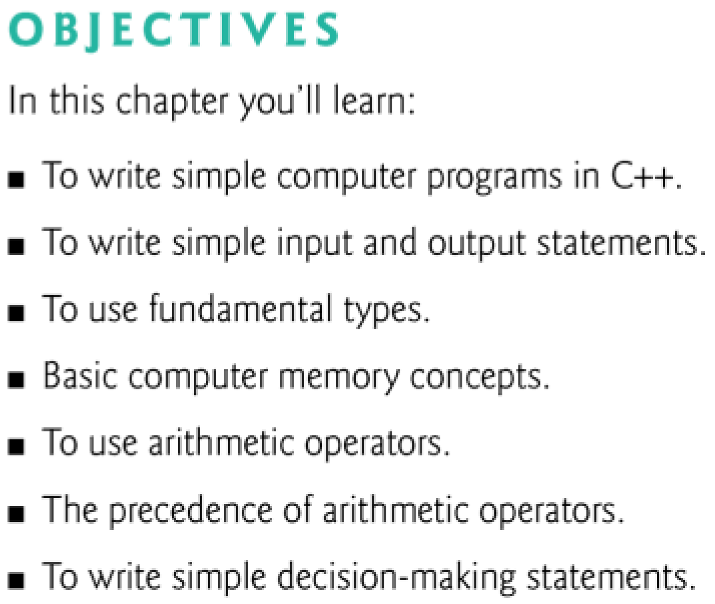
Introduction to C++ Programming

C++ How to Program, 7/e

©1992-2010 by Pearson Education, Inc. All Rights Reserved.



OBJECTIVES

In this chapter you'll learn:

- To write simple computer programs in C++.
- To write simple input and output statements.
- To use fundamental types.
- Basic computer memory concepts.
- To use arithmetic operators.
- The precedence of arithmetic operators.
- To write simple decision-making statements.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

2.1 Introduction

- We now introduce C++ programming, which facilitates a disciplined approach to program design.
- Most of the C++ programs you'll study in this book process information and display results.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

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

- Simple program that prints a line of text (Fig. 2.1).

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

```cpp
// Fig. 2.1: fig02_01.cpp
// Text-printing program.
#include <iostream> // allows program to output data to the screen

// function main begins program execution
int main()
{
   std::cout << "Welcome to C++!\n"; // display message

   return 0; // indicate that program ended successfully
} // end function main
```

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

Fig. 2.1 | Text-printing program.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

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

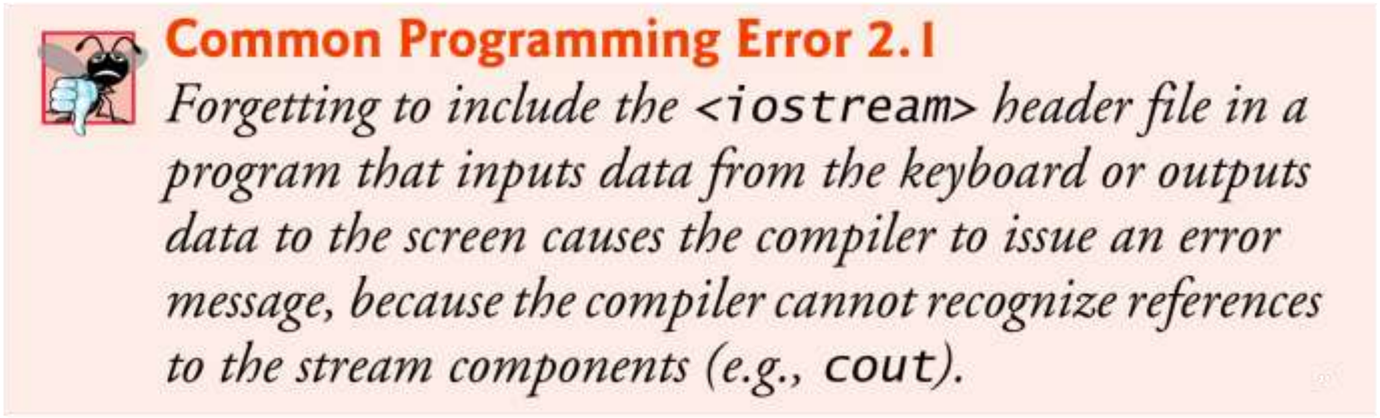
- // indicates that the remainder of each line is a comment.
  - You insert comments to document your programs and to help other people read and understand them.
  - Comments are ignored by the C++ compiler and do not cause any machine-language object code to be generated.
- A comment beginning with // is called a single-line comment because it terminates at the end of the current line.
- You also may use C's style in which a comment—possibly containing many lines—begins with /* and ends with */.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

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

- A preprocessor 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>.
  - Must be included for any program that outputs data to the screen or inputs data from the keyboard using C++-style stream input/output.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.


