Lab 06: Processing Arrays

CSci 515 Spring 2015

2015-02-18

Dates:

Due: In Lab, