Chapter 3: Expressions and Interactivity

3.1 The cin Object

The cin Object
- 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

The cin Object in Program 3-1

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

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.

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

- Can be used to input more than one value:
  ```cpp```
  `cin >> height >> width;`
  ```cpp```
- Multiple values from keyboard must be separated by spaces
- Order is important: first value entered goes to first variable, etc.

The `cin` Object Gathers Multiple Values in Program 3-2

```cpp```
```cpp```
```cpp```
```cpp```
```cpp```
```cpp```

The `cin` Object Reads Different Data Types in Program 3-3

```cpp```

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:
  ```cpp```
  ```cpp```

Order of Operations

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

1. Evaluate second
2. Evaluate first
3. Evaluate third
Order of Operations

### Table 3.2 Some Simple Expressions and Their Values

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 + 2 × 4</td>
<td>13</td>
</tr>
<tr>
<td>10 / 2 + 3</td>
<td>2</td>
</tr>
<tr>
<td>8 + 12 × 2 - 4</td>
<td>24</td>
</tr>
<tr>
<td>4 + 17 % 2 - 1</td>
<td>4</td>
</tr>
<tr>
<td>6 - 3 × 2 + 7 ÷ 1</td>
<td>6</td>
</tr>
</tbody>
</table>

Associativity of Operators

- (unary negation) associates right to left
- *, /, %, +, - associate right to left
- parentheses () can be used to override the order of operations:
  - \(2 + 2 ∗ 2 - 2 = 4\)
  - \((2 + 2) ∗ 2 - 2 = 6\)
  - \(2 + 2 ∗ (2 - 2) = 2\)
  - \((2 + 2) ∗ (2 - 2) = 0\)

Grouping with Parentheses

### Table 3.4 More Simple Expressions and Their Values

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5 + 2) × 4</td>
<td>28</td>
</tr>
<tr>
<td>10 / (5 - 3)</td>
<td>5</td>
</tr>
<tr>
<td>8 + 12 × (6 - 2)</td>
<td>56</td>
</tr>
<tr>
<td>(4 + 17) % 2 - 1</td>
<td>0</td>
</tr>
<tr>
<td>(6 - 3) × (2 + 7) / 3</td>
<td>9</td>
</tr>
</tbody>
</table>

Algebraic Expressions

- 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 \(m = (y_2-y_1) / (x_2-x_1)\);

Algebraic Expressions

### Table 3.5 Algebraic and C++ Multiplication Expressions

<table>
<thead>
<tr>
<th>Algebraic Expression</th>
<th>Operation</th>
<th>C++ Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>6l</td>
<td>6 * l</td>
<td>6 * l</td>
</tr>
<tr>
<td>(3)(12)</td>
<td>3 * 12</td>
<td>3 * 12</td>
</tr>
<tr>
<td>dxy</td>
<td>d * x * y</td>
<td>d * x * y</td>
</tr>
</tbody>
</table>

3.3

When You Mix Apples with Oranges: Type Conversion
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.

Hierarchy of Types

Highest:
- long double
- double
- float
- unsigned long
- long
- unsigned int
- int

Lowest:
- Ranked by largest number they can hold

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

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

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
3.5 Type Casting

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);

3.6 Multiple Assignment and Combined Assignment

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));
Combined Assignment

Look at the following statement:

\[ \text{sum} = \text{sum} + 1; \]

This adds 1 to the variable \text{sum}.

Other Similar Statements

Table 3-8 (Assume \( x = 0 \))

<table>
<thead>
<tr>
<th>Statement</th>
<th>What It Does</th>
<th>Value of ( x ) After the Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = x + 1 )</td>
<td>Adds 1 to ( x )</td>
<td>10</td>
</tr>
<tr>
<td>( x = x - 3 )</td>
<td>Subtracts 3 from ( x )</td>
<td>3</td>
</tr>
<tr>
<td>( x = x \times 10 )</td>
<td>Multiplies ( x ) by 10</td>
<td>60</td>
</tr>
<tr>
<td>( x = x / 2 )</td>
<td>Divides ( x ) by 2</td>
<td>3</td>
</tr>
<tr>
<td>( x = x \mod 4 )</td>
<td>Makes ( x ) the remainder of ( x / 4 )</td>
<td>2</td>
</tr>
</tbody>
</table>

Combined Assignment Operators

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

The statement

\[ \text{sum} = \text{sum} + 1; \]

is equivalent to

\[ \text{sum} += 1; \]

Formatting Output

Can control how output displays for numeric, string data:

- size
- position
- number of digits

Requires \text{iomanip} header file
Stream Manipulators

- 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

The `setw` Stream Manipulator in Program 3-13

Program 3-13

```
1 // This program displays three rows of numbers.
2 #include <iostream>
3 #include <iomanip>
4 // Required for setw
5 using namespace std;
6
7 int main()
8 {
9   int num1 = 1234, num2 = 5678, num3 = 9012;
10   // Display the first row of numbers
11   cout << setw(10) << num1 << ' ' << num2 << ' ' << num3 << endl;
12   // Display the second row of numbers
13   cout << setw(10) << num1 << ' ' << num2 << ' ' << num3 << endl;
14   // Display the third row of numbers
15   cout << setw(10) << num1 << ' ' << num2 << ' ' << num3 << endl;
16   return 0;
```

Stream Manipulators

- 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

More Stream Manipulators in Program 3-17

