Algorithm Development and Control Statements: Part 1

4

Objectives

In this chapter you'll:

- Learn basic problem-solving techniques.
- Develop algorithms through the process of top-down, stepwise refinement.
- Use the `if` and `if`...`else` selection statements to choose between alternative actions.
- Use the `while` iteration statement to execute statements in a program repeatedly.
- Use counter-controlled iteration and sentinel-controlled iteration.
- Use nested control statements.
- Use the compound assignment operator and the increment and decrement operators.
- Learn about the portability of fundamental data types.

Self-Review Exercises

4.1 Answer each of the following questions.

a) All programs can be written in terms of three types of control statements: ________, ________ and ________.

ANS: Sequence, selection and iteration.

b) The ________selection statement is used to execute one action when a condition is *true* or a different action when that condition is *false*.

ANS: `if...else`.

c) Repeating a set of instructions a specific number of times is called ________ iteration.

ANS: Counter-controlled or definite.

d) When it isn't known in advance how many times a set of statements will be repeated, a(n)________value can be used to terminate the iteration.

ANS: Sentinel, signal, flag or dummy.

4.2 Write four different C++ statements that each add 1 to integer variable `x`.

ANS:
```
x = x + 1;
x += 1;
++x;
x++;
```

4.3 Write C++ statements to accomplish each of the following:

a) In one statement, assign the sum of the current value of `x` and `y` to `z` and postincrement the value of `x`.

ANS: `z = x++ + y;`

b) Determine whether the value of the variable `count` is greater than 10. If it is, print `"Count is greater than 10"`.

ANS:
```
if (count > 10) {
    cout << "Count is greater than 10" << endl;
}
```

c) Predecrement the variable `x` by 1, then subtract it from the variable `total`.

ANS: `total -= --x;`

d) Calculate the remainder after `q` is divided by `divisor` and assign the result to `q`. Write this statement two different ways.

ANS:
```
q %= divisor;
q = q % divisor;
```

4.4 Write C++ statements to accomplish each of the following tasks.

a) Declare variable `sum` to be of type `unsigned int` and initialize it to `0`.

ANS: `unsigned int sum{0};`

b) Declare variable `x` to be of type `unsigned int` and initialize it to `1`.

ANS: `unsigned int x{1};`

c) Add variable `x` to variable `sum` and assign the result to variable `sum`.

ANS: `sum += x;`

or

`sum = sum + x;`

d) Print `"The sum is: "` followed by the value of variable `sum`.

ANS: `cout << "The sum is: " << sum << endl;`

4.5 Combine the statements that you wrote in Exercise 4.4 into a program that calculates and prints the sum of the integers from 1 to 10. Use the `while` statement to loop through the calculation and increment statements. The loop should terminate when the value of `x` becomes 11.

ANS: See the following code:

```
// Exercise 4.5: Calculate.cpp
// Calculate the sum of the integers from 1 to 10
#include <iostream>
using namespace std;

int main() {
   unsigned int sum{0};
   unsigned int x{1};

   while (x <= 10) { // while x is less than or equal to 10
      sum += x; // add x to sum
      ++x; // increment x
   }

   cout << "The sum is: " << sum << endl;
}
```

```
The sum is: 55
```

4.6 State the values of *each* of these `unsigned int` variables after the calculation is performed. Assume that, when each statement begins executing, all variables have the integer value 5.

a) `product *= x++;`

ANS: `product = 25, x = 6;`

b) `quotient /= ++x;`

ANS: `quotient = 0, x = 6;`

4.7 Write single C++ statements or portions of statements that do the following:

a) Input `unsigned int` variable `x` with `cin` and `>>`.

ANS: `cin >> x;`

b) Input `unsigned int` variable `y` with `cin` and `>>`.

ANS: `cin >> y;`

c) Declare `unsigned int` variable `i` and initialize it to `1`.

ANS: `unsigned int i{1};`

d) Declare `unsigned int` variable `power` and initialize it to `1`.

ANS: `unsigned int power{1};`

e) Multiply variable `power` by `x` and assign the result to `power`.

ANS: `power *= x;`

or

`power = power * x;`

f) Preincrement variable `i` by `1`.

ANS: `++i;`

g) Determine whether `i` is less than or equal to `y`.

