Assg 06: Calculations on Arrays

CSci 515 Spring 2015

<2015-02-19 Thu>

Dates:

Due: Tuesday March 3, 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 4 function to process the data in your `values` array. Your function will find the minimum value in an array, the maximum value in an array, the average of the values in an array, and the standard deviation of the values in an array. All 4 of these function take the array, and an integer parameter indicating the size of the array, as input parameters to them. All 4 of these function return a floating point result,

which is the result of finding/calculating the minimum / maximum / average / 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 `findSumValue` 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:

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

1. In your `main()` function, prompt the user to enter a value for n. You should then call each of your two different implementations, and display the result of their calculation of the n^{th} term of the Fibonacci sequence, which if you implement the two different versions correctly, should always give the same result. Be careful of using large values of n, as the recursive implementation especially may take quite a while to calculate in cases of large n.

Your program output should look something close to the following when I run your program. I have run the program multiple times, so that you can also see some examples of correct output for some larger values of n:

```
$ assg05
Enter n (an integer >= 0), and I will calculate the n^th
Fibonacci term for you using two different methods: 0

0 term of the Fibonacci series, using iterative method: 0
0 term of the Fibonacci series, using recursive method: 0

$ assg05
Enter n (an integer >= 0), and I will calculate the n^th
Fibonacci term for you using two different methods: 1

1 term of the Fibonacci series, using iterative method: 1
1 term of the Fibonacci series, using recursive method: 1

$ assg05
Enter n (an integer >= 0), and I will calculate the n^th
Fibonacci term for you using two different methods: 8
```

```
8 term of the Fibonacci series, using iterative method: 21
8 term of the Fibonacci series, using recursive method: 21

$ assg05
Enter n (an integer >= 0), and I will calculate the n^th
Fibonacci term for you using two different methods: 10

10 term of the Fibonacci series, using iterative method: 55
10 term of the Fibonacci series, using recursive method: 55

$ assg05
Enter n (an integer >= 0), and I will calculate the n^th
Fibonacci term for you using two different methods: 35

35 term of the Fibonacci series, using iterative method: 9227465
35 term of the Fibonacci series, using recursive method: 9227465
```

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 (`main`, `nthFibonacciIterative`, `nthFibonacciRecursive`) 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 05 Fibonacci Sequence` 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. 25+ pts. Your program must have the 2 required named functions, that accept the required input parameters and return the required values (if any).

3. 25+ pts. Your iterative implementation must use loops/iteration to implement its calculation. The function must of course correctly compute the n^{th} term of the series.

4. 40+ pts. Your recursive implementation must perform its calculation using recursion. You must have the correct base cases defined. Your function must of course correctly compute the n^{th} term of the series. trials, and count up the successful trials from all of the trials performed, and return the correct probability ratio. Your ratio must be correct.

5. 10+ pts. You must prompt the user for n in main, and correctly display the returned results form your function as shown.

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.

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.

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.

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.