Chapter 3:

Expressions and Interactivity

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.1

The cin Object

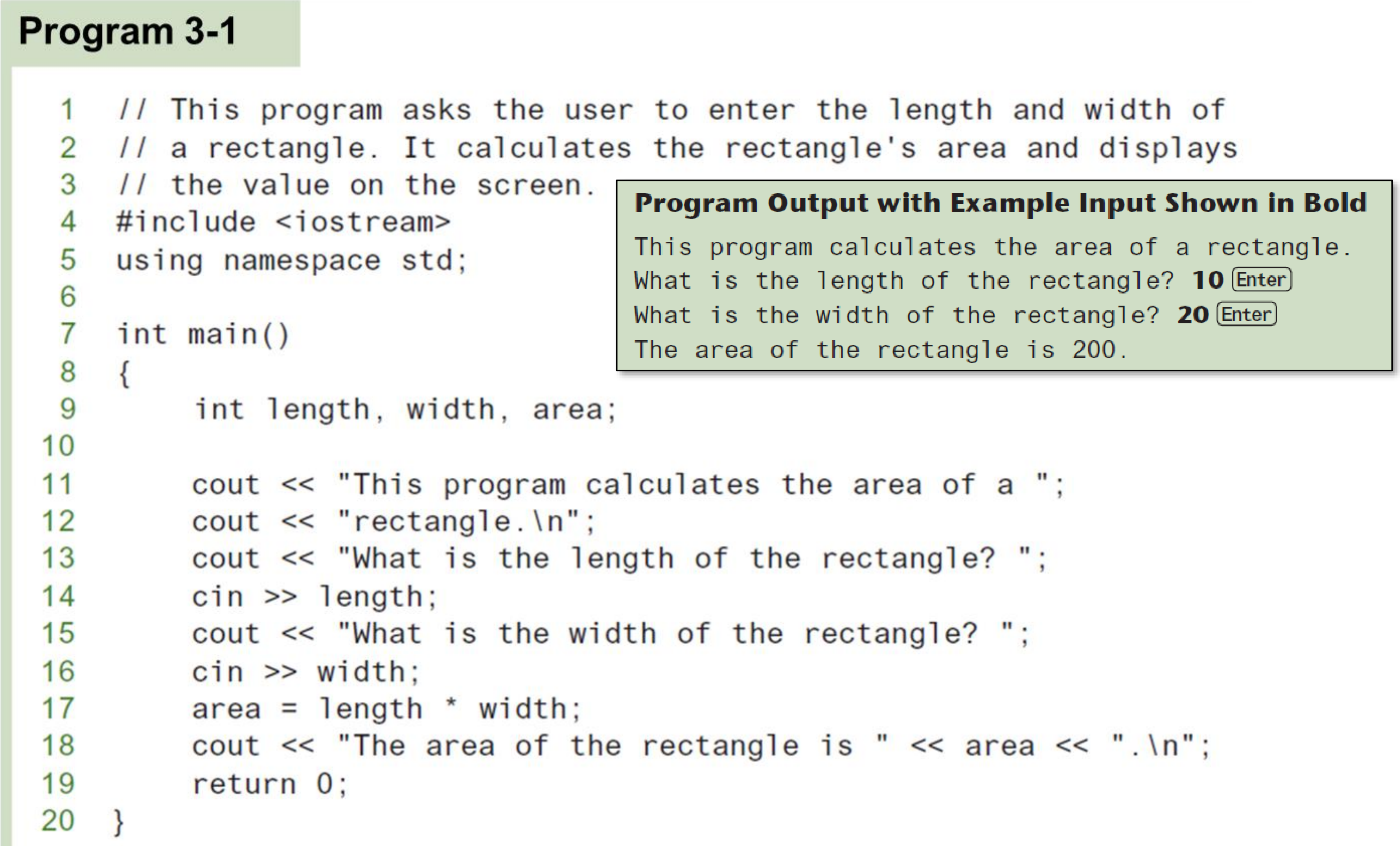
Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

The `cin` Object (1 of 4)

- Standard input object
- Like `cout`, requires `iostream` file
- Used to read input from keyboard
- Information retrieved from `cin` with `>>`
- Input is stored in one or more variables

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

The `cin` Object (2 of 4)



Program 3-1

```cpp
// This program asks the user to enter the length and width of
// a rectangle. It calculates the rectangle's area and displays
// the value on the screen.
#include <iostream>
using namespace std;

int main()
{
    int length, width, area;

    cout << "This program calculates the area of a ";
    cout << "rectangle.\n";
    cout << "What is the length of the rectangle? ";
    cin >> length;
    cout << "What is the width of the rectangle? ";
    cin >> width;
    area = length * width;
    cout << "The area of the rectangle is " << area << ".\n";
    return 0;
}
```

Program Output with Example Input Shown in Bold

```
This program calculates the area of a rectangle.
What is the length of the rectangle? 10 Enter
What is the width of the rectangle? 20 Enter
The area of the rectangle is 200.
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

The `cin` Object (3 of 4)

- **`cin`** converts data to the type that matches the variable:

```
int height;
cout << "How tall is the room? ";
cin >> height;
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Displaying a Prompt

- A prompt is a message that instructs the user to enter data.
- You should always use **`cout`** to display a prompt before each **`cin`** statement.

```
cout << "How tall is the room? ";
cin >> height;
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

The `cin` Object (4 of 4)

- Can be used to input more than one value:

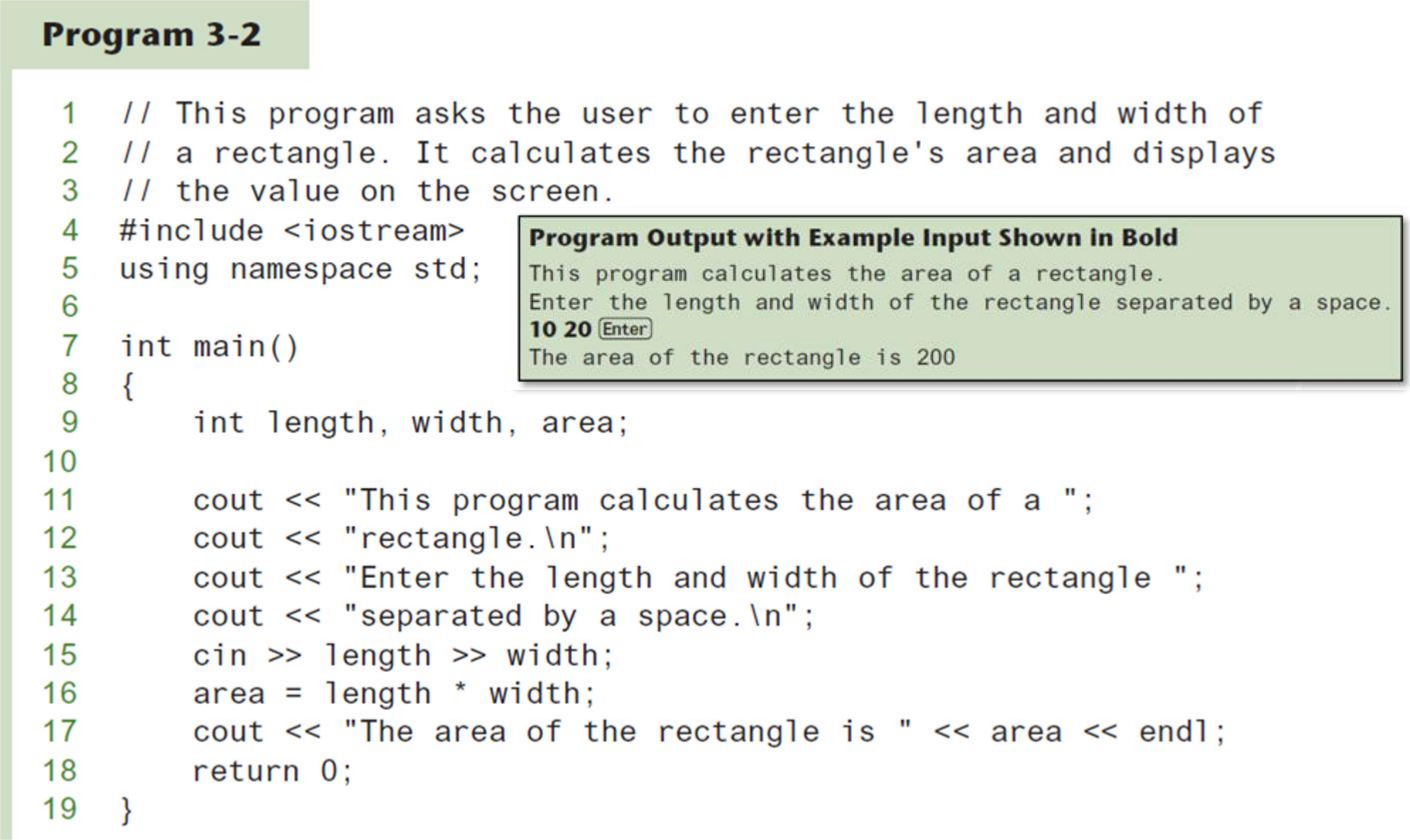
```
cin >> height >> width;
```

- Multiple values from keyboard must be separated by spaces

- Order is important: first value entered goes to first variable, etc.

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Using `cin` To Read Multiple Values



Program 3-2

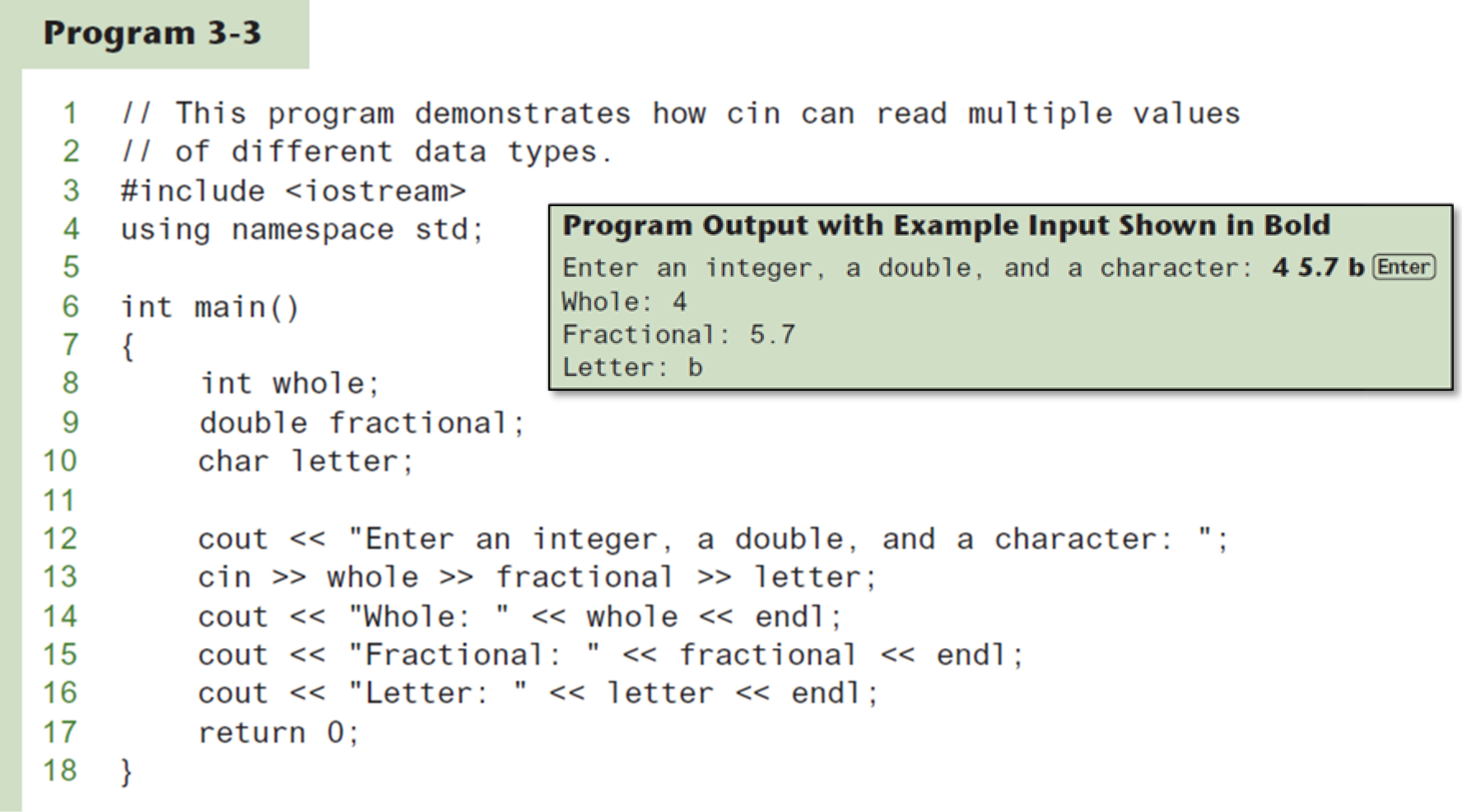
```
// This program asks the user to enter the length and width of
// a rectangle. It calculates the rectangle's area and displays
// the value on the screen.
#include <iostream>
using namespace std;

int main()
{
    int length, width, area;

    cout << "This program calculates the area of a ";
    cout << "rectangle.\n";
    cout << "Enter the length and width of the rectangle ";
    cout << "separated by a space.\n";
    cin >> length >> width;
    area = length * width;
    cout << "The area of the rectangle is " << area << endl;
    return 0;
}
```

Program Output with Example Input Shown in Bold

```
This program calculates the area of a rectangle.
Enter the length and width of the rectangle separated by a space.
10 20 [Enter]
The area of the rectangle is 200
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Using `cin` To Read Values of Different Data Types