ANS: `if (i <= y)`

h) Output integer variable `power` with `cout` and `<<`.

ANS: `cout << power << endl;`

4.8 Write a C++ program that uses the statements in Exercise 4.7 to calculate `x` raised to the `y` power. The program should have a `while` iteration statement.

ANS: See the following code:

```
// Exercise 4.8 Solution: power.cpp
// Raise x to the y power.
#include <iostream>
using namespace std;

int main() {
   unsigned int i{1}; // initialize i to begin counting from 1
   unsigned int power{1}; // initialize power

   cout << "Enter base as an integer: ";  // prompt for base
   unsigned int x; // base
   cin >> x; // input base

   cout << "Enter exponent as an integer: "; // prompt for exponent
   unsigned int y; // exponent
   cin >> y; // input exponent

   // count from 1 to y and multiply power by x each time
   while (i <= y) {
      power *= x;
      ++i;
   } // end while

   cout << power << endl; // display result
} // end main
```

```
Enter base as an integer: 2
Enter exponent as an integer: 3
8
```

4.9 Identify and correct the errors in each of the following:

a)
```
while (c <= 5) {
   product *= c;
   ++c;
```

ANS: *Error:* Missing the closing right brace of the `while` body.
Correction: Add closing right brace after the statement `++c;`.

b) `cin << value;`

ANS: *Error:* Used stream insertion instead of stream extraction.
Correction: Change `<<` to `>>`.

c)
```
if (gender == 1) {
   cout << "Woman" << endl;
else; {
   cout << "Man" << endl;
}
```

ANS: *Error:* Semicolon after `else` is a logic error. The second output statement always executes.
Correction: Remove the semicolon after `else`.

4.10 What's wrong with the following `while` iteration statement?

```
while (z >= 0) {
   sum += z;
}
```

ANS: The value of the variable `z` is never changed in the `while` statement. Therefore, if the loop-continuation condition `(z >= 0)` is initially *true*, an infinite loop is created. To prevent the infinite loop, `z` must be decremented so that it eventually becomes less than 0.

Exercises

4.11 *(Correct the Code Errors)* Identify and correct the error(s) in each of the following:

a)
```
if (age >= 65); {
   cout << "Age is greater than or equal to 65" << endl;
}
else {
   cout << "Age is less than 65 << endl";
}
```

ANS: The semicolon at the end of the `if` condition should be removed. The closing double quote after the second `endl` should be placed after `65`.

b)
```
if (age >= 65) {
   cout << "Age is greater than or equal to 65" << endl;
else; {
   cout << "Age is less than 65 << endl";
}
```

ANS: The semicolon after the `else` should be removed. The closing double quote after the second `endl` should be placed after `65`.

c)
```
unsigned int x{1};
unsigned int total;

while (x <= 10) {
   total += x;
   ++x;
}
```

ANS: Variable `total` should be initialized to `0`.

d)
```
While (x <= 100)
   total += x;
   ++x;
```

ANS: The `W` in `while` should be lowercase. The `while`'s body should be enclosed in braces `{}`.

e)
```
while (y > 0) {
   cout << y << endl;
   ++y;
}
```

ANS: The variable `y` should be decremented (i.e., `--y;`), not incremented (`++y;`).

4.12 *(What Does this Program Do?)* What does the following program print?

ANS: The program prints the squares of the integers from 1 to 10 and the sum of those squares. **Instructor Note:** We should have changed line 10 to `int y{x * x};`

```
1  // Exercise 4.12: Mystery.cpp
2  #include <iostream>
3  using namespace std;
```

```

int main() {
   unsigned int x{1};
   unsigned int total{0};

   while (x <= 10) {
      int y = x * x;
      cout << y << endl;
      total += y;
      ++x;
   }

   cout << "Total is " << total << endl;
}
```

For Exercises 4.13–4.16, perform each of these steps:

a) Read the problem statement.
b) Formulate the algorithm using pseudocode and top-down, stepwise refinement.
c) Write a C++ program.
d) Test, debug and execute the C++ program.

4.13 *(Gas Mileage)* Drivers are concerned with the mileage obtained by their automobiles. One driver has kept track of several trips by recording miles driven and gallons used for each trip. Develop a C++ program that uses a `while` statement to input the miles driven and gallons used for each trip. The program should calculate and display the miles per gallon obtained for each trip and print the combined miles per gallon obtained for all tankfuls up to this point.

```
Enter miles driven (-1 to quit): 287
Enter gallons used: 13
MPG this trip: 22.076923
Total MPG: 22.076923

Enter miles driven (-1 to quit): 200
Enter gallons used: 10
MPG this trip: 20.000000
Total MPG: 21.173913

Enter the miles driven (-1 to quit): 120
Enter gallons used: 5
MPG this trip: 24.000000
Total MPG: 21.678571

Enter the miles used (-1 to quit): -1
```

ANS:

Top:

```
    Determine the current and combined miles/gallon for each trip
```

First refinement:

```
    Initialize variables
    For each trip, input the miles driven and gallons used, then calculate and print the miles/gallon for that trip
    Calculate and print the overall average miles/gallon
```

Second refinement:

```
    Initialize totalGallons to zero
```

```
Initialize totalMiles to zero

Prompt the user to enter the miles used for the first trip
Input the miles used for the first trip (possibly the sentinel)

While the sentinel value (-1) has not been entered for the miles
    Prompt the user to enter the gallons used for the current trip
    Input the gallons used for the current trip

    Add miles to the running total in totalMiles
    Add gallons to the running total in totalGallons

    If gallons is not zero
        Calculate and print the miles/gallon

    If totalGallons is not zero
        Calculate and print the totalMiles/totalGallons

    Prompt the user for the next trip's number of miles
    Input the miles used for the next trip
```

4.14 *(Credit Limits)* Develop a C++ program that will determine whether a department-store customer has exceeded the credit limit on a charge account. For each customer, the following facts are available:

a) Account number (an integer)
b) Balance at the beginning of the month
c) Total of all items charged by this customer this month
d) Total of all credits applied to this customer's account this month
e) Allowed credit limit

The program should use a `while` statement to input each of these facts, calculate the new balance (= beginning balance + charges – credits) and determine whether the new balance exceeds the customer's credit limit. For those customers whose credit limit is exceeded, the program should display the customer's account number, credit limit, new balance and the message "Credit Limit Exceeded."

```
Enter account number (or -1 to quit): 100
Enter beginning balance: 5394.78
Enter total charges: 1000.00
Enter total credits: 500.00
Enter credit limit: 5500.00
New balance is 5894.78
Account:      100
Credit limit: 5500.00
Balance:      5894.78
Credit Limit Exceeded.

Enter Account Number (or -1 to quit): 200
Enter beginning balance: 1000.00
Enter total charges: 123.45
Enter total credits: 321.00
Enter credit limit: 1500.00
New balance is 802.45

Enter Account Number (or -1 to quit): -1
```

ANS:
Top:

Determine if each of an arbitrary number of department store customers has exceeded the credit limit on a charge account

First refinement:

```
Input customer's data
For each customer, calculate and display the customer's new balance, and display a
message if user's balance exceeds credit limit
Process next customer
```

Second refinement:

```
Prompt the user for the first customer's account number
Input the first customer's account number

While the sentinel value (-1) has not been entered for the account number
    Prompt the user for the customer's beginning balance
    Input the customer's beginning balance

    Prompt the user for the customer's total charges
    Input the customer's total charges

    Prompt the user for the customer's total credits
    Input the customer's total credits

    Prompt the user for the customer's credit limit
    Input the customer's credit limit

    Calculate and display the customer's new balance

    If the balance exceeds the credit limit
        Print the account number
        Print the credit limit
        Print the balance
        Print "Credit Limit Exceeded"

    Input the next customer's account number
```

4.15 *(Sales-Commission Calculator)* A large company pays its salespeople on a commission basis. The salespeople each receive $200 per week plus 9% of their gross sales for that week. For example, a salesperson who sells $5000 worth of chemicals in a week receives $200 plus 9% of $5000, or a total of $650. Develop a C++ program that uses a `while` statement to input each salesperson's gross sales for last week and calculates and displays that salesperson's earnings. Process one salesperson's figures at a time.

```
Enter sales in dollars (-1 to end): 5000.00
Salary is: $650.00

Enter sales in dollars (-1 to end): 6000.00
Salary is: $740.00

Enter sales in dollars (-1 to end): 7000.00
Salary is: $830.00

Enter sales in dollars (-1 to end): -1
```

