
- All these operators, with the exception of the assignment operator =, associate from left to right.

Precedence and Associativity of Operators (cont.)

Operators				Associativity	Type
()				*[See caution in Fig. 2.10]*	grouping parentheses
*	/	%		left to right	multiplicative
+	-			left to right	additive
<<	>>			left to right	stream insertion/extraction
<	<=	>	>=	left to right	relational
==	!=			left to right	equality
=				right to left	assignment