Common Programming Error 2.1

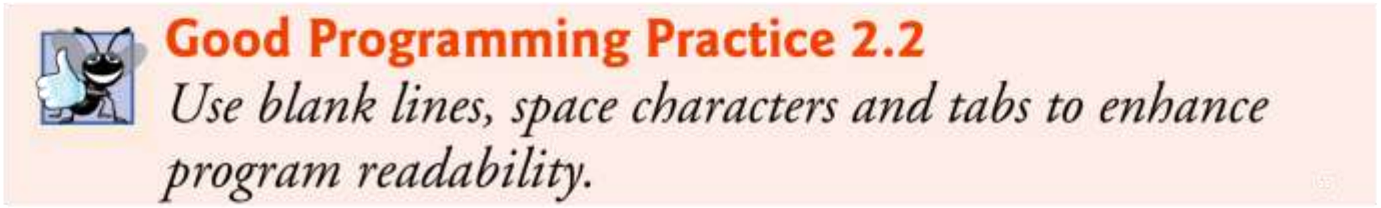
Forgetting to include the `<iostream>` *header file in a program that inputs data from the keyboard or outputs data to the screen causes the compiler to issue an error message, because the compiler cannot recognize references to the stream components (e.g.,* `cout`*).*

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

2.2 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.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.



Good Programming Practice 2.2

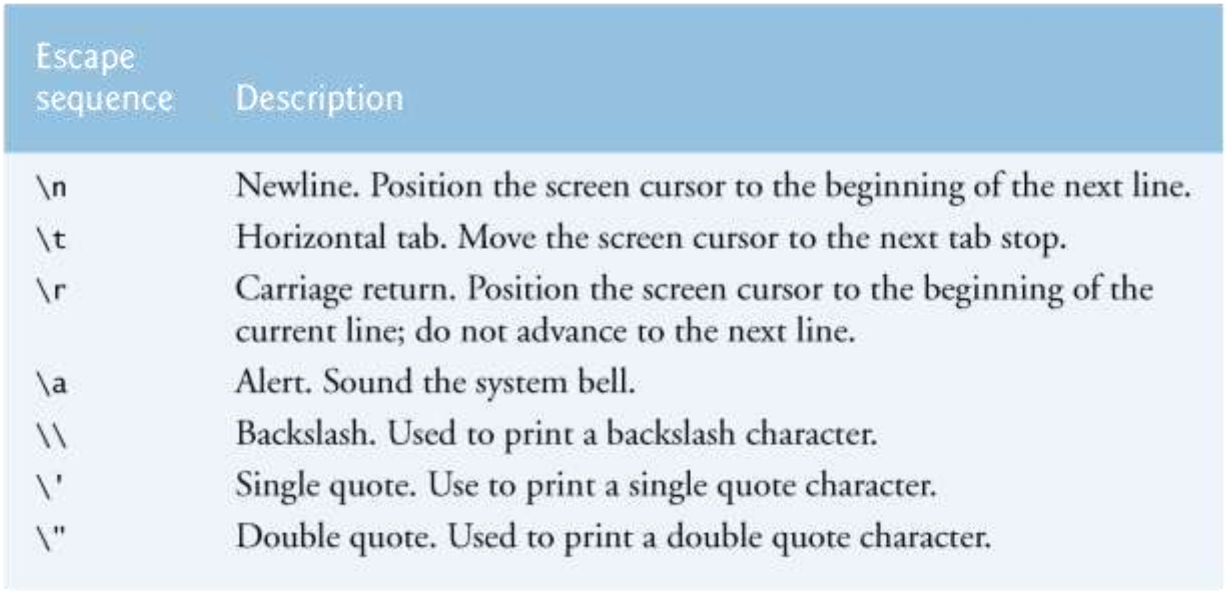
Use blank lines, space characters and tabs to enhance program readability.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

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

- 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 preprocessor 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`.
- 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.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.



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. Use to print a single quote character.
\"	Double quote. Used to print a double quote character.

Fig. 2.2 | Escape sequences.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

2.3 Modifying Our First C++ Program

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

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

```cpp
// Fig. 2.3: fig02_03.cpp
// Printing a line of text with multiple statements.
#include <iostream> // allows program to output data to the screen

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

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

Fig. 2.3 | Printing a line of text with multiple statements.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

2.3 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.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

```cpp
// Fig. 2.4: fig02_04.cpp
// Printing multiple lines of text with a single statement.
#include <iostream> // allows 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++!
```

Fig. 2.4 | Printing multiple lines of text with a single statement.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

2.4 Another C++ Program: Adding Integers

- The input stream object std::cin and the stream extraction operator-, >>, can be used obtain data from the user at the keyboard.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

```cpp
// Fig. 2.5: fig02_05.cpp
// Addition program that displays the sum of two integers.
#include <iostream> // allows program to perform input and output