ANS:

Top:

```
For an arbitrary number of salespeople, determine each salesperson's earnings for the
previous week
```

First refinement:

```
For each salesperson
    Input the salesperson's sales for the week
    Calculate and print the salesperson's wages for the week
```

Second refinement:

```
Prompt the user for the first salesperson's sales in dollars
Input the first salesperson's sales in dollars

While the sentinel value (-1) has not been entered for the sales
    Calculate the salesperson's wages for the week as 200 dollars added to 9 percent of
        the salesperson's sales (calculated by multiplying the sales with .09)
    Print the salesperson's wages for the week

    Prompt the user for the next salesperson's sales in dollars
    Input the next salesperson's sales in dollars
```

4.16 *(Salary Calculator)* Develop a C++ program that uses a `while` statement to determine the gross pay for each of several employees. The company pays "straight time" for the first 40 hours worked by each employee and pays "time-and-a-half" for all hours worked in excess of 40 hours. You are given a list of the employees of the company, the number of hours each employee worked last week and the hourly rate of each employee. Your program should input this information for each employee and should determine and display the employee's gross pay.

```
Enter hours worked (-1 to end): 39
Enter hourly rate of the employee ($00.00): 10.00
Salary is $390.00

Enter hours worked (-1 to end): 40
Enter hourly rate of the employee ($00.00): 10.00
Salary is $400.00

Enter hours worked (-1 to end): 41
Enter hourly rate of the employee ($00.00): 10.00
Salary is $415.00

Enter hours worked (-1 to end): -1
```

ANS:

Top:

```
Determine the gross pay for an arbitrary number of employees
```

First refinement:

```
For each employee
    Input the hours worked
    Input the hourly rate
    Calculate and display worker's gross pay
```

Second refinement:

```
Prompt the user for the hours worked for the first employee
```

```
Input the hours worked for the first employee

While the sentinel value (-1) has not been entered for the hours
    Prompt the user for the employee's hourly rate
    Input the employee's hourly rate

    If the hours input is less than or equal to 40
        Calculate gross pay by multiplying hours worked by hourly rate
    Else
        Calculate gross pay by multiplying hours worked above forty by 1.5 times
            the hourly rate and adding 40 hours worked by the hourly rate

    Display the employee's gross pay.
    Input the hours worked for the next employee
```

4.21 *(What Does this Program Do?)* What does the following program print?

```cpp
// Exercise 4.21: Mystery2.cpp
#include <iostream>
using namespace std;

int main() {
   unsigned int count{1};

   while (count <= 10) {
      cout << (count % 2 == 1 ? "****" : "++++++++") << endl;
      ++count;
   }
}
```

ANS:

```
****
++++++++
****
++++++++
****
++++++++
****
++++++++
****
++++++++
```

4.22 *(What Does this Program Do?)* What does the following program print?

```cpp
// Exercise 4.22: Mystery3.cpp
#include <iostream>
using namespace std;

int main() {
   unsigned int row{10};

   while (row >= 1) {
      unsigned int column{1};

```

```
         while (column <= 10) {
            cout << (row % 2 == 1 ? "<" : ">");
            ++column;
         }

         --row;
         cout << endl;
      }
   }
```

ANS:

```
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
>>>>>>>>>>
<<<<<<<<<<
```

4.23 *(Dangling-`else` Problem)* C++ compilers always associate an `else` with the immediately preceding `if` unless told to do otherwise by the placement of braces (`{` and `}`). This behavior can lead to what is referred to as the **dangling-`else` problem**. The indentation of the nested statement

```
if (x > 5)
   if (y > 5)
      cout << "x and y are > 5";
else
   cout << "x is <= 5";
```

appears to indicate that if `x` is greater than `5`, the nested `if` statement determines whether `y` is also greater than `5`. If so, the statement outputs the string `"x and y are > 5"`. Otherwise, it appears that if `x` is not greater than `5`, the `else` part of the `if...else` outputs the string `"x is <= 5"`. Beware! This nested `if...else` statement does *not* execute as it appears. The compiler actually interprets the statement as

```
if (x > 5)
   if (y > 5)
      cout << "x and y are > 5";
   else
      cout << "x is <= 5";
```

