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));
```

value is 5 — value is 5 — value is 5

Combined Assignment

- Look at the following statement:

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

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

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

Combined Assignment

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

- The statement

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

is equivalent to

```
sum += 1;
```

Combined Assignment Operators (1 of 2)

UT DALLAS
The University of Texas at Dallas

- C++ has some combined assignment operators.
- These operators allow the programmer to perform an arithmetic operation and assignment with a single operator.
- Although not required, these operators are popular since they shorten simple equations.

Combined Assignment Operators (2 of 2)

Operator	Example	Equivalent	Value of variable after operation
+=	`x += 5;`	`x = x + 5;`	The old value of **x** plus 5.
-=	`y -= 2;`	`y = y - 2;`	The old value of **y** minus 2
***=**	`z *= 10;`	`z = z * 10;`	The old value of **z** times 10
/=	`a /= b;`	`a = a / b;`	The old value of **a** divided by **b**.
%=	`c %= 3;`	`c = c % 3;`	The remainder of the division of the old value of **c** divided by 3.

- 3.11

3.7

Formatting Output

Formatting Output

- Can control how output displays for numeric, string data:
  - size
  - position
  - number of digits

- Requires `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
- 3.12

The `setw` Stream Manipulator in 3-13

Program 3-13

```
1  // This program displays three rows of numbers.
2  #include <iostream>
3  #include <iomanip>      // Required for setw
4  using namespace std;
5
6  int main()
7  {
8     int num1 = 2897, num2 = 5,    num3 = 837,
9         num4 = 34,   num5 = 7,    num6 = 1623,
10        num7 = 390,  num8 = 3456, num9 = 12;
11
12    // Display the first row of numbers
13    cout << setw(6) << num1 << setw(6)
14         << num2 << setw(6) << num3 << endl;
15
16    // Display the second row of numbers
17    cout << setw(6) << num4 << setw(6)
18         << num5 << setw(6) << num6 << endl;
19
20    // Display the third row of numbers
21    cout << setw(6) << num7 << setw(6)
22         << num8 << setw(6) << num9 << endl;
23    return 0;
24 }
```

Continued…

The `setw` Stream Manipulator in 3-13

Program Output

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

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

- 3.15

More Stream Manipulators in 3-17

Program 3-17

```
1  // This program asks for sales amounts 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
11      // Get the sales for each day.
12      cout << "Enter the sales for day 1: ";
13      cin >> day1;
14      cout << "Enter the sales for day 2: ";
15      cin >> day2;
16      cout << "Enter the sales for day 3: ";
17      cin >> day3;
18
19      // Calculate the total sales.
20      total = day1 + day2 + day3;
21
```

More Stream Manipulators in 3-17

```
22       // Display the sales amounts.
23       cout << "\nSales Amounts\n";
24       cout << "-------------\n";
25       cout << setprecision(2) << fixed;
26       cout << "Day 1: " << setw(8) << day1 << endl;
27       cout << "Day 2: " << setw(8) << day2 << endl;
28       cout << "Day 3: " << setw(8) << day3 << endl;
29       cout << "Total: " << setw(8) << total << endl;
30       return 0;
31   }
```

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
```

Stream Manipulators

Table 3-12 Stream Manipulators

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 zeros 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.

- Precision Demo
- fixedResetFixed.cpp

- Precision.cpp In class ass 5(PrecisionStudent.cpp)
- PrettyPrinting In class ass 6 (PrettyPrintingStudent.cpp)

3.8

Working with Characters and `string` Objects

Working with Characters and string Objects

- 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`.
- 3.18

Using `getline` in Program 3-19

Program 3-19

```
1  // This program demonstrates using the getline function
2  // to read character data into a string object.
3  #include <iostream>
4  #include <string>
5  using namespace std;
6
7  int main()
8  {
9     string name;
10    string city;
11
12    cout << "Please enter your name: ";
13    getline(cin, name);
14    cout << "Enter the city you live in: ";
15    getline(cin, city);
16
17    cout << "Hello, " << name << endl;
18    cout << "You live in " << city << endl;
19    return 0;
20 }
```

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
```

Working with Characters and string Objects

- 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
- 3.20

Using `cin.get()` in Program 3-21

Program 3-21

```
1  // This program demonstrates three ways
2  // to use cin.get() to pause a program.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     char ch;
9
10    cout << "This program has paused. Press Enter to continue.";
11    cin.get(ch);
12    cout << "It has paused a second time. Please press Enter again.";
13    ch = cin.get();
14    cout << "It has paused a third time. Please press Enter again.";
15    cin.get();
16    cout << "Thank you!";
17    return 0;
18 }
```

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!
```

Working with Characters and string Objects

- Mixing `cin >>` and `cin.get()` in the same program can cause input errors that are hard to detect 3.22

- 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'
```

3.23

`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;
```

- stringDemo.cpp

3.9

More Mathematical Library Functions

More Mathematical Library Functions

- Require `cmath` header file 3.24
- 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)

- mathsFunctions.cpp

- Go to Random numbers chapter.