Program 3-3

```
// This program demonstrates how cin can read multiple values
// of different data types.
#include <iostream>
using namespace std;

int main()
{
     int whole;
     double fractional;
     char letter;

     cout << "Enter an integer, a double, and a character: ";
     cin >> whole >> fractional >> letter;
     cout << "Whole: " << whole << endl;
     cout << "Fractional: " << fractional << endl;
     cout << "Letter: " << letter << endl;
     return 0;
}
```

Program Output with Example Input Shown in Bold

```
Enter an integer, a double, and a character: 4 5.7 b [Enter]
Whole: 4
Fractional: 5.7
Letter: b
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.2

Mathematical Expressions

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Mathematical Expressions

- Can create complex expressions using multiple mathematical operators
- An expression can be a literal, a variable, or a mathematical combination of constants and variables
- Can be used in assignment, `cout`, other statements:

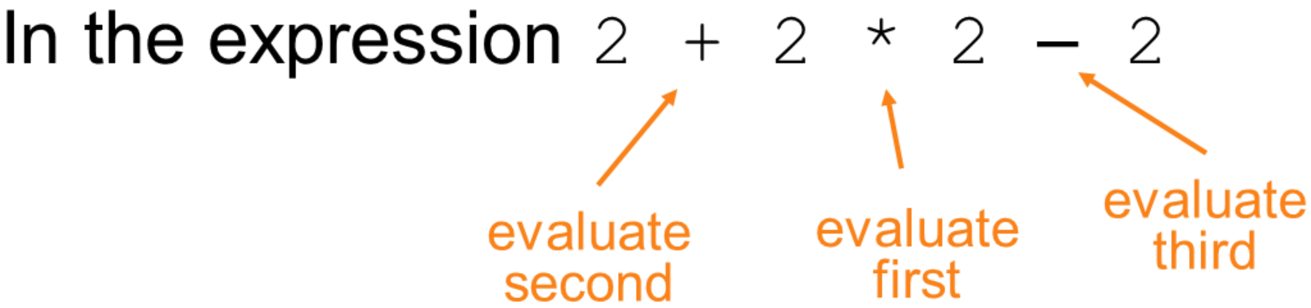
```
area = 2 * PI * radius;
cout << "border is: " << 2*(l+w);
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Order of Operations (1 of 2)

In an expression with more than one operator, evaluate in this order:

- – (unary negation), in order, left to right
- * / %, in order, left to right
- + –, in order, left to right



In the expression 2 + 2 * 2 – 2

evaluate second

evaluate first

evaluate third

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Order of Operations (2 of 2)

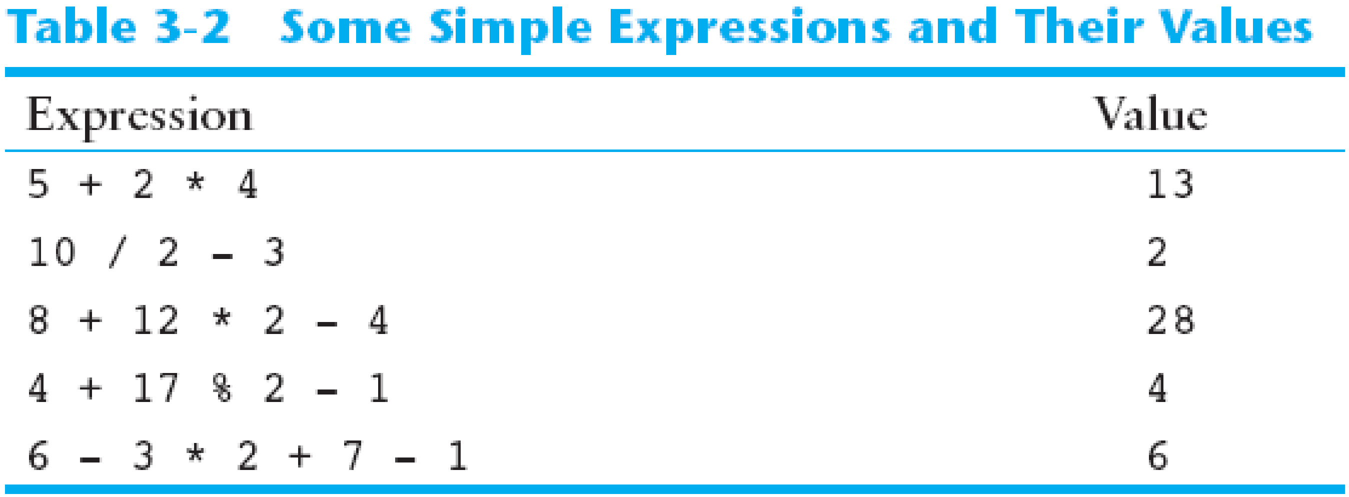


Table 3-2 Some Simple Expressions and Their Values

Expression	Value
5 + 2 * 4	13
10 / 2 - 3	2
8 + 12 * 2 - 4	28
4 + 17 % 2 - 1	4
6 - 3 * 2 + 7 - 1	6

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Associativity of Operators

- $-$ (unary negation) associates right to left
- $*, \ /, \ \%, +, -$ associate right to left
- parentheses ( ) can be used to override the order of operations:

$$2 + 2 * 2 - 2 = 4$$

$$(2 + 2) * 2 - 2 = 6$$

$$2 + 2 * (2 - 2) = 2$$

$$(2 + 2) * (2 - 2) = 0$$

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Grouping with Parentheses

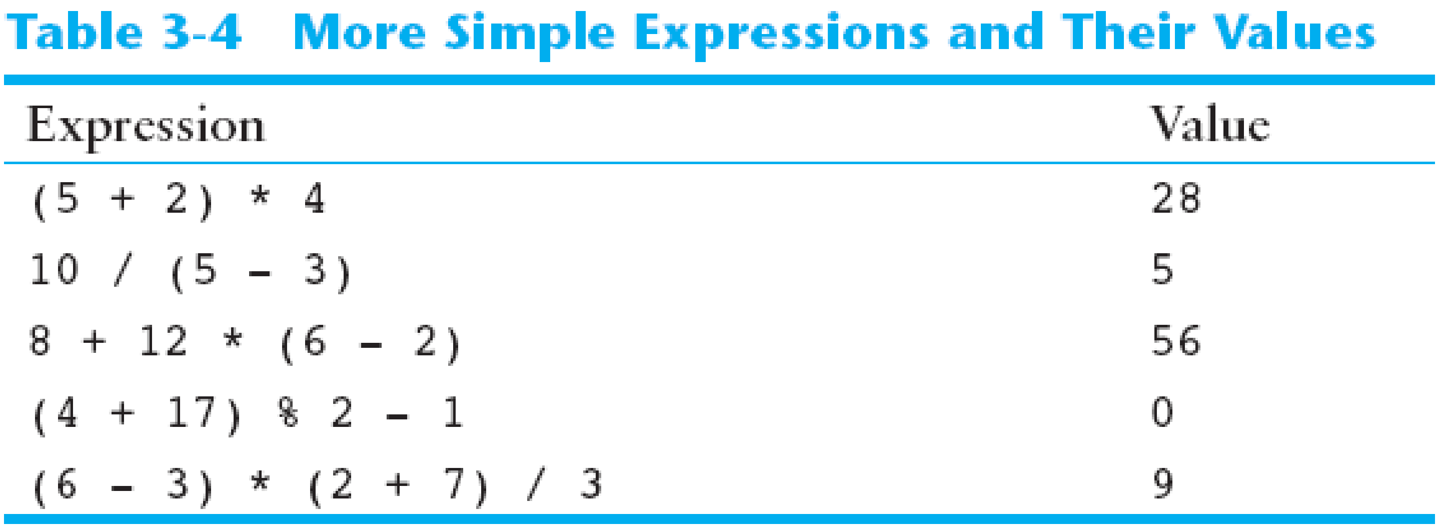


Table 3-4 More Simple Expressions and Their Values

Expression	Value
`(5 + 2) * 4`	28
`10 / (5 - 3)`	5
`8 + 12 * (6 - 2)`	56
`(4 + 17) % 2 - 1`	0
`(6 - 3) * (2 + 7) / 3`	9

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Algebraic Expressions (1 of 2)

- Multiplication requires an operator:

  $Area = lw$ is written as `Area = l * w;`

- There is no exponentiation operator:

  $Area = s^2$ is written as `Area = pow(s, 2);`

- Parentheses may be needed to maintain order of operations:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

is written as

`m = (y2 - y1) / (x2 - x1);`

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Algebraic Expressions (2 of 2)

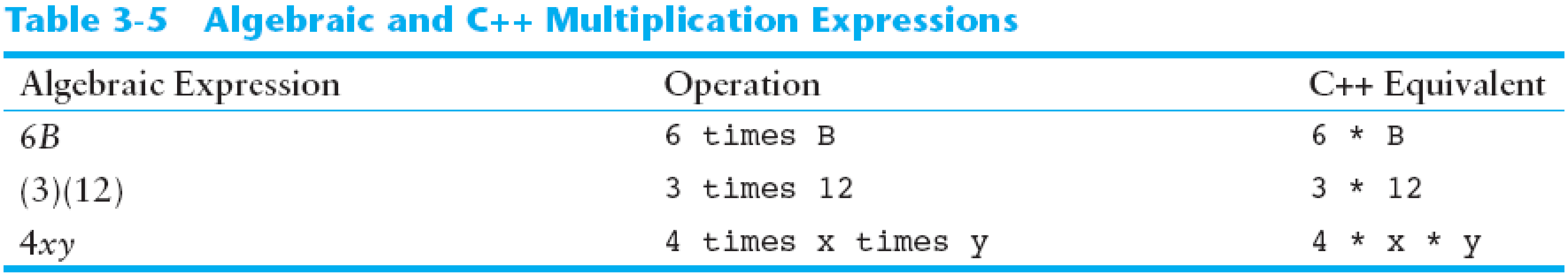


Table 3-5 Algebraic and C++ Multiplication Expressions

Algebraic Expression	Operation	C++ Equivalent
$6B$	6 times B	6 * B
$(3)(12)$	3 times 12	3 * 12
$4xy$	4 times x times y	4 * x * y

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.3

When You Mix Apples with Oranges: Type Conversion

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

When You Mix Apples with Oranges: Type Conversion

- Operations are performed between operands of the same type.
- If not of the same type, C++ will convert one to be the type of the other
- This can impact the results of calculations.

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Hierarchy of Types

Highest:

```
long double
double
float
unsigned long
long
unsigned int
int
```

Lowest:

Ranked by largest number they can hold

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Type Coercion

- Type Coercion: automatic conversion of an operand to another data type
- Promotion: convert to a higher type
- Demotion: convert to a lower type

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Coercion Rules

1) `char`, `short`, `unsigned short` automatically promoted to `int`
2) When operating on values of different data types, the lower one is promoted to the type of the higher one.
3) When using the `=` operator, the type of expression on right will be converted to type of variable on left

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.4

Overflow and Underflow

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Overflow and Underflow

- Occurs when assigning a value that is too large (overflow) or too small (underflow) to be held in a variable
- Variable contains value that is ‘wrapped around’ set of possible values
- Different systems may display a warning/error message, stop the program, or continue execution using the incorrect value

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.5

Type Casting

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Type Casting

- Used for manual data type conversion
- Useful for floating point division using ints:

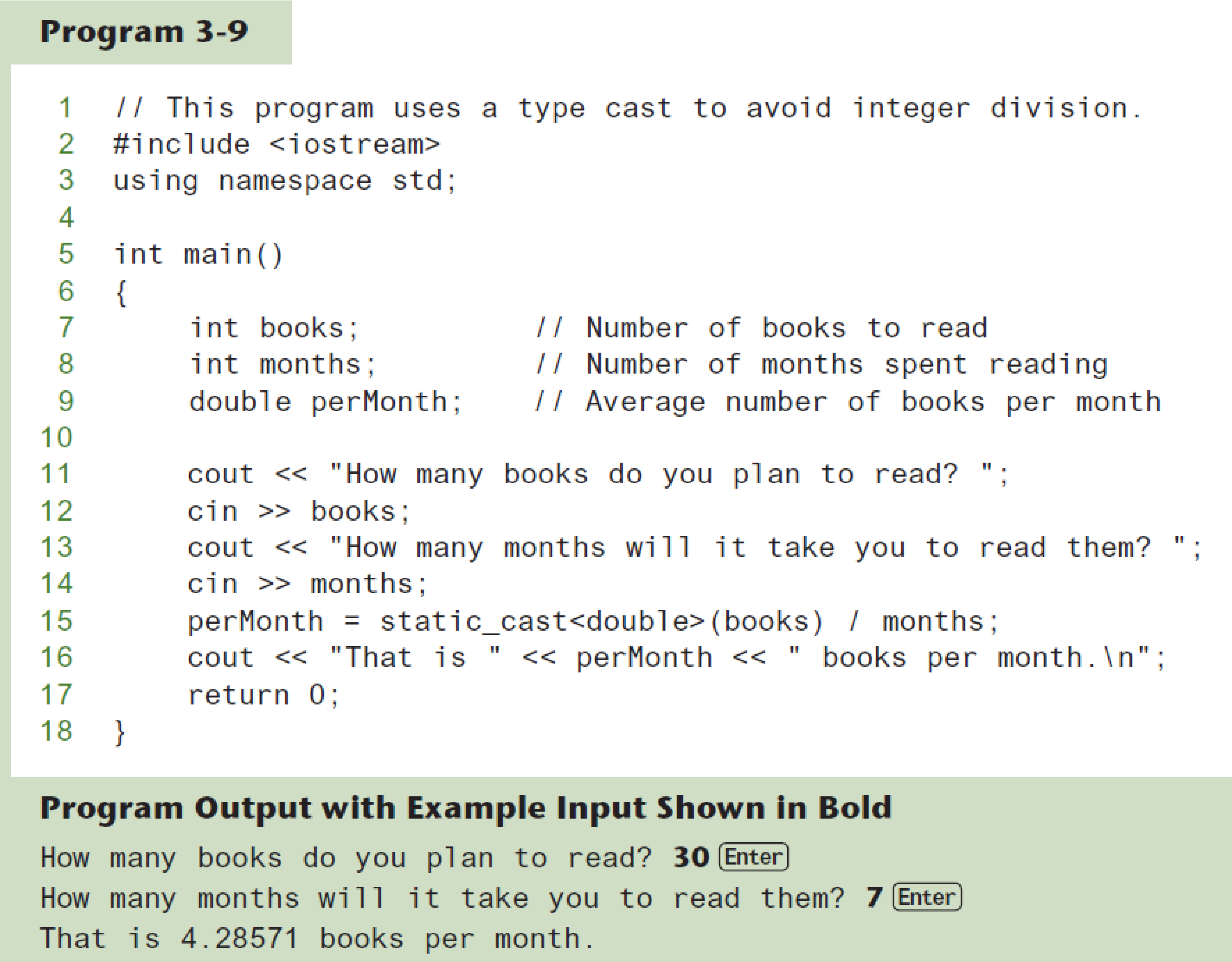
```
double m;
m = static_cast<double>(y2-y1)
                      /(x2-x1);
```