```
1 // This program asks for sales figures for 3 days. The total
2 // sales are calculated and displayed in a table.
3 #include <iostream>
4 #include <iomanip>
5 using namespace std;
6
7 int main()
8 {
9   double day1, day2, day3, total;
10   // Get the sales for each day.
11   cout << "Enter the sales for day 1: " ;
12   cin >> day1;
13   cout << "Enter the sales for day 2: " ;
14   cin >> day2;
15   cout << "Enter the sales for day 3: " ;
16   cin >> day3;
17   // Calculate the total sales.
18   total = day1 + day2 + day3;
19   // Display the sales figures.
20   cout << setw(10) << "Day" << setw(10) << "Sales" << endl;
21   cout << setw(10) << "1" << setw(10) << 1234.56 << endl;
22   cout << setw(10) << "2" << setw(10) << 7890.12 << endl;
23   cout << setw(10) << "3" << setw(10) << 4567.89 << endl;
24   cout << setw(10) << "Total" << setw(10) << total << endl;
25   return 0;
```

More Stream Manipulators in Program 3-17

```
21 // Display the sales figures.
22 cout << setw(10) << "Day" << setw(10) << "Sales" << endl;
23 cout << setw(10) << "1" << setw(10) << 1234.56 << endl;
24 cout << setw(10) << "2" << setw(10) << 7890.12 << endl;
25 cout << setw(10) << "3" << setw(10) << 4567.89 << endl;
26 cout << setw(10) << "Total" << setw(10) << total << endl;
27 return 0;
}
```

Program Output with Example Input Shown in Bold

```
Day 1: 1234.56 [Enter]
Day 2: 7890.12 [Enter]
Day 3: 4567.89 [Enter]
```

Sales Figures

```
Day 1: 1234.56
Day 2: 7890.12
Day 3: 4567.89
Total: 24802.57
```
Stream Manipulators

Table 3.12

<table>
<thead>
<tr>
<th>Stream Manipulator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>setw()</td>
<td>Establishes a print field of a specific width.</td>
</tr>
<tr>
<td>fill()</td>
<td>Displays floating-point numbers in fixed-point notation.</td>
</tr>
<tr>
<td>setprecision()</td>
<td>Sets the precision of floating-point numbers.</td>
</tr>
<tr>
<td>left()</td>
<td>Causes subsequent output to be left justified.</td>
</tr>
<tr>
<td>right()</td>
<td>Causes subsequent output to be right justified.</td>
</tr>
</tbody>
</table>

3.8 Working with Characters and string Objects

Using \texttt{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 \texttt{getline}.

Working with Characters and string Objects

- To read a single character:
  - Use \texttt{cin}:
    ```cpp
cchar ch;
cout << "Strike any key to continue": cin >> ch;
```
  - Problem: will skip over blanks, tabs, <CR>
  - Use \texttt{cin.get()}:
    ```cpp
cin.get(ch);
```
  - Will read the next character entered, even whitespace

Using \texttt{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>
#include <cassert>

int main()
{
    std::string name;
    std::cout << "Enter the name you like (Hit Enter): " << std::endl;
    getline(std::cin, name);
    std::cout << "You like " << name << std::endl;
    return 0;
}
```

Program Output with Example Input Shown in Bold

Please enter your name: Kate Smith [Enter]
You like Kate Smith

Using \texttt{cin.get()} in Program 3-21

```
Program 3-21

// This program demonstrates three ways
// To see cin.get() to pause a program.
#include <iostream>
#include <cassert>

int main()
{
    std::cout << "Enter any character: " << std::endl;
    std::cin.get();
    return 0;
}
```

Program Output with Example Input Shown in Bold

Please enter any character: B [Enter]
Working with Characters and string Objects

- 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():
  ```
  cin.ignore(); // skip next char
  cin.ignore(10, '\n'); // skip the next
  // 10 char. or until a '\n'
  ```

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;
  ```

More Mathematical Library Functions

- 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)
  ```

More Mathematical Library Functions

- These requirecstdlib 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

3.9

3.10

Hand Tracing a Program
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

Program 3-27 with Hand Trace Chart

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

Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crate_vol_cost</td>
<td>A named constant, declared as a double and initialized with the value 0.25. This represents the cost to build a crate, per cubic foot.</td>
</tr>
<tr>
<td>crate_vol_cost</td>
<td>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.</td>
</tr>
<tr>
<td>length</td>
<td>A double variable to hold the length of the crate, which is input by the user.</td>
</tr>
<tr>
<td>width</td>
<td>A double variable to hold the width of the crate, which is input by the user.</td>
</tr>
<tr>
<td>height</td>
<td>A double variable to hold the height of the crate, which is input by the user.</td>
</tr>
<tr>
<td>volume</td>
<td>A double variable to hold the volume of the crate. The value stored in this variable is calculated.</td>
</tr>
<tr>
<td>cost</td>
<td>A double variable to hold the cost of building the crate. The value stored in this variable is calculated.</td>
</tr>
<tr>
<td>charge</td>
<td>A double variable to hold the amount charged to the customer for the crate. The value stored in this variable is calculated.</td>
</tr>
<tr>
<td>profit</td>
<td>A double variable to hold the profit GCI makes from the crate. The value stored in this variable is calculated.</td>
</tr>
</tbody>
</table>

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.
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.

