

## Arrays: Tracking Precipitation Amounts

The National Climatic Data Center has hired you to write a report generating program for daily precipitation amounts. Normally, this task would be straightforward, but a computer glitch has caused some of the data to appear out of order, to be lost, or to have the recorded date corrupted. You are responsible for storing as much valid data as possible, recognizing errors in the data file, and displaying a summary of daily precipitation amounts for the specified month.

You can find out more about the National Climatic Data Center and find real online precipitation data at <http://www.ncdc.noaa.gov/ol/ncdc.html>

### The input file:

The input file for this program is named "Precip.txt". The first line contains human readable information that is not needed by the program. The second line contains the location of the weather station reporting precipitation. This line will always contain a newline immediately following the location. The third line contains the full month name, which is capitalized, followed immediately by a comma, and then the year for recorded information. The fourth and subsequent lines contain the day for recorded information and the amount of precipitation in inches reported on that day, which are separated by whitespace characters (spaces or tabs).

```
CS1044 Project 8 Spring 2001
Blacksburg, VA
December, 2000
2 0
4 0
5 0
26 0
6 0
8 0.01
9 0
10 0.02
13 0
114 0
15 0
17 0.55
19 1.8
8 0.08
20 4.12
24 0
32 0.73
28 0
11 0.05
12 0.01
29 0
```

### Calculations and error checking:

Your program will read and store the precipitation information in an array. The array will have to be large enough to store information for any month, although the entire array might not be used. You should initialize the entries in the array to some well-known, but impossible precipitation value so that you can readily identify when data is unavailable. Once all of the precipitation data has been read, you will need to calculate the minimum, maximum, and average precipitation values.

The input file may contain two kinds of errors. The first is that the day given in the input might be invalid (too small or large). The second is that a file might contain multiple entries for the same day. These errors should be recognized and an error message displayed, including the line number where the error occurred. See the sample output file below for the exact format of the error messages.

You will need to determine if a year is a leap year to determine the number of days properly. A year divisible by 4 but not 100 is generally a leap year. Years divisible by 400 are also leap years (as was the case with the year 2000).

**The output file:**

The output file is named "Report.txt". An output file, which corresponds to the given input file, is shown below. The first two lines contain programmer and project information. The third line is blank. The fourth line contains the location, month, and year of the precipitation data. The fifth line is blank. Remaining information is printed out as follows:

- ◆ Any errors detected in the input file. If errors are detected, a blank line is printed after the last error.
- ◆ A histogram for the precipitation amounts. The day and precipitation amount to two digits of precision are printed. Following the day and amount, a graph containing one star for each 0.25 inches or part thereof is displayed. For example, 0.01-0.25 inches will display one star, 0.26-0.50 two stars, 0.51-0.75 three stars, etc. If no precipitation data is available for a day, NA should be printed for the amount with no stars following. A blank line is printed after the graph.
- ◆ The maximum, minimum, and average precipitation amounts are printed. **The average is the sum of valid precipitation values divided by the number of days in the month.** If no information is available for the entire month, NA should be printed for all three values.

```

Programmer: Craig Struble
CS 1044 Project 8 Spring 2001

Precipitation report for Blacksburg, VA during December, 2000

Error          Day          Line
Invalid        114          13
Repeated       8            17
Invalid        32           20

Day Amount Graph
1      NA
2     0.00
3      NA
4     0.00
5     0.00
6     0.00
7      NA
8     0.01 *
9     0.00
10    0.02 *
11    0.05 *
12    0.01 *
13    0.00
14     NA
15    0.00
16     NA
17    0.55 ***
18     NA
19    1.80 *****
20    4.12 *****
21     NA
22     NA
23     NA
24    0.00
25     NA
26    0.00
27     NA
28    0.00
29    0.00
30     NA
31     NA

Minimum      Maximum      Average
0.00         4.12         0.21

```

**Documentation and other requirements:**

You must meet the following requirements (in addition to designing and implementing a program that merely produces correct output):

- Your program, at a minimum, must contain 6 functions in addition to main().
- You must make appropriate use of C++ parameter-passing mechanisms.
- At least one of your functions must be a value-returning function.
- You must include a top-down design of your program in comments following your pledge statement. See Appendices 3 and 4 in the course notepack for design examples.
- Document each function with a function documentation header as described in the notes
- Define your functions after your main program
- Use function prototypes appropriately
- Write a header comment with your identification information, the required pledge statement (below), and a brief description of what the program does.
- Write a comment explaining the purpose of every variable and named constant you use.
- Write comments describing what most of the statements in your program do.
- Use descriptive identifiers for variables and for constants.
- Use named constants instead of “magic values” whenever it is appropriate.
- Do **NOT** use `break`, `continue`, `return`, or `goto` statements to exit a loop prematurely.
- Do **NOT** use global variables. Global constants are OK.

**Submitting your program:**

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* can be found at: <http://ei.cs.vt.edu/~eags/Curator.html>

The submission client can be found at: <http://spasm.cs.vt.edu:8080/curator/>

**Evaluation:**

Your submitted program will be assigned a score based upon the runtime testing performed by the Curator System. We may very well evaluate your submission of this program for documentation style, and to see whether you followed the requirements given in this specification. Therefore, you should compare your comments to those given in the programs for projects 1 and 2, as well as consider the scoring of your fourth and fifth projects. The programs serve as useful a guide to acceptable documentation style at this point in the course.

If your program is evaluated for documentation and requirements, your instructor will specify how that score will be counted.

**Pledge:**

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:  
//  
// - I have not discussed the C++ language code in my program with  
//   anyone other than my instructor or the teaching assistants  
//   assigned to this course.  
//  
// - I have not used C++ language code obtained from another student,  
//   or any other unauthorized source, either modified or unmodified.  
//  
// - If any C++ language code or documentation used in my program  
//   was obtained from another source, such as a text book or course  
//   notes, that has been clearly noted with a proper citation in  
//   the comments of my program.  
//  
// - I have not designed this program in such a way as to defeat or  
//   interfere with the normal operation of the Curator System.  
//  
// <Student Name>
```

**Failure to include this pledge in a submission is a violation of the Honor Code.**