- Useful to see `int` value of a `char` variable:

```
char ch = 'C';
cout << ch << " is "
     << static_cast<int>(ch);
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Type Casting (2 of 2)



Program 3-9

```
// This program uses a type cast to avoid integer division.
#include <iostream>
using namespace std;

int main()
{
    int books;           // Number of books to read
    int months;          // Number of months spent reading
    double perMonth;     // Average number of books per month

    cout << "How many books do you plan to read? ";
    cin >> books;
    cout << "How many months will it take you to read them? ";
    cin >> months;
    perMonth = static_cast<double>(books) / months;
    cout << "That is " << perMonth << " books per month.\n";
    return 0;
}
```

Program Output with Example Input Shown in Bold

```
How many books do you plan to read? 30 Enter
How many months will it take you to read them? 7 Enter
That is 4.28571 books per month.
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

C-Style and Prestandard Type Cast Expressions

- C-Style cast: data type name in ()

  ```
  cout << ch << " is " << (int)ch;
  ```

- Prestandard C++ cast: value in ()

  ```
  cout << ch << " is " << int(ch);
  ```

- Both are still supported in C++, although static_cast is preferred

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.6

Multiple Assignment and Combined Assignment

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

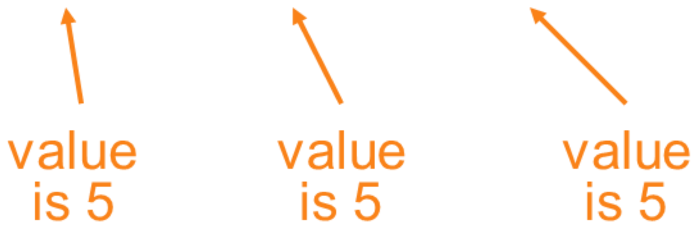
Multiple Assignment and Combined Assignment

- The = can be used to assign a value to multiple variables:

```
x = y = z = 5;
```

- Value of = is the value that is assigned
- Associates right to left:

```
x = (y = (z = 5));
```



value is 5

value is 5

value is 5

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Combined Assignment (1 of 2)

- Look at the following statement:

```
sum = sum + 1;
```

This adds 1 to the variable **`sum`**.

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Other Similar Statements

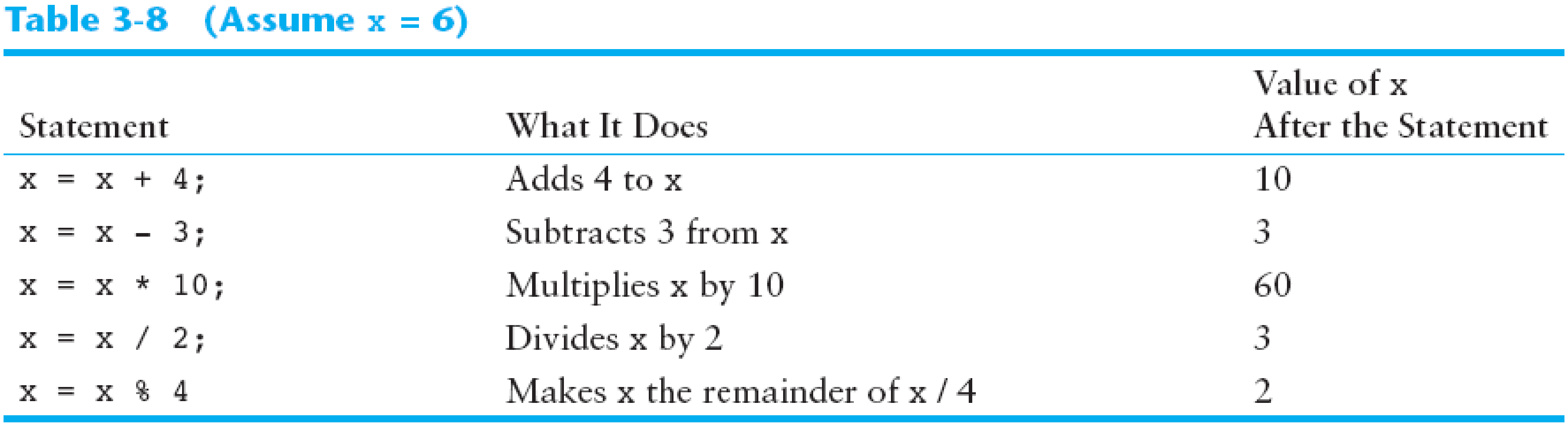


Table 3-8 (Assume x = 6)

Statement	What It Does	Value of x After the Statement
`x = x + 4;`	Adds 4 to x	10
`x = x - 3;`	Subtracts 3 from x	3
`x = x * 10;`	Multiplies x by 10	60
`x = x / 2;`	Divides x by 2	3
`x = x % 4`	Makes x the remainder of x / 4	2

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Combined Assignment (2 of 2)

- The combined assignment operators provide a shorthand for these types of statements.
- The statement

```
    sum = sum + 1;
```

is equivalent to

```
  sum += 1;
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Combined Assignment Operators

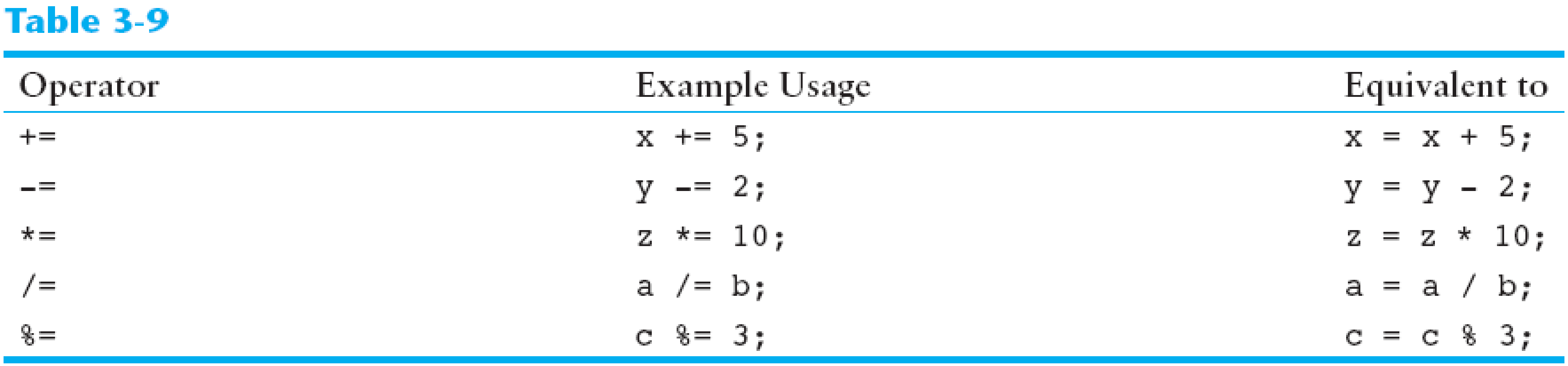


Table 3-9

Operator	Example Usage	Equivalent to
`+=`	`x += 5;`	`x = x + 5;`
`-=`	`y -= 2;`	`y = y - 2;`
`*=`	`z *= 10;`	`z = z * 10;`
`/=`	`a /= b;`	`a = a / b;`
`%=`	`c %= 3;`	`c = c % 3;`

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.7

Formatting Output

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Formatting Output

- Can control how output displays for numeric, string data:
  - size
  - position
  - number of digits
- Requires `iomanip` header file

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

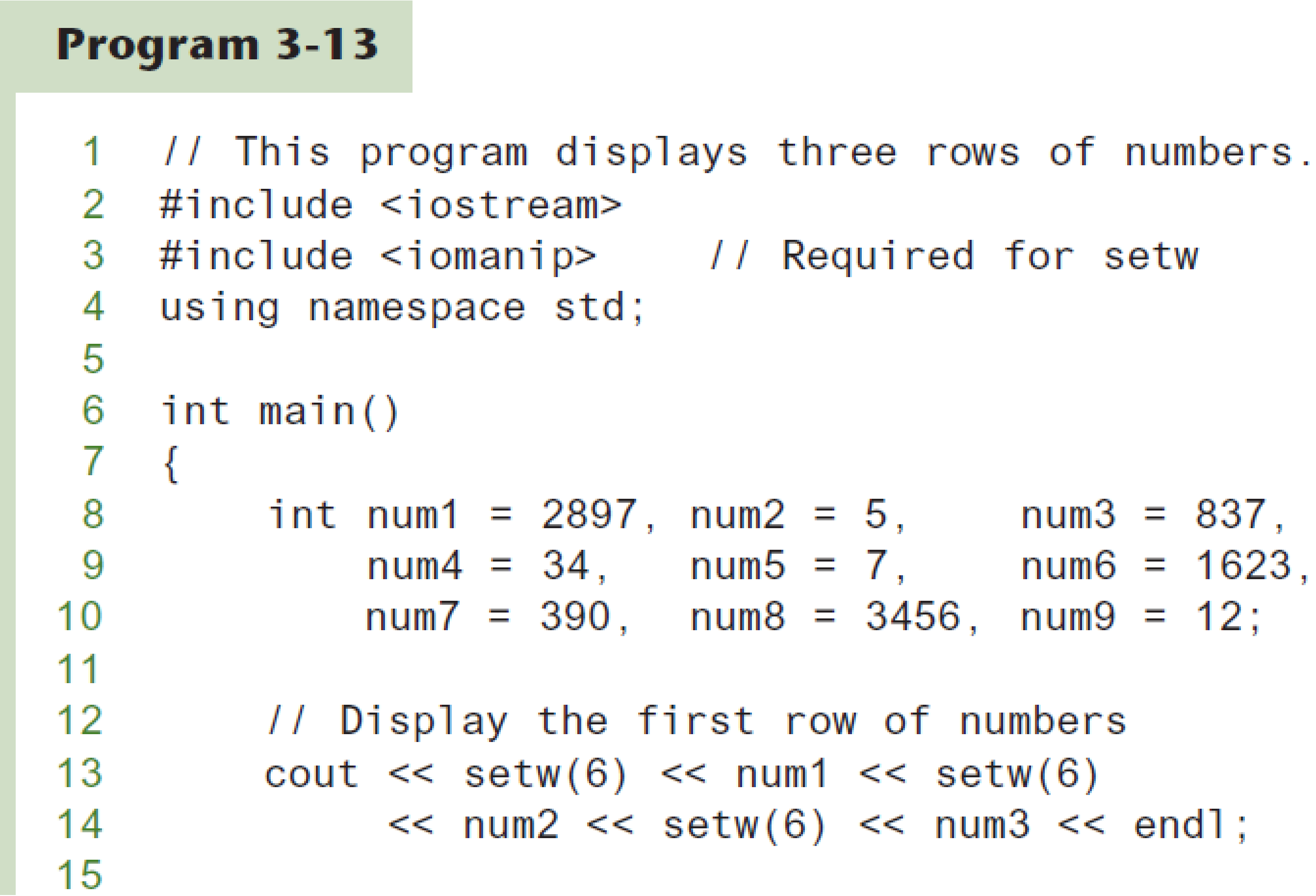
Stream Manipulators (1 of 3)

- Used to control how an output field is displayed
- Some affect just the next value displayed:
  - `setw(x)`: print in a field at least `x` spaces wide. Use more spaces if field is not wide enough

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

The `setw` Stream Manipulator

(1 of 2)



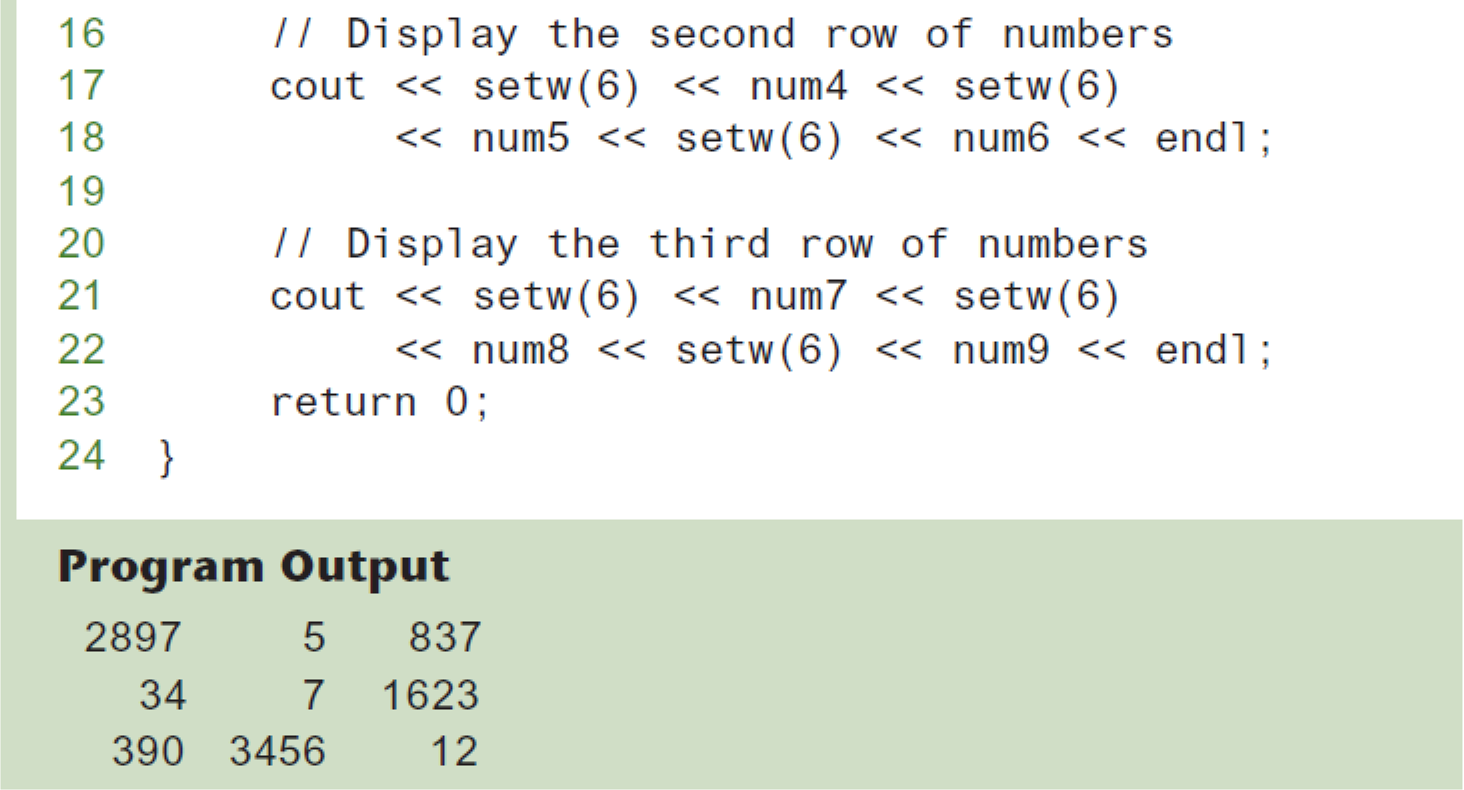
Program 3-13