More Mathematical Library Functions

- 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`
- 3.25

Generate Random numbers within a range

- Get one end of the range and store -> end1
- Get the other end and store -> end2
- Range = end2 – end1 + 1
- Seed the random number generator with current time
- Output = end1 + random() % range
- `Rand_test1.cpp`

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

Program 3-27 **(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

3.11

A Case Study

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 3-14 Named Constants and Variables

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.

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

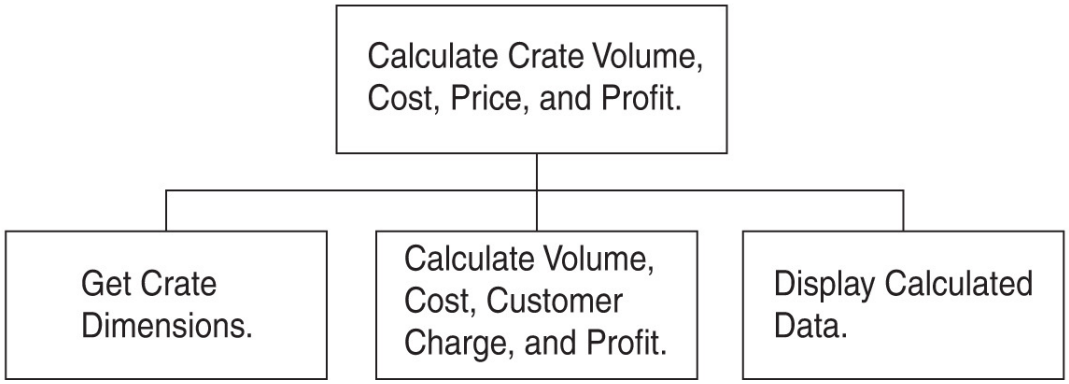
the profit made

Step 3:

Display the data calculated in Step 2.