// function main begins program execution
int main()
{
   // variable declarations
   int number1; // first integer to add
   int number2; // second integer to add
   int sum; // sum of number1 and number2

   std::cout << "Enter first integer: "; // prompt user for data
   std::cin >> number1; // read first integer from user into number1

   std::cout << "Enter second integer: "; // prompt user for data
   std::cin >> number2; // read second integer from user into number2

   sum = number1 + number2; // add the numbers; store result in sum

   std::cout << "Sum is " << sum << std::endl; // display sum; end line
} // end function main
```

Fig. 2.5 | Addition program that displays the sum of two integers entered at the keyboard. (Part 1 of 2.)

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

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

Fig. 2.5 | Addition program that displays the sum of two integers entered at the keyboard. (Part 2 of 2.)

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

2.4 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.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

2.4 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.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

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

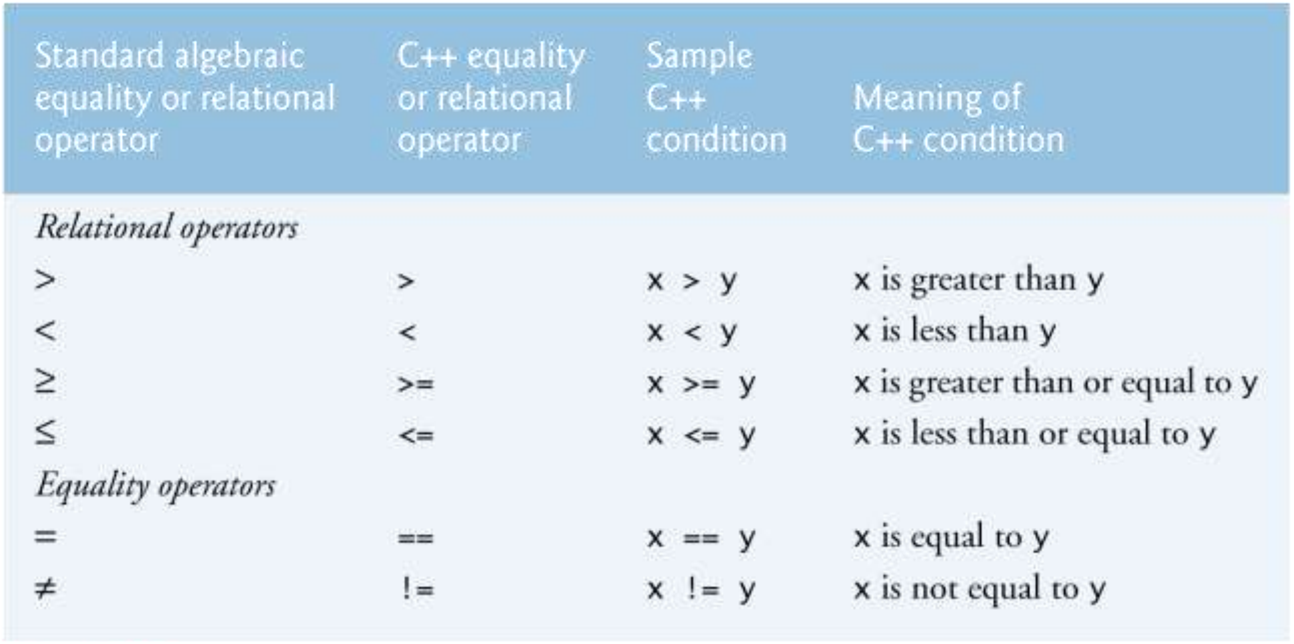
- Using multiple stream insertion operators (<<) in a single statement is referred to as concatenating, chaining or cascading stream insertion operations.
- Calculations can also be performed in output statements.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

2.7 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 summarized in 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.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.



Standard algebraic equality or relational operator	C++ equality or relational 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

Fig. 2.12 | Equality and relational operators.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

```cpp
// Fig. 2.13: fig02_13.cpp
// Comparing integers using if statements, relational operators
// and equality operators.
#include <iostream> // allows 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; // first integer to compare
   int number2; // second integer to compare

   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;

```

Fig. 2.13 | Comparing integers using `if` statements, relational operators and equality operators. (Part 1 of 3.)

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

```
   if ( number1 != number2 )
      cout << number1 << " != " << number2 << endl;

   if ( number1 < number2 )
      cout << number1 << " < " << number2 << endl;

   if ( number1 > number2 )
      cout << number1 << " > " << number2 << endl;

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

   if ( number1 >= number2 )
      cout << number1 << " >= " << number2 << endl;
} // end function main
```

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

Fig. 2.13 | Comparing integers using `if` statements, relational operators and equality operators. (Part 2 of 3.)

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

```
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
```

Fig. 2.13 | Comparing integers using `if` statements, relational operators and equality operators. (Part 3 of 3.)

©1992-2010 by Pearson Education, Inc. All Rights Reserved.

Operators				Associativity	Type
()				left to right	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

Fig. 2.14 | Precedence and associativity of the operators discussed so far.

©1992-2010 by Pearson Education, Inc. All Rights Reserved.