```
// This program displays three rows of numbers.
#include <iostream>
#include <iomanip>    // Required for setw
using namespace std;

int main()
{
    int num1 = 2897, num2 = 5,    num3 = 837,
        num4 = 34,   num5 = 7,    num6 = 1623,
        num7 = 390,  num8 = 3456, num9 = 12;

    // Display the first row of numbers
    cout << setw(6) << num1 << setw(6)
         << num2 << setw(6) << num3 << endl;

```

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

The **`setw`** Stream Manipulator

(2 of 2)



```
    // Display the second row of numbers
    cout << setw(6) << num4 << setw(6)
         << num5 << setw(6) << num6 << endl;

    // Display the third row of numbers
    cout << setw(6) << num7 << setw(6)
         << num8 << setw(6) << num9 << endl;
    return 0;
}
```

Program Output

```
  2897     5   837
    34     7  1623
   390  3456    12
```

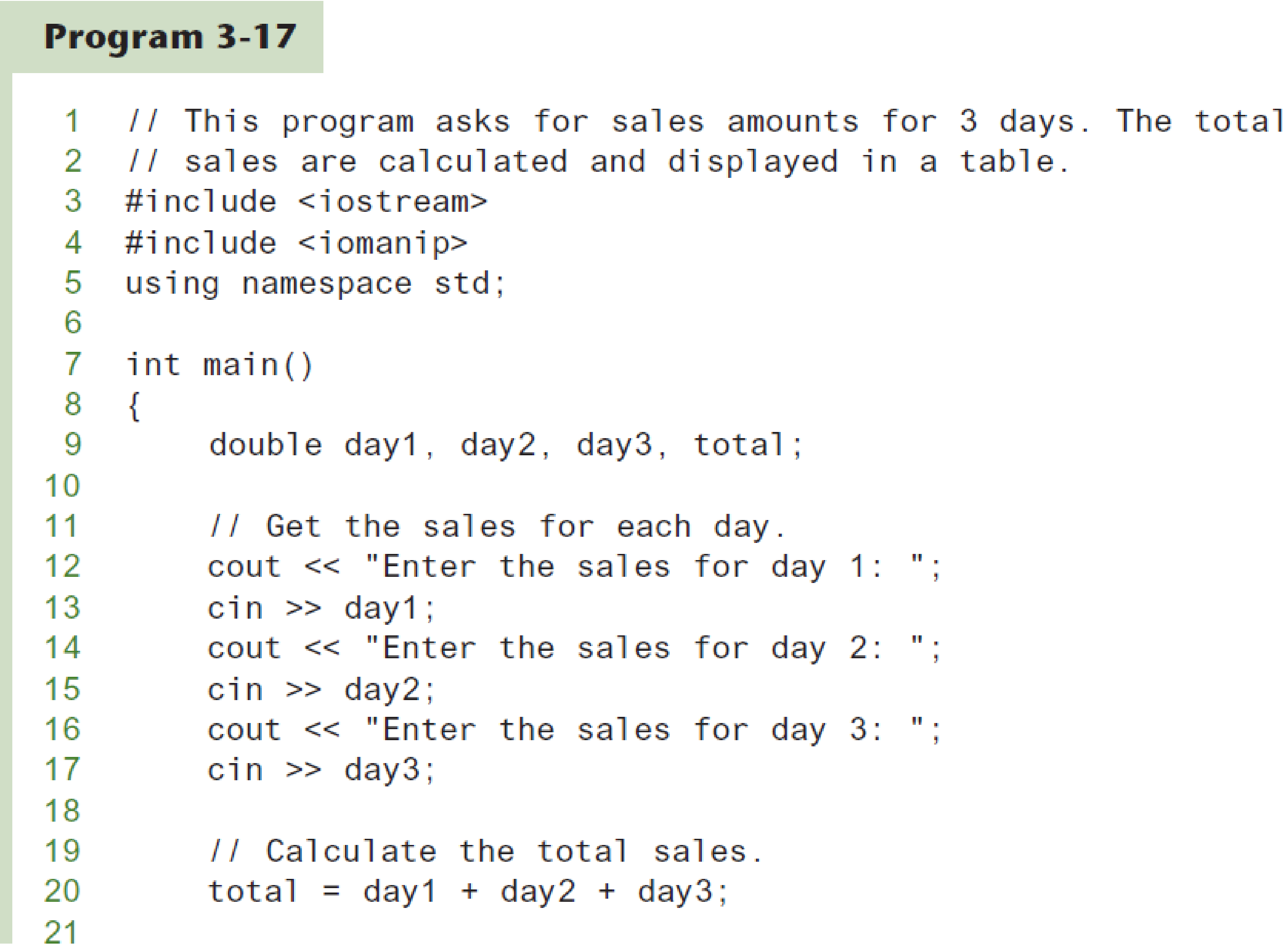
Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Stream Manipulators (2 of 3)

- Some affect values until changed again:
  - `fixed`: use decimal notation for floating-point values
  - `setprecision(x)`: when used with `fixed`, print floating-point value using `x` digits after the decimal. Without `fixed`, print floating-point value using x significant digits
  - `showpoint`: always print decimal for floating-point values

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

More Stream Manipulators (1 of 2)

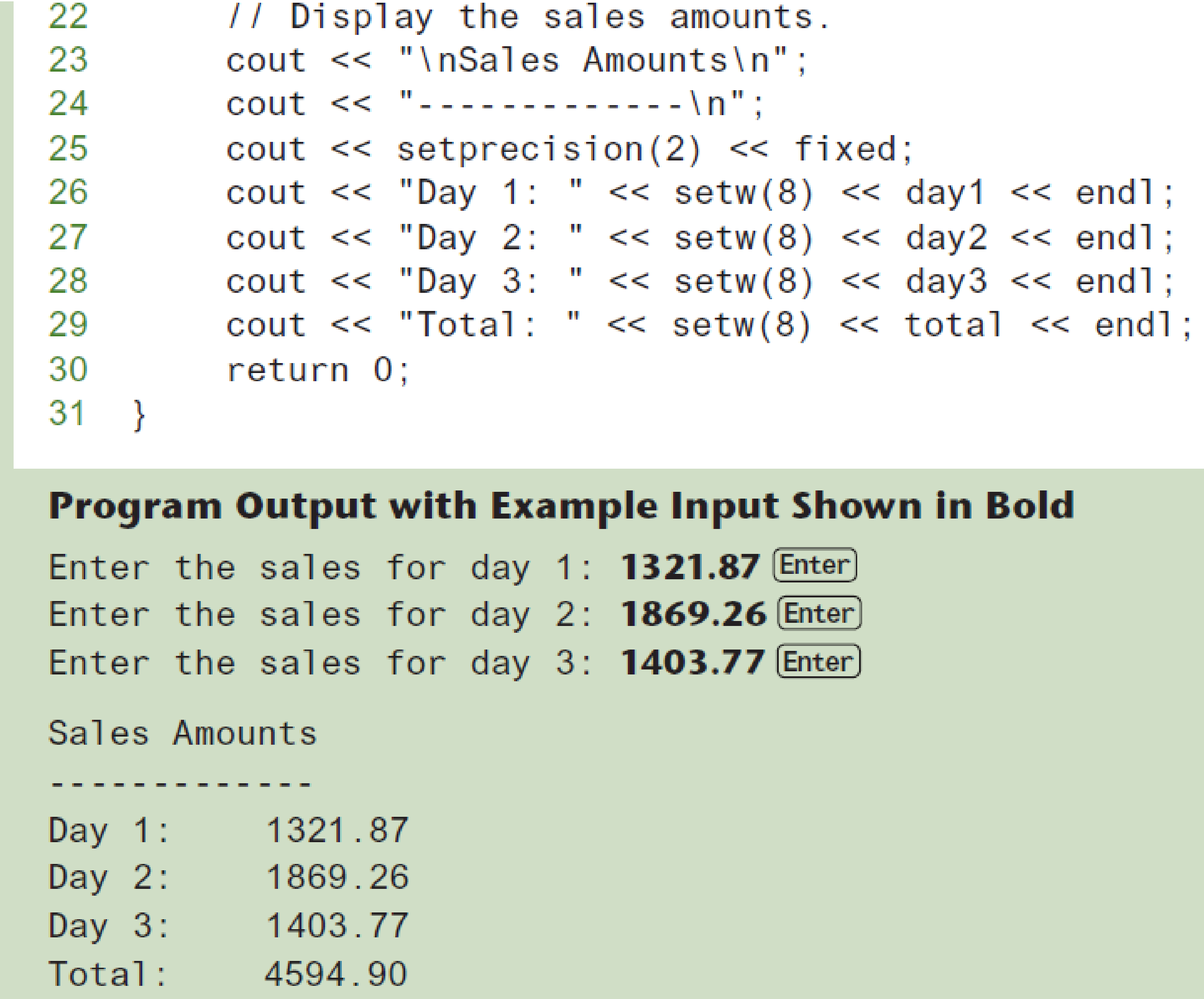


Program 3-17

```cpp
// This program asks for sales amounts for 3 days. The total
// sales are calculated and displayed in a table.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    double day1, day2, day3, total;

    // Get the sales for each day.
    cout << "Enter the sales for day 1: ";
    cin >> day1;
    cout << "Enter the sales for day 2: ";
    cin >> day2;
    cout << "Enter the sales for day 3: ";
    cin >> day3;

    // Calculate the total sales.
    total = day1 + day2 + day3;

```

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

More Stream Manipulators (2 of 2)



```
    // Display the sales amounts.
    cout << "\nSales Amounts\n";
    cout << "-------------\n";
    cout << setprecision(2) << fixed;
    cout << "Day 1: " << setw(8) << day1 << endl;
    cout << "Day 2: " << setw(8) << day2 << endl;
    cout << "Day 3: " << setw(8) << day3 << endl;
    cout << "Total: " << setw(8) << total << endl;
    return 0;
}
```

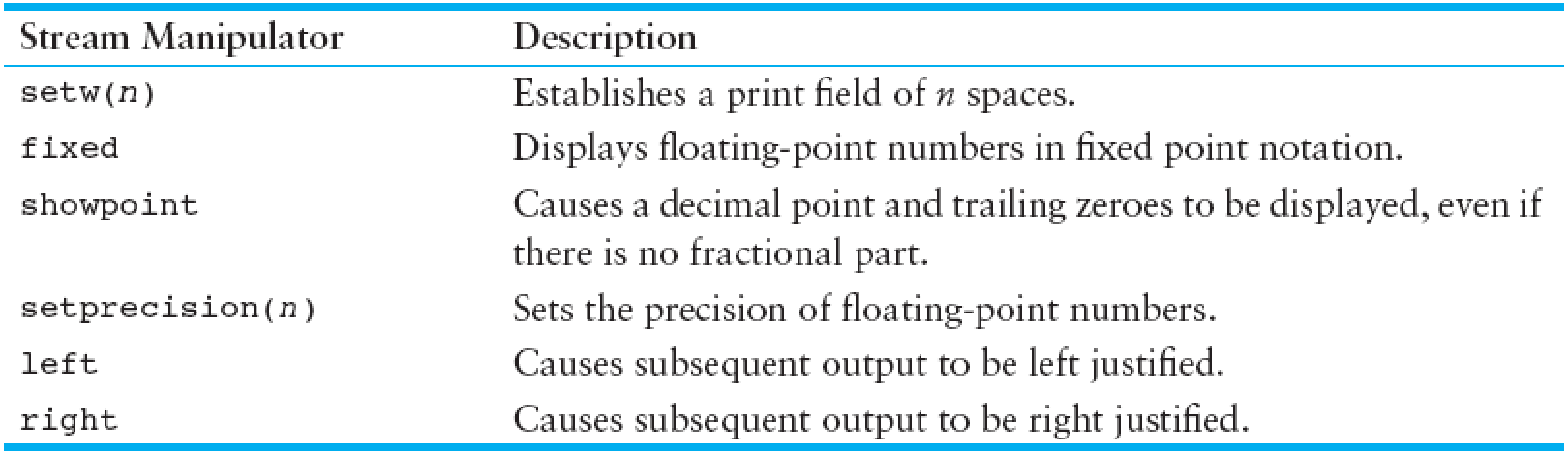
Program Output with Example Input Shown in Bold

```
Enter the sales for day 1: 1321.87 Enter
Enter the sales for day 2: 1869.26 Enter
Enter the sales for day 3: 1403.77 Enter

Sales Amounts
-------------
Day 1:     1321.87
Day 2:     1869.26
Day 3:     1403.77
Total:     4594.90
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Stream Manipulators (3 of 3)

Table 3-12



Stream Manipulator	Description
`setw(`*n*`)`	Establishes a print field of *n* spaces.
`fixed`	Displays floating-point numbers in fixed point notation.
`showpoint`	Causes a decimal point and trailing zeroes to be displayed, even if there is no fractional part.
`setprecision(`*n*`)`	Sets the precision of floating-point numbers.
`left`	Causes subsequent output to be left justified.
`right`	Causes subsequent output to be right justified.