in which the body of the first `if` is a *nested* `if...else`. The outer `if` statement tests whether `x` is greater than `5`. If so, execution continues by testing whether `y` is also greater than `5`. If the second condition is *true*, the proper string—`"x and y are > 5"`—is displayed. However, if the second condition is *false*, the string `"x is <= 5"` is displayed, even though we know that `x` is greater than `5`. Equally bad, if the outer `if` statement's condition is *false*, the inner `if...else` is skipped and nothing is displayed. For this exercise, add braces to the preceding code snippet to force the nested `if...else` statement to execute as it was originally intended.

ANS:

```
if (x > 5) {
   if (y > 5) {
      cout << "x and y are > 5";
   }
}
else {
   cout << "x is <= 5";
}
```

4.24 *(Another Dangling-`else` Problem)* Based on the dangling-`else` discussion in Exercise ANS: , state the output for each of the following code snippets when `x` is `9` and `y` is `11` and when `x` is `11` and `y` is `9`. We eliminated the indentation from the following code to make the problem more challenging. [*Hint:* Apply indentation conventions you've learned.]

a)
```
if (x < 10)
if (y > 10)
cout << "*****" << endl;
else
cout << "#####" << endl;
cout << "$$$$$"  << endl;
```

ANS: When `x` is `9` and `y` is `11`

```
*****
$$$$$
```

ANS: When `x` is `11` and `y` is `9`

```
$$$$$
```

b)
```
if (x < 10)
{
if (y > 10)
cout << "*****" << endl;
}
else
{
cout << "#####" << endl;
cout << "$$$$$" << endl;
}
```

ANS: When `x` is `9` and `y` is `11`

```
*****
```

ANS: When `x` is `11` and `y` is `9`

```
#####
$$$$$
```

4.25 *(Another Dangling-`else` Problem)* Based on the dangling-`else` discussion in Exercise ANS: , modify the following code to produce the output shown. Use proper indentation techniques. You must not make any additional changes other than inserting braces. We eliminated the indentation from the following code to make the problem more challenging. [*Note:* It's possible that no modification is necessary.]

```
if (y == 8)
if (x == 5)
cout << "@@@@@" << endl;
else
cout << "#####" << endl;
cout << "$$$$$" << endl;
cout << "&&&&&" << endl;
```

a) Assuming `x = 5` and `y = 8`, the following output is produced.

```
@@@@@
$$$$$
&&&&&
```

ANS:
```
if (y == 8) {
   if (x == 5) {
      cout << "@@@@@" << endl;
   }
   else {
      cout << "#####" << endl;
   }
}

cout << "$$$$$" << endl;
cout << "&&&&&" << endl << endl;
```

b) Assuming `x = 5` and `y = 8`, the following output is produced.

```
@@@@@
```

ANS:
```
if (y == 8) {
   if (x == 5) {
      cout << "@@@@@" << endl;
   }
   else {
      cout << "#####" << endl;
      cout << "$$$$$" << endl;
      cout << "&&&&&" << endl;
   }
}
```

c) Assuming `x = 5` and `y = 8`, the following output is produced.

```
@@@@@
&&&&&
```

ANS:
```
if (y == 8) {
   if (x == 5) {
      cout << "@@@@@" << endl;
   }
   else {
      cout << "#####" << endl;
      cout << "$$$$$" << endl;
   }
}

cout << "&&&&&" << endl << endl;
```

d) Assuming `x = 5` and `y = 7`, the following output is produced. [*Note:* The last three output statements after the `else` are all part of a block.]

```
#####
$$$$$
&&&&&
```

ANS:
```
if (y == 8) {
   if (x == 5) {
      cout << "@@@@@" << endl;
   }
}
else {
   cout << "#####" << endl;
   cout << "$$$$$" << endl;
   cout << "&&&&&" << endl;
}
```

4.32 What's wrong with the following statement? Provide the correct statement to accomplish what the programmer was probably trying to do.

```
cout << ++(x + y);
```

ANS: The ++ operator must be used in conjunction with variables. The programmer probably intended to write the statement: `cout << x + y + 1;`.