Assg 06: Calculations on Arrays

CSci 515 Spring 2015

<2015-02-19 Thu>

Dates:

Due: Tuesday March 10, by Midnight

Objectives

- More practice writing C functions that take arrays as parameters
- More work on declaring, indexing and processing arrays.
- Learn about performing calculations over arrays of values.
- More examples of breaking problems into subproblems using functions.

Description

In our lab for this week, you were given a template that reads in an array of floating point values from a file and saves them in to an array for processing. For this assignment, you need to start with the same template.

For this assignment, I have provided a different file off 100 floating point values, named `assg-06-float-array.txt`. Modify the constants and code in the original template for lab 06 to read in the 100 values from this file into your array named `values`.

In this assignment you will be writing 5 functions to process the data in your `values` array. Your functions will find the minimum value in an array, the maximum value in an array, the average of the values in an array, the sum of the values in an array, and the standard deviation of the values in an array. All 5 of these functions take the array, and an integer parameter indicating the size of the array, as input parameters to them. All 5 of these function return a floating point result, which is the result of finding or calculating the

minimum / maximum / average / sum / standard deviation of the values in the array.

Perform the following tasks:

1. Write a function called `findMinimumValue`. This function takes an array of floats as its first parameter, and the size of the array as its second parameter. This function should search through all of the values in the array and return the value that is the smallest (the minimum) in the array.

2. Write a function called `findMaximumValue`. This function takes an array of floats as its first parameter, and the size of the array as its second parameter. This function should search through all of the values in the array and return the value that is the largest (the maximum) in the array. This function is basically almost an exact repeat of the previous function, except for one small change. In general, repeating code like this is a bad idea, but for this assignment simply repeat your logic and make the one small change to find the maximum instead of the minimum. Later in this course we will look at some ways we can encapsulate this repeated bit of functionality into a function or abstraction.

3. Write a function called `findSum`. This function takes an array of floats as its first parameter, and the size of the array as its second parameter. This function again is almost a complete repeat of the previous two function, but in this case you need to add a variable to keep track of and calculate the running `sum` of the values in the array. This function returns a float value, which should be the sum of all of the values in the given input array.

4. Write a function called `findAverage`. This function takes an array of floats as its first parameter, and the size of the array as its second parameter. This function will calculate the average of the values in the input array, and return this average (a float) as its result. The average is calculated by taking the sum of the values and dividing this by the total number of values. Thus your implementation of this function **MUST** use the `findSum` function to calculate the sum, then compute the average from (reusing) this functions result. Thus this function will be quite a bit different from the previous 3 functions, as you are reusing the sum function as a black box in order to calculate an average.

5. Write a function called `findStandardDeviation`. As with all of the previous functions, this function takes an array of floats as its first parameter, and the size of the array as its second parameter. This function will calculate the standard deviation of the values in the given input array. The formula to calculate the standard deviation is as follows:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(x_i - \mu)^2}$$

Here μ represents the average of the values in the array (which you can obtain from the `findAverage` function), and x_i is mathematical notation that basically represents the elements of our array indexed from 0 up to $N-1$. So to find the standard deviation, you have to subtract each value in the array from the average value and square this difference. You add up all of the squared differences, and divide the result by N. This value, finally, you take the square root of. This is, by definition, the standard deviation of the values in the array. For this function, you should use the `findAverage` function to get the average (μ), and then you need to loop through all of the values in the array to square the difference, sum them up, and then divide by N and finally take the square root. You will want to use the `pow()` method from the `cmath` library in order to do the squaring and square root calculations.

6. In your `main()` function, modify the code at the beginning as already described, to read in the 100 values from the file provided for assignment 06. Then use your 5 functions to calculate the minimum, maximum, sum, average and standard deviation of the values in the loaded array.

Your program output should look exactly like the following when I run your program. These are the correct calculated values for the given file of 100 floating point values you will use for this assignment.

```
Minimum:  0.00479889
Maximum:  0.99799562
    Sum: 50.07558441
Average:  0.50075585
    Std:  0.27921614
```

NOTE: Now that our programs have more functions than just the `main()` function, the use of the function headers becomes meaningful and required. Make sure that all of your functions have function headers preceding

them that document the purpose of the functions, and the input parameters and return value of the function.

Assignment Submission

An eCollege dropbox has been created for this assignment. You should upload your version of the assignment to the eCollege dropbox named `Assg 06 Array Calculations` created for this submission. Work submitted by the due date will be considered for evaluation.

Requirements and Grading Rubrics

Program Execution, Output and Functional Requirements

1. Your program must compile, run and produce some sort of output to be graded. 0 if not satisfied.

2. 40+ pts. Your program must the first 4 functions described.

3. 40+ pts. Each of the first 4 functions must correctly calculate the desired value of the given input array. Your average function must reuse the sum function to perform its calculation.

4. 20+ pts. You must correctly modify the original template to read from the new input file for assignment 06, with 100 elements in the array. Don't use magic numbers, modify the global constants to do this correctly.

5. 5 extra credit pts. Your program should correctly implement the calculation to determine the standard deviation. Your standard deviation function must reuse the average function in doing its calculation.

Program Style

Your programs must conform to the style and formatting guidelines given for this course. The following is a list of the guidelines that are required for the assignment to be submitted this week.

1. The file header for the file with your name and program information and the function header for your main function must be present, and filled out correctly.

2. A function header must be present for all functions you define. You must document the purpose, input parameters and return values of all functions. Your function headers must be formatted exactly as shown in the style guidelines for the class.

3. You must indent your code correctly and have no embedded tabs in your source code. (Don't forget about the Visual Studio Format Selection command).

4. You must not have any statements that are hacks in order to keep your terminal from closing when your program exits (e.g. no calls to system()).

5. You must have a single space before and after each binary operator.

6. You must have a single blank line after the end of your declaration of variables at the top of a function, before the first code statement.

7. You must have a single blank space after , and ; operators used as a separator in lists of variables, parameters or other control structures.

8. You must have opening `{` and closing `}` for control statement blocks on their own line, indented correctly for the level of the control statement block.

9. All control statement blocks (if, for, while, etc.) must have `{ }` enclosing them, even when they are not strictly necessary (when there is only 1 statement in the block).

10. You should attempt to use meaningful variable and function names in your program, for program clarity. Of course, when required, you must name functions, parameters and variables as specified in the assignments. Variable and function names must conform to correct `camelCaseNameingConvention` .

Failure to conform to any of these formatting and programming practice guidelines for this assignment will result in at least 1/3 of the points (33) for the assignment being removed for each guideline that is not followed (up to 3 before getting a 0 for the assignment). Failure to follow other class/textbook programming guidelines may result in a loss of points, especially for those programming practices given in our Deitel textbook that have been in our required reading so far.