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.8

Working with Characters and `string` Objects

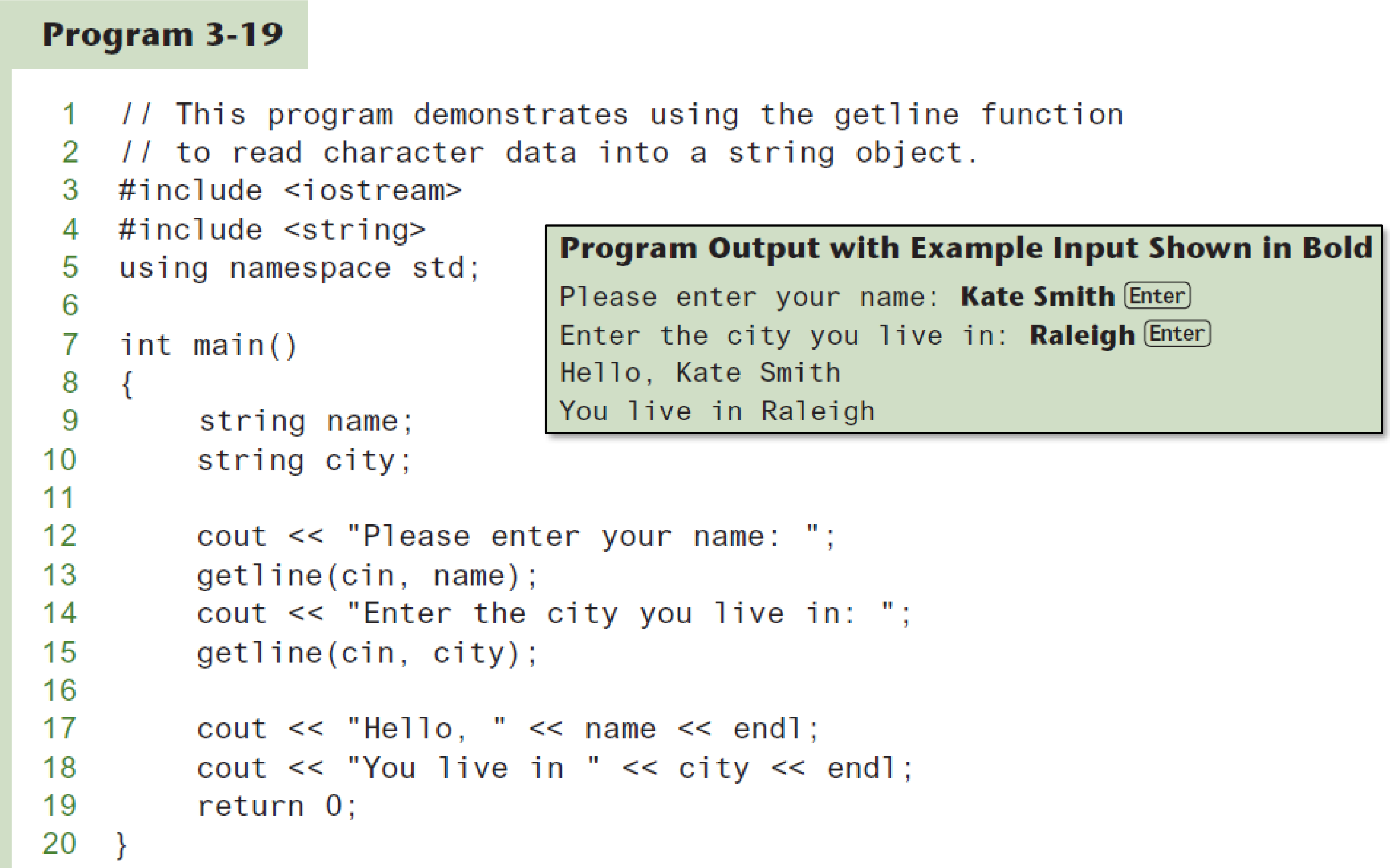
Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Working with Characters and `string` Objects (1 of 3)

- Using `cin` with the >> operator to input strings can cause problems:
- It passes over and ignores any leading *whitespace characters (spaces, tabs, or line breaks)*
- To work around this problem, you can use a C++ function named `getline`.

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Using getline in Program 3-19



Program 3-19

```
// This program demonstrates using the getline function
// to read character data into a string object.
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string name;
    string city;

    cout << "Please enter your name: ";
    getline(cin, name);
    cout << "Enter the city you live in: ";
    getline(cin, city);

    cout << "Hello, " << name << endl;
    cout << "You live in " << city << endl;
    return 0;
}
```

Program Output with Example Input Shown in Bold

```
Please enter your name: Kate Smith Enter
Enter the city you live in: Raleigh Enter
Hello, Kate Smith
You live in Raleigh
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Working with Characters and `string` Objects (2 of 3)

- To read a single character:
  - Use `cin`:

    ```
    char ch;
    cout << "Strike any key to continue";
    cin >> ch;
    ```

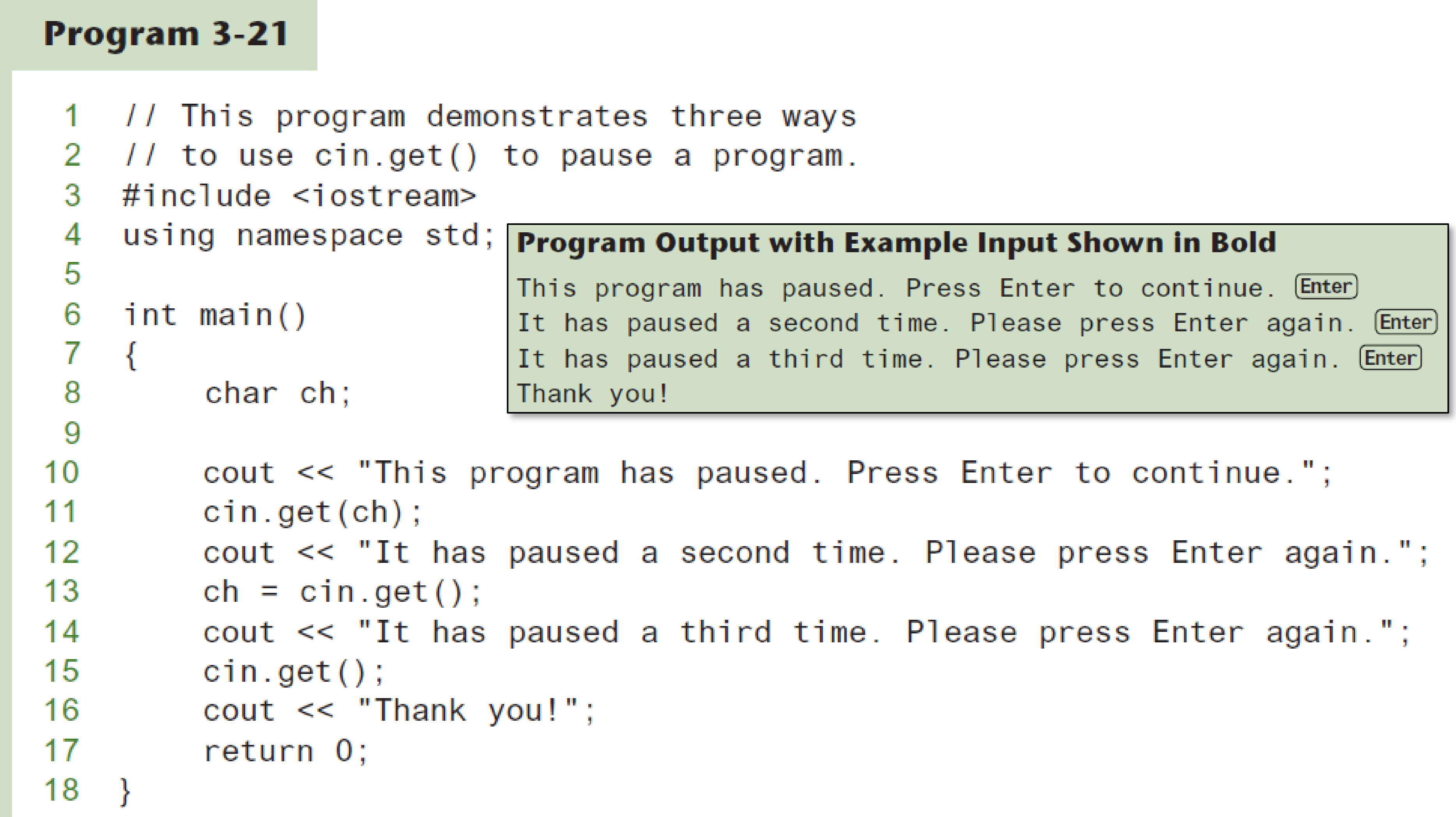
    Problem: will skip over blanks, tabs, `<CR>`
  - Use `cin.get()`:

    ```
    cin.get(ch);
    ```

    Will read the next character entered, even whitespace

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Using `cin.get()`



Program 3-21

```
// This program demonstrates three ways
// to use cin.get() to pause a program.
#include <iostream>
using namespace std;

int main()
{
    char ch;

    cout << "This program has paused. Press Enter to continue.";
    cin.get(ch);
    cout << "It has paused a second time. Please press Enter again.";
    ch = cin.get();
    cout << "It has paused a third time. Please press Enter again.";
    cin.get();
    cout << "Thank you!";
    return 0;
}
```

Program Output with Example Input Shown in Bold

```
This program has paused. Press Enter to continue. Enter
It has paused a second time. Please press Enter again. Enter
It has paused a third time. Please press Enter again. Enter
Thank you!
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Working with Characters and `string` Objects (3 of 3)

- Mixing `cin >>` and `cin.get()` in the same program can cause input errors that are hard to detect
- To skip over unneeded characters that are still in the keyboard buffer, use `cin.ignore()`:

```
// skip next char
cin.ignore();

// skip the next 10 chars or until '\n'
cin.ignore(10, '\n');
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

`string` Member Functions and Operators

- To find the length of a string:

```
string state = "Texas";
int size = state.length();
```

- To concatenate (join) multiple strings:

```
greeting2 = greeting1 + name1;
greeting1 = greeting1 + name2;
```

Or using the **+=** combined assignment operator:

```
greeting1 += name2;
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.9

More Mathematical Library Functions

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

More Mathematical Library Functions (1 of 2)

- Require `cmath` header file
- Take `double` as input, return a `double`
- Commonly used functions:

`sin`	Sine
`cos`	Cosine
`tan`	Tangent
`sqrt`	Square root
`log`	Natural (e) log
`abs`	Absolute value (takes and returns an int)

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

More Mathematical Library Functions (2 of 2)

- These require `cstdlib` header file
- `rand()`: returns a random number (`int`) between `0` and the largest int the compute holds. Yields same sequence of numbers each time program is run.
- `srand(x)`: initializes random number generator with `unsigned int x`

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Random Numbers

Random numbers are useful in many applications, such as

- Games and simulations
- Statistical analysis
- Data encryption

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Generating Random Numbers

(1 of 5)

- Use this `#include` statement:

```
#include <random>
```

- Create the following objects:
  - A random number engine to generate a random sequence of bits
  - A distribution object to format the bits into numbers of a specific data type, within a specified range

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Generating Random Numbers

(2 of 5)

- Example: Generate a random integer in the range 0-100:

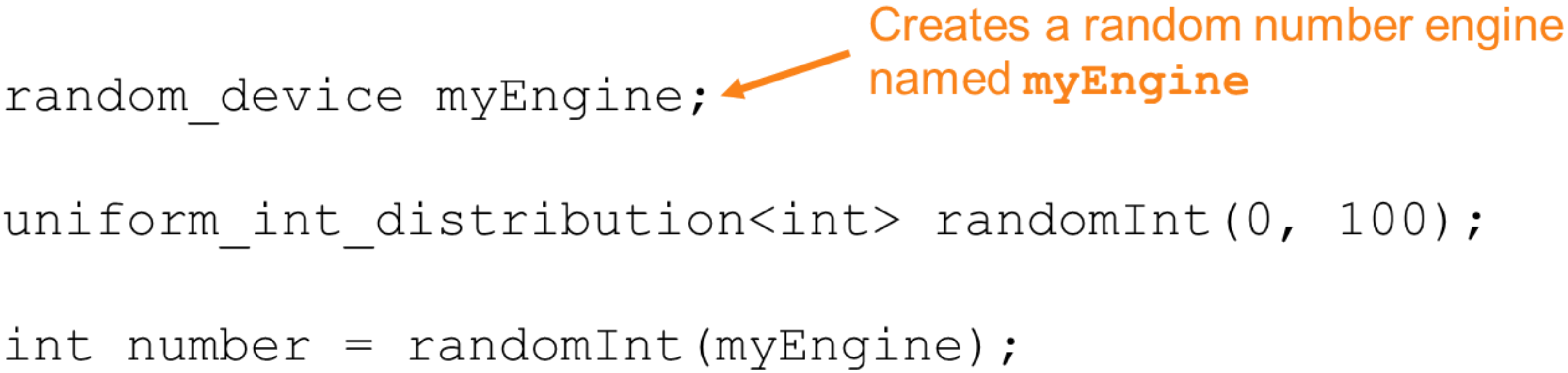
```
random_device myEngine;

uniform_int_distribution<int> randomInt(0, 100);

int number = randomInt(myEngine);
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Generating Random Numbers

(3 of 5)

- Example: Generate a random integer in the range 0-100:



```
random_device myEngine;

uniform_int_distribution<int> randomInt(0, 100);

int number = randomInt(myEngine);
```

Creates a random number engine named **`myEngine`**

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Generating Random Numbers

(4 of 5)

- Example: Generate a random integer in the range 0-100:

Creates a distribution object named **randomInt**

```
random_device myEngine;

uniform_int_distribution<int> randomInt(0, 100);

int number = randomInt(myEngine);
```

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Generating Random Numbers

(5 of 5)

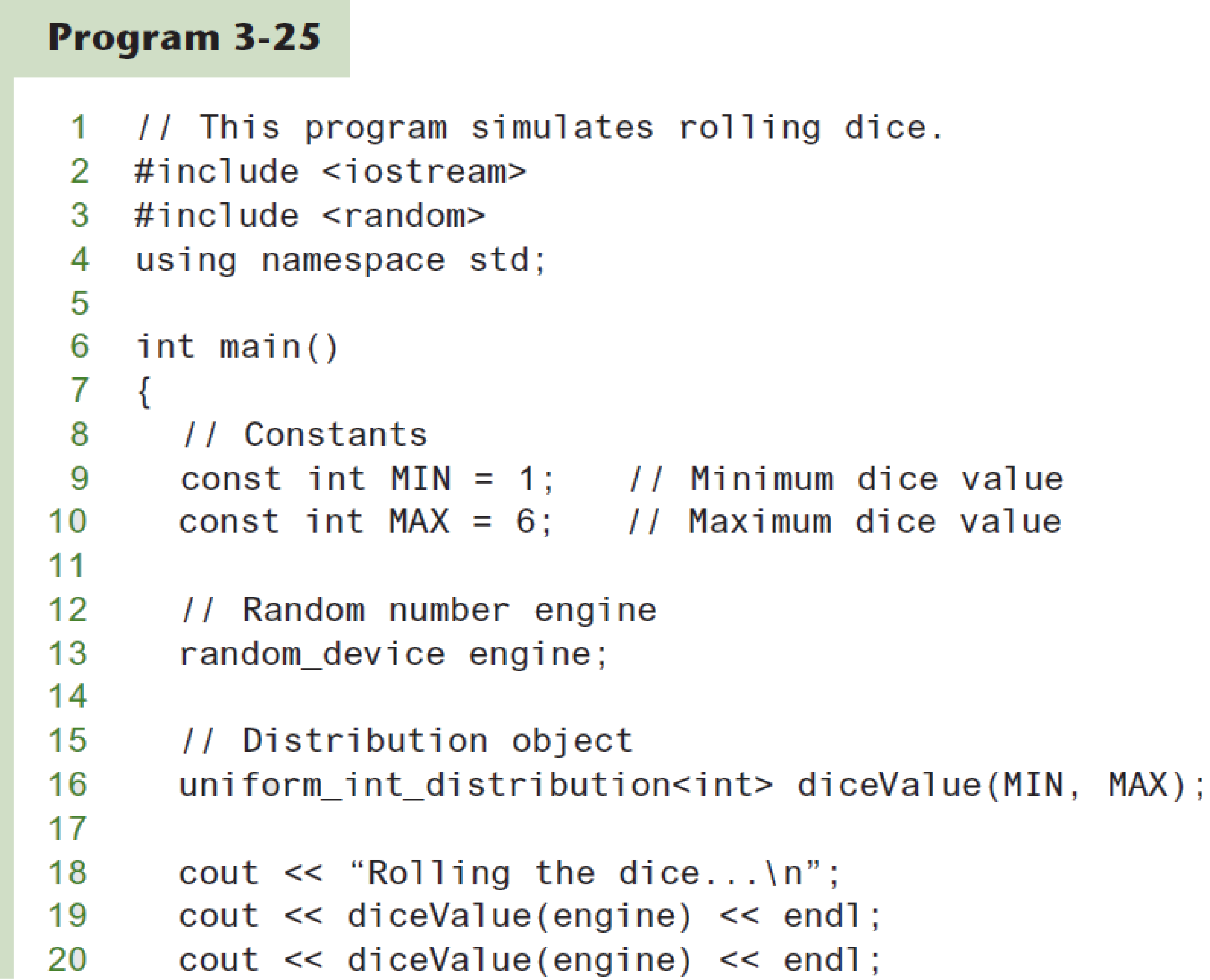
- Example: Generate a random integer in the range 0-100:

```
random_device myEngine;

uniform_int_distribution<int> randomInt(0, 100);

int number = randomInt(myEngine);
```

Generates a random **`int`** in the range 0-100 and assigns it to **`number`**

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Simulating Dice (1 of 2)

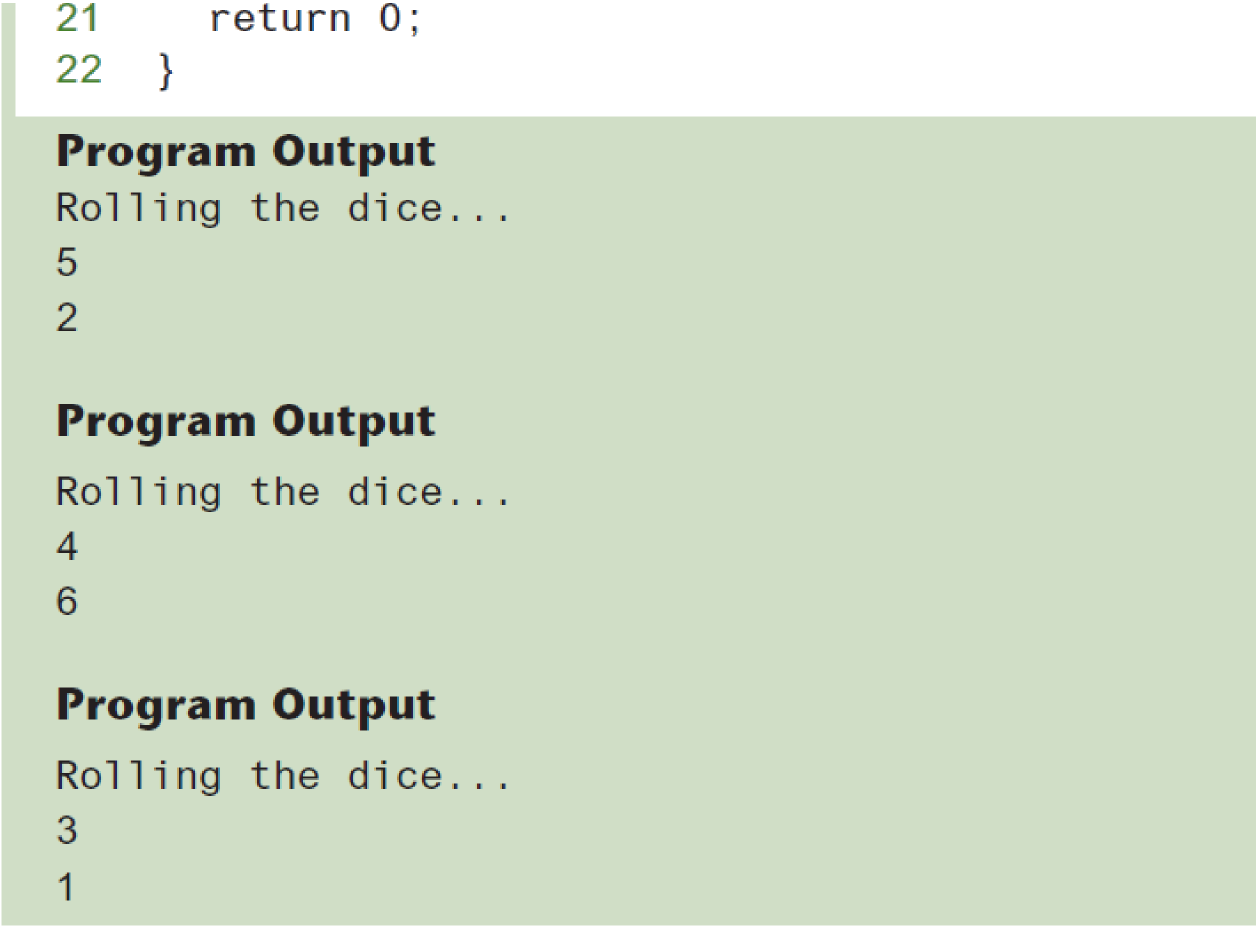


Program 3-25

```
// This program simulates rolling dice.
#include <iostream>
#include <random>
using namespace std;

int main()
{
   // Constants
   const int MIN = 1;    // Minimum dice value
   const int MAX = 6;    // Maximum dice value

   // Random number engine
   random_device engine;

   // Distribution object
   uniform_int_distribution<int> diceValue(MIN, MAX);

   cout << “Rolling the dice...\n”;
   cout << diceValue(engine) << endl;
   cout << diceValue(engine) << endl;
```

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Simulating Dice (2 of 2)



```
    return 0;
}
```

Program Output

```
Rolling the dice...
5
2
```

Program Output

```
Rolling the dice...
4
6
```

Program Output

```
Rolling the dice...
3
1
```

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.10

Hand Tracing a Program

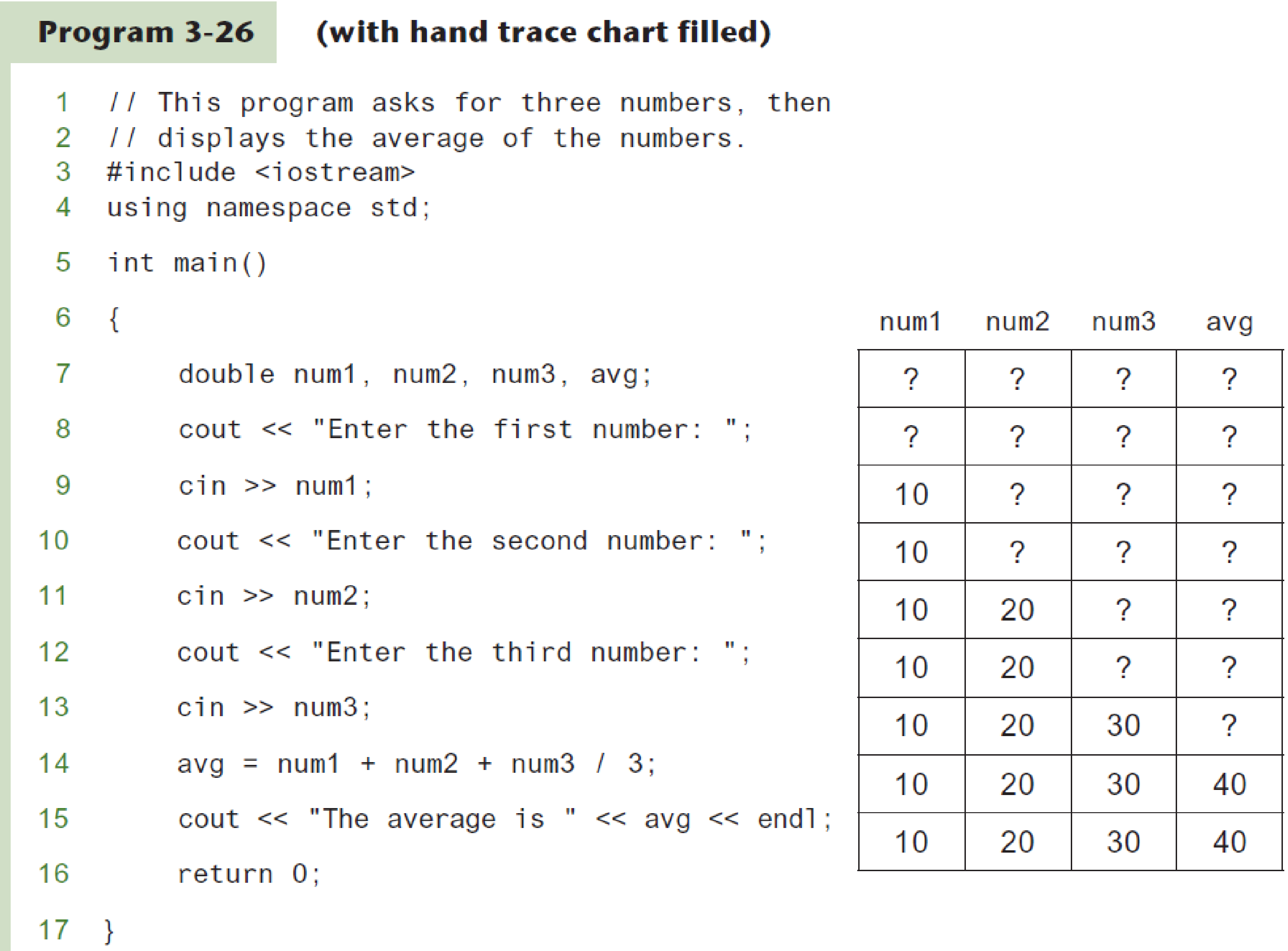
Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Hand Tracing a Program

- Hand trace a program: act as if you are the computer, executing a program:
  - step through and ‘execute’ each statement, one-by-one
  - record the contents of variables after statement execution, using a hand trace chart (table)
- Useful to locate logic or mathematical errors

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Program 3-26 with Hand Trace Chart



Program 3-26 **(with hand trace chart filled)**

