Programming Fundamentals
CS1336

Loops and File I/O

Loops and File I/O

Introduction

Your seventh programming assignment will consist of two small C++ programs. Each program will be independent of the other program. Each one should compile correctly and produce the specified output.

Please note that each of the programs should comply with the commenting and formatting rules we discussed in class. For example, there should be a header for the whole program that gives the author's name, class name, date, and description. End braces should be commented, and there are alignment and indenting requirements as discussed. Please ask if you have any questions.

Program #1

Write a program that asks the user to enter an integer between 1 and 15. If the number entered is outside that range, your program should print out an error message and re-prompt for another number. (This is an opportunity to write an **input validation loop**.)

Once the user enters a number in the correct range, your program will print out four patterns of asterisks that look like the following. Note that the number provided by the user is equal to the height of the pattern.

Let us say that the input value was 5. Then please print the following four patterns:

```
*****
****
***
**
*
```

```
*****
 ****
  ***
   **
    *
```

```
    *
   ***
  *****
 *******
*********
```

```
    *
   * *
  *   *
 *     *
*********
```

If the input value is 8, then the output should be:

```
********
*******
******
*****
****
***
**
*
```

```
********
 *******
  ******
   *****
    ****
     ***
      **
       *
```

```
       *
      ***
     *****
    *******
   *********
  ***********
 *************
***************
```

```
       *
      * *
     *   *
    *     *
   *       *
  *         *
 *           *
***************
```

As you can see, the input value represents the height of the triangle.

Your program should only print one asterisk at a time and use loops to control the number of asterisks printed per line.

<u>Bonus</u>

For a 5% bonus, ask the user for an odd number between 1 and 15. (Your program should use an input validation loop to restrict valid input from the user to odd numbers between 1 and 15.) If that number is 7, then the diagram appears as follows:

```
* * * * * * *
  *       *
    *   *
      *
    *   *
  *       *
* * * * * * *
```

Note that, in contrast to the previous diagrams, there are spaces between adjacent asterisks on the first and last lines.

Program #2

This problem is a modification of Problem 24 on page 302 of the Gaddis text (9E) with two small additions. (A scan of this problem can be found on the last page of the assignment.)

In this assignment, your program will read in a list of random numbers from an input file, `random.txt` (found on eLearning in the "Homework Input Files" folder), and calculate the following statistics on those numbers:

A. The number of numbers in the file.
B. The sum of the numbers in the file.
C. The average of the numbers in the file.
D. The largest number in the file.
E. The smallest number in the file.
F. The second largest number in the file.

G. The second smallest number in the file.

Your program should then print out a report, similar to the following (note that these are demonstration numbers only; they are not accurate for the homework data file):

```
Number of numbers:          235
Sum of numbers:            5346
Average of numbers:        22.8
Largest number:             515
Smallest number:              3
Second largest number:      512
Second smallest number:       7
```

Programming Notes:

1. Note that the number of numbers, the sum of the numbers, the largest and smallest numbers are integers, whereas the average should be a floating point data type printed to one decimal place.

2. All of the numbers in the file are between 0 and 1000. Therefore, the minimum value can be found by using the following technique:
   a. Create an `int` variable `minValue` and initialize it to some number larger than any number in the file, say 2000.
   b. Then include the following statement inside your file processing loop:

   ```
   if (nextValue < minValue)
      minValue = nextValue;
   ```

   This assumes that `nextValue` is the next value being read from the input file.
   c. Note that the value in `minValue` is reset only if we find a number smaller than its previous value. After the processing loop ends, `minValue` will hold the smallest value in the data set.
   d. A similar technique will calculate the maximum value.

3. All of these problems can be solved with a single pass through the data set. Your program should not have to make multiple passes through the data set.

4. Please make sure your program follows the protocols for good file use. Especially, (a) test to make sure the file was opened properly before using it, and (b) close the file when finished. <u>If the file was not opened properly, the program should print out an error message but not attempt to read or process the data from the file</u>.

Supplemental

24. Using Files—Numeric Processing

If you have downloaded this book's source code from the companion Web site, you will find a file named Random.txt in the Chapter 05 folder. (The companion Web site is at www.pearsonhighered.com/gaddis.) This file contains a long list of random numbers. Copy the file to your hard drive and then write a program that opens the file, reads all the numbers from the file, and calculates the following:

A) The number of numbers in the file

B) The sum of all the numbers in the file (a running total)

C) The average of all the numbers in the file

The program should display the number of numbers found in the file, the sum of the numbers, and the average of the numbers.