Calculations

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

- \( \text{volume} = \text{length} \times \text{width} \times \text{height} \)
- \( \text{cost} = \text{volume} \times 0.23 \)
- \( \text{charge} = \text{volume} \times 0.5 \)
- \( \text{profit} = \text{charge} - \text{cost} \)
The Program

```java
// Program 3-20
1 // This program is used by George's crane, Inc. to calculate
2 // the volume, cost, customer charge, and profit of a crate
3 // of any size. It calculates this data from user input, which
4 // consists of the dimensions of the crate.
5
6 include <iostream>
7 include <iomanip>
8
9 double length;
10 double width;
11 double height;
12 double volume;
13 double cost;
14 double customer charges;
15 double profit;
16
17 // Function declarations
18 void getCraneData();
19 void calculate();
20 void displayPrice();
21
22 int main()
23 {
24     getCraneData();
25     calculate();
26     displayPrice();
27     return 0;
28 }
29
30 // Function definitions
31 void getCraneData()
32 {
33     cout << "Enter the dimensions of the crate (in feet): " << flush;
34     cin >> length >> width >> height;
35     volume = length * width * height;
36     cout << "The volume of the crate is: " << volume << " cubic feet."
37     cout << "Cost to build: " << cost << endl;
38     cout << "Charge to customer: " << customerCharges << endl;
39     profit = (volume - cost) - customerCharges;
40     cout << "Profit: " << profit << endl;
41 }
42
43 void calculate()
44 {
45     cost = volume * COST_PER_CUBIC_FEET;
46     customerCharges = volume * CREDIT_CARGO_FOOT;
47     profit = cost - customerCharges;
48 }
49
50 void displayPrice()
51 {
52     cout << "The Program continued..." << endl;
53     cout << "Enter the dimensions of the crate (in feet): " << flush;
54     cin >> length >> width >> height;
55     volume = length * width * height;
56     cost = volume * COST_PER_CUBIC_FEET;
57     customerCharges = volume * CREDIT_CARGO_FOOT;
58     profit = cost - customerCharges;
59     cout << "Cost to build: " << cost << endl;
60     cout << "Charge to customer: " << customerCharges << endl;
61     cout << "Profit: " << profit << endl;
62 }
```

The Program

```java
27 // Prompt the user for the crate's length, width, and height
28 cost = 0.00; // The volume of the crate (in feet^3)
29 cost = 0.00; // Cost to build: $0.00
30 customerCharges = 0.00; // Charge to customer: $0.00
31 profit = 0.00; // Profit: $0.00
32
33 // Calculate the crate's volume, the cost to produce it, and the profit.
34 volume = length * width * height;
35 cost = volume * COST_PER_CUBIC_FEET;
36 customerCharges = volume * CREDIT_CARGO_FOOT;
37 profit = cost - customerCharges;
38
39 // Display the calculated data.
40 cout << "The volume of the crate is: " << volume << " cubic feet."
41 cost = 0.00; // Cost to build: $0.00
42 customerCharges = 0.00; // Charge to customer: $0.00
43 profit = 0.00; // Profit: $0.00
44 return 0;
45 }
46
47 // Function definitions
48 void getCraneData()
49 {
50     cout << "Enter the dimensions of the crate (in feet): " << flush;
51     cin >> length >> width >> height;
52     volume = length * width * height;
53     cost = volume * COST_PER_CUBIC_FEET;
54     customerCharges = volume * CREDIT_CARGO_FOOT;
55     profit = cost - customerCharges;
56     cout << "Cost to build: " << cost << endl;
57     cout << "Charge to customer: " << customerCharges << endl;
58     cout << "Profit: " << profit << endl;
59 }
60
61 void calculate()
62 {
63     cost = 0.00; // Cost to build: $0.00
64     customerCharges = 0.00; // Charge to customer: $0.00
65     profit = 0.00; // Profit: $0.00
66 }
67
68 void displayPrice()
69 {
70     cout << "The Program continued..." << endl;
71     cout << "Enter the dimensions of the crate (in feet): " << flush;
72     cin >> length >> width >> height;
73     volume = length * width * height;
74     cost = volume * COST_PER_CUBIC_FEET;
75     customerCharges = volume * CREDIT_CARGO_FOOT;
76     profit = cost - customerCharges;
77     cout << "Cost to build: " << cost << endl;
78     cout << "Charge to customer: " << customerCharges << endl;
79     cout << "Profit: " << profit << endl;
80 }
```