```
 1  // This program asks for three numbers, then
 2  // displays the average of the numbers.
 3  #include <iostream>
 4  using namespace std;

 5  int main()

 6  {

 7      double num1, num2, num3, avg;

 8      cout << "Enter the first number: ";

 9      cin >> num1;

10      cout << "Enter the second number: ";

11      cin >> num2;

12      cout << "Enter the third number: ";

13      cin >> num3;

14      avg = num1 + num2 + num3 / 3;

15      cout << "The average is " << avg << endl;

16      return 0;

17  }
```

num1	num2	num3	avg
?	?	?	?
?	?	?	?
10	?	?	?
10	?	?	?
10	20	?	?
10	20	?	?
10	20	30	?
10	20	30	40
10	20	30	40

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

3.11

A Case Study

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

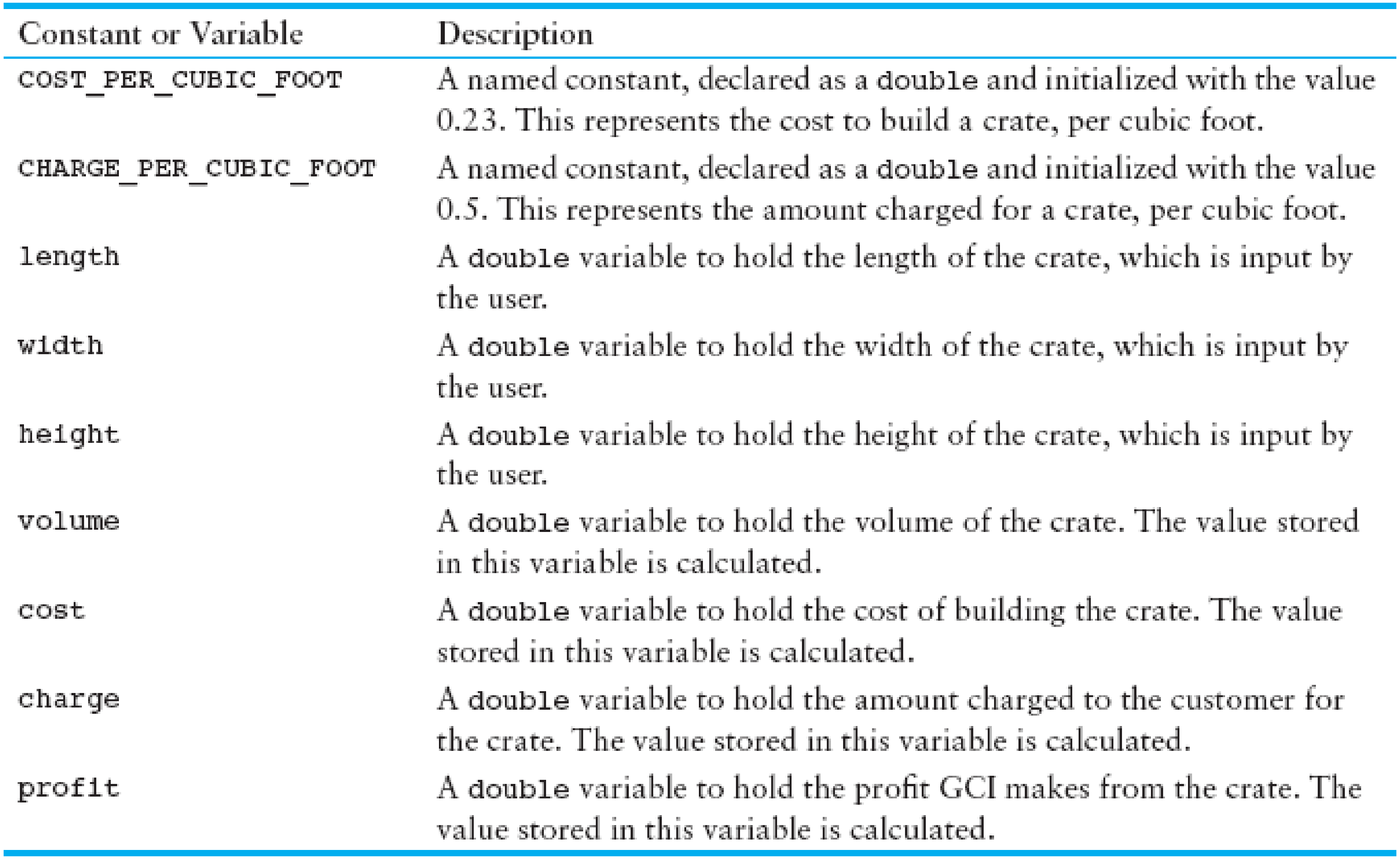
A Case Study

- General Crates, Inc. builds custom-designed wooden crates.
- You have been asked to write a program that calculates the:
  - Volume (in cubic feet)
  - Cost
  - Customer price
  - Profit of any crate GCI builds

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Variables

Table 3-14



Constant or Variable	Description
`COST_PER_CUBIC_FOOT`	A named constant, declared as a `double` and initialized with the value 0.23. This represents the cost to build a crate, per cubic foot.
`CHARGE_PER_CUBIC_FOOT`	A named constant, declared as a `double` and initialized with the value 0.5. This represents the amount charged for a crate, per cubic foot.
`length`	A `double` variable to hold the length of the crate, which is input by the user.
`width`	A `double` variable to hold the width of the crate, which is input by the user.
`height`	A `double` variable to hold the height of the crate, which is input by the user.
`volume`	A `double` variable to hold the volume of the crate. The value stored in this variable is calculated.
`cost`	A `double` variable to hold the cost of building the crate. The value stored in this variable is calculated.
`charge`	A `double` variable to hold the amount charged to the customer for the crate. The value stored in this variable is calculated.
`profit`	A `double` variable to hold the profit GCI makes from the crate. The value stored in this variable is calculated.

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Program Design

The program must perform the following general steps:

Step 1:

Ask the user to enter the dimensions of the crate

Step 2:

Calculate:

- the crate's volume
- the cost of building the crate
- the customer's charge
- the profit made

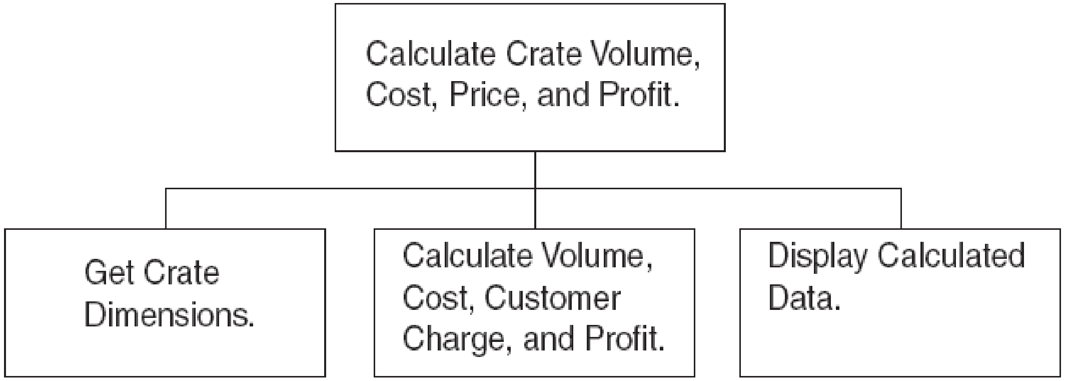
Step 3:

Display the data calculated in Step 2.

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

General Hierarchy Chart

Figure 3-7



Calculate Crate Volume, Cost, Price, and Profit.

Get Crate Dimensions.

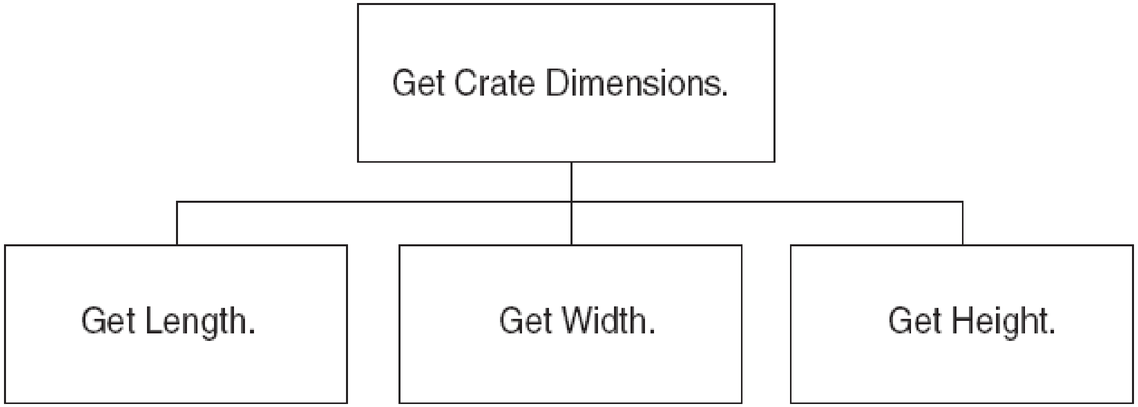
Calculate Volume, Cost, Customer Charge, and Profit.

Display Calculated Data.

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Get Crate Dimensions

Figure 3-8



Get Crate Dimensions.

Get Length.

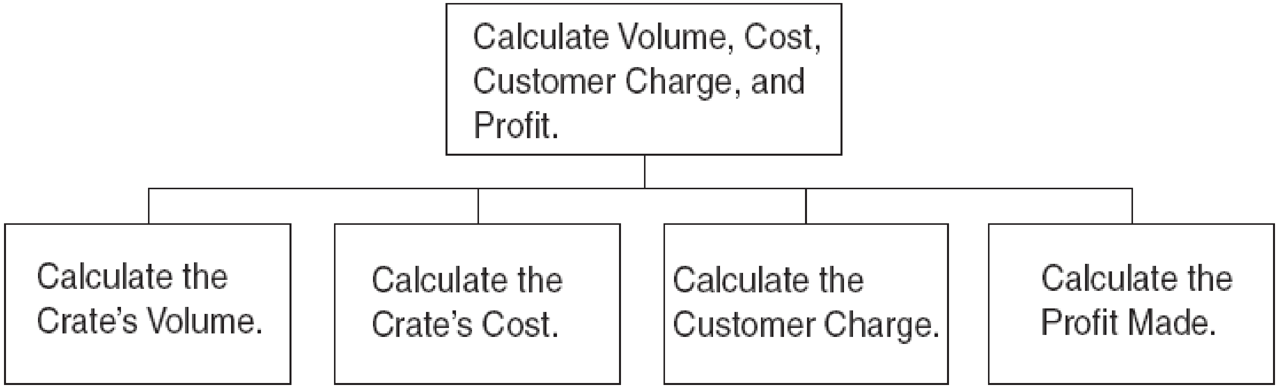
Get Width.

Get Height.

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Calculate Volume, Cost, Customer Charge, and Profit

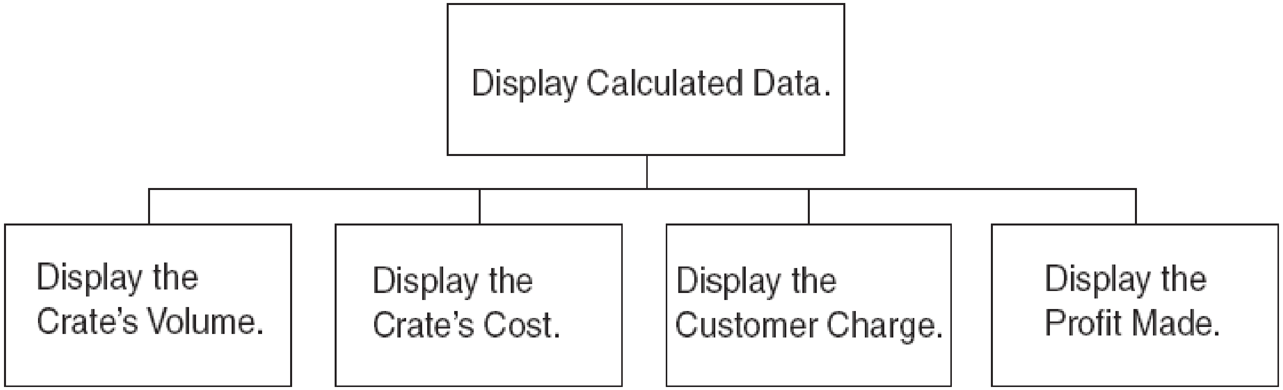
Figure 3-9


Calculate Volume, Cost, Customer Charge, and Profit.
Calculate the Crate's Volume.
Calculate the Crate's Cost.
Calculate the Customer Charge.
Calculate the Profit Made.

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Display Calculated Data

Figure 3-10



Display Calculated Data.

Display the Crate's Volume.

Display the Crate's Cost.

Display the Customer Charge.

Display the Profit Made.

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Psuedocode

```
Ask the user to input the crate's length.
Ask the user to input the crate's width.
Ask the user to input the crate's height.
Calculate the crate's volume.
Calculate the cost of building the crate.
Calculate the customer's charge for the crate.
Calculate the profit made from the crate.
Display the crate's volume.
Display the cost of building the crate.
Display the customer's charge for the crate.
Display the profit made from the crate.
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

Calculations

The following formulas will be used to calculate the crate’s volume, cost, charge, and profit:

$$\text{volume} = \text{length} \times \text{width} \times \text{height}$$

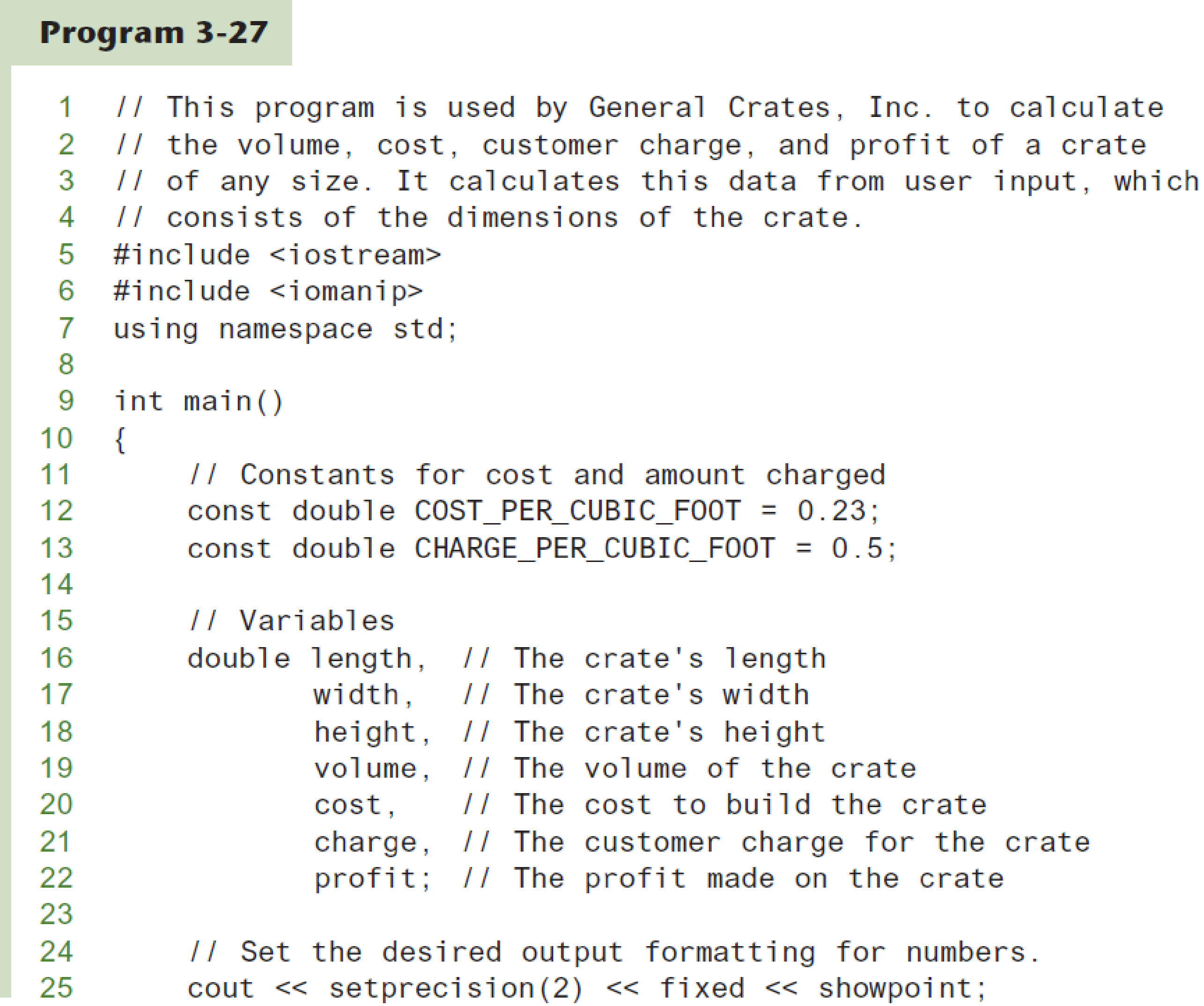
$$\text{cost} = \text{volume} \times 0.23$$

$$\text{charge} = \text{volume} \times 0.5$$

$$\text{profit} = \text{charge} - \text{cost}$$

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

The Program (1 of 3)

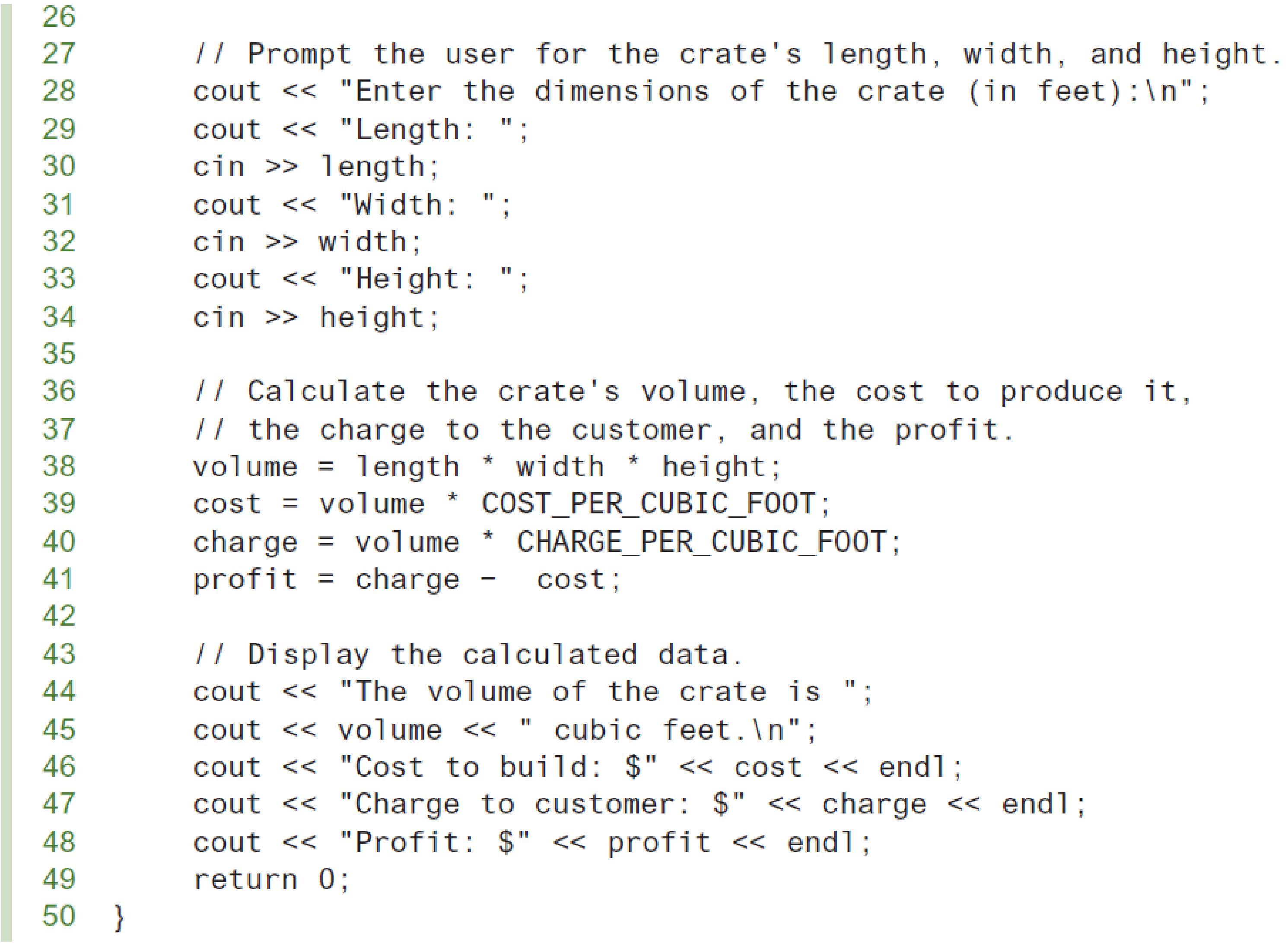


Program 3-27

```
// This program is used by General Crates, Inc. to calculate
// the volume, cost, customer charge, and profit of a crate
// of any size. It calculates this data from user input, which
// consists of the dimensions of the crate.
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    // Constants for cost and amount charged
    const double COST_PER_CUBIC_FOOT = 0.23;
    const double CHARGE_PER_CUBIC_FOOT = 0.5;

    // Variables
    double length,  // The crate's length
           width,   // The crate's width
           height,  // The crate's height
           volume,  // The volume of the crate
           cost,    // The cost to build the crate
           charge,  // The customer charge for the crate
           profit;  // The profit made on the crate

    // Set the desired output formatting for numbers.
    cout << setprecision(2) << fixed << showpoint;
```

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

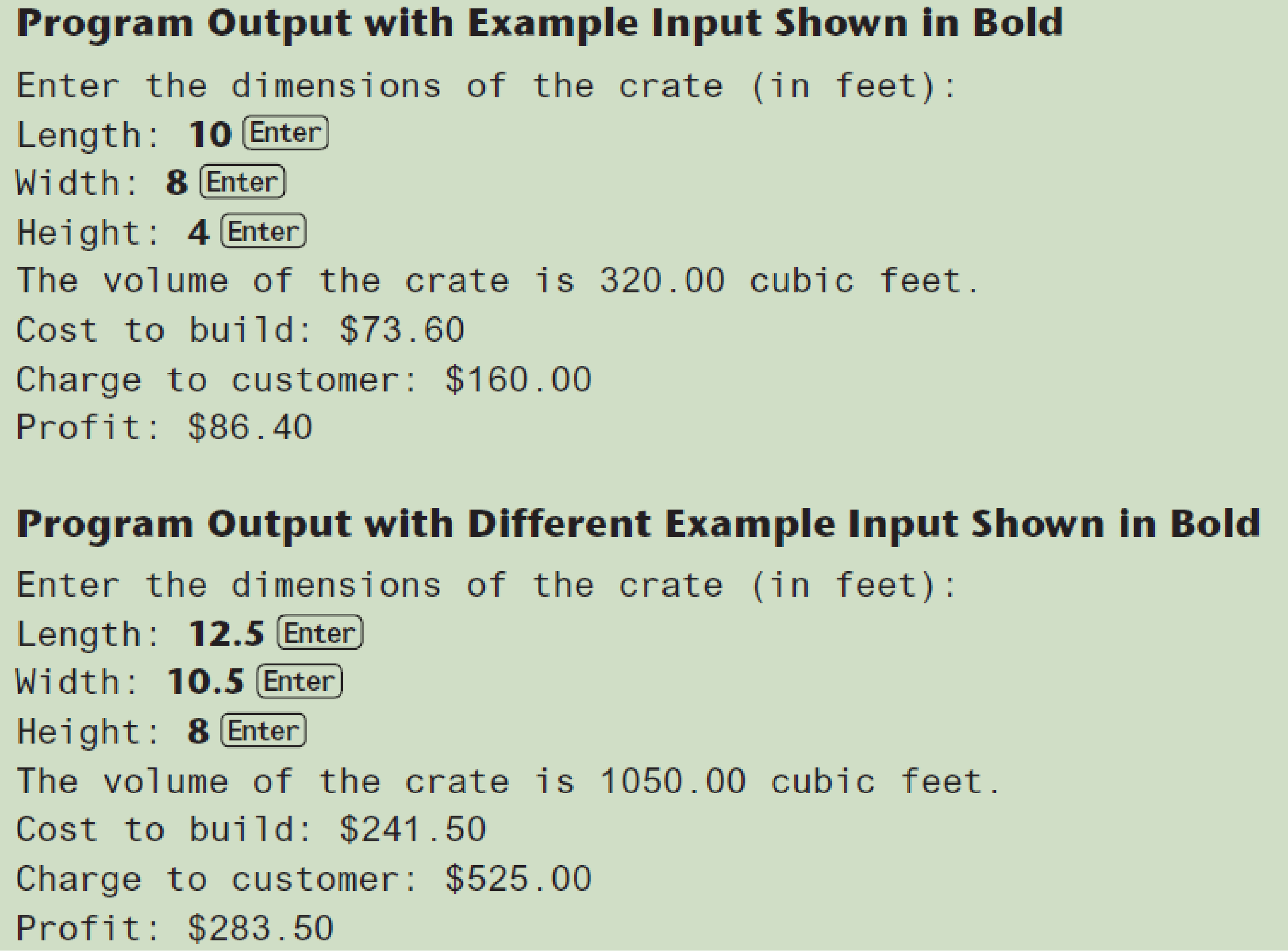
The Program (2 of 3)



```

    // Prompt the user for the crate's length, width, and height.
    cout << "Enter the dimensions of the crate (in feet):\n";
    cout << "Length: ";
    cin >> length;
    cout << "Width: ";
    cin >> width;
    cout << "Height: ";
    cin >> height;

    // Calculate the crate's volume, the cost to produce it,
    // the charge to the customer, and the profit.
    volume = length * width * height;
    cost = volume * COST_PER_CUBIC_FOOT;
    charge = volume * CHARGE_PER_CUBIC_FOOT;
    profit = charge -  cost;

    // Display the calculated data.
    cout << "The volume of the crate is ";
    cout << volume << " cubic feet.\n";
    cout << "Cost to build: $" << cost << endl;
    cout << "Charge to customer: $" << charge << endl;
    cout << "Profit: $" << profit << endl;
    return 0;
}
```

Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.

The Program (3 of 3)



Program Output with Example Input Shown in Bold

```
Enter the dimensions of the crate (in feet):
Length: 10 Enter
Width: 8 Enter
Height: 4 Enter
The volume of the crate is 320.00 cubic feet.
Cost to build: $73.60
Charge to customer: $160.00
Profit: $86.40
```

Program Output with Different Example Input Shown in Bold

```
Enter the dimensions of the crate (in feet):
Length: 12.5 Enter
Width: 10.5 Enter
Height: 8 Enter
The volume of the crate is 1050.00 cubic feet.
Cost to build: $241.50
Charge to customer: $525.00
Profit: $283.50
```

Pearson Copyright © 2021, 2018, 2015, 2012, 2009 Pearson Education, Inc. All rights reserved.