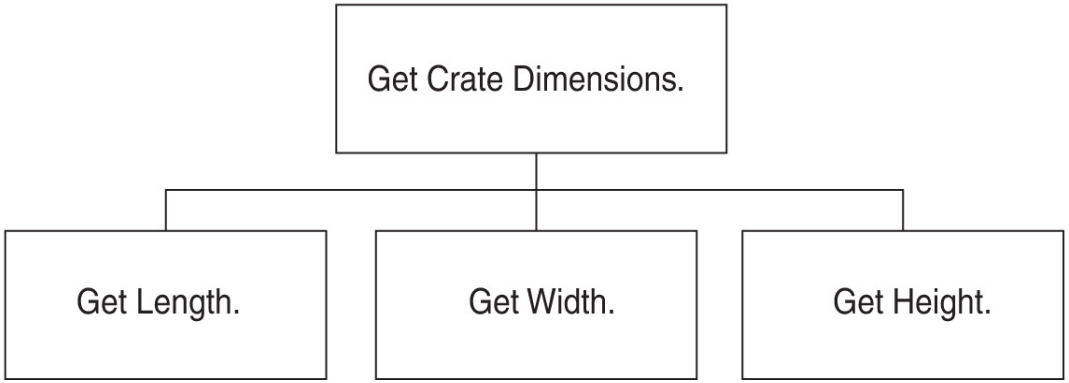
General Hierarchy Chart

Figure 3-7



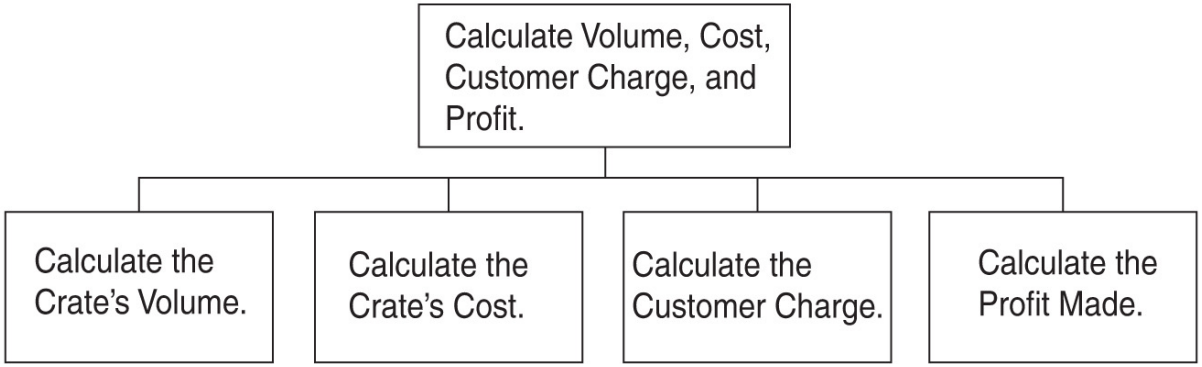
Get Crate Dimensions

Figure 3-8



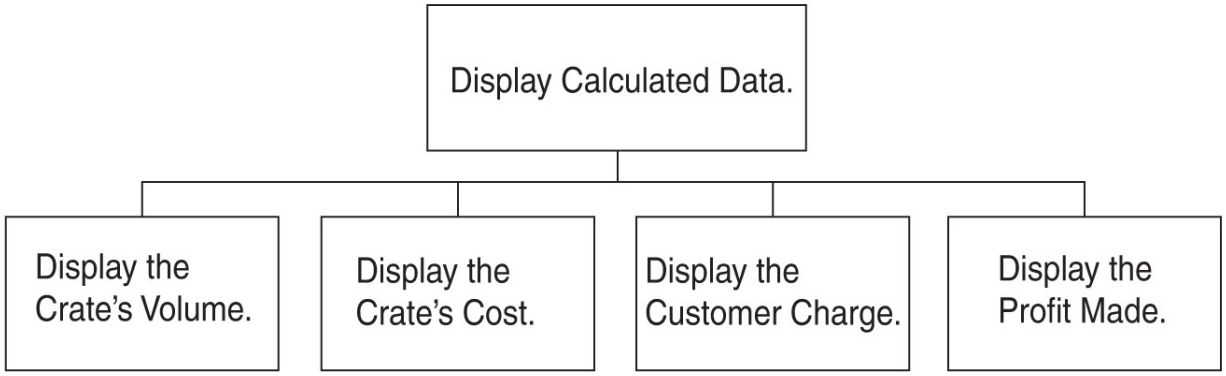
Calculate Volume, Cost, Customer Charge, and Profit

Figure 3-9



Display Calculated Data

Figure 3-10



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

Program 3-28

```
1  // This program is used by General Crates, 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  #include <iostream>
6  #include <iomanip>
7  using namespace std;
8
9  int main()
10 {
11    // Constants for cost and amount charged
12    const double COST_PER_CUBIC_FOOT = 0.23;
13    const double CHARGE_PER_CUBIC_FOOT = 0.5;
14
15    // Variables
16    double length,   // The crate's length
17           width,    // The crate's width
18           height,   // The crate's height
19           volume,   // The volume of the crate
20           cost,     // The cost to build the crate
21           charge,   // The customer charge for the crate
22           profit;   // The profit made on the crate
23
24    // Set the desired output formatting for numbers.
25    cout << setprecision(2) << fixed << showpoint;
26
```

Continued…

The Program

```
27      // Prompt the user for the crate's length, width, and height
28      cout << "Enter the dimensions of the crate (in feet):\n";
29      cout << "Length: ";
30      cin >> length;
31      cout << "Width: ";
32      cin >> width;
33      cout << "Height: ";
34      cin >> height;
35
36      // Calculate the crate's volume, the cost to produce it,
37      // the charge to the customer, and the profit.
38      volume = length * width * height;
39      cost = volume * COST_PER_CUBIC_FOOT;
40      charge = volume * CHARGE_PER_CUBIC_FOOT;
41      profit = charge - cost;
42
43      // Display the calculated data.
44      cout << "The volume of the crate is ";
45      cout << volume << " cubic feet.\n";
46      cout << "Cost to build: $" << cost << endl;
47      cout << "Charge to customer: $" << charge << endl;
48      cout << "Profit: $" << profit << endl;
49      return 0;
50  }
```

Continued…

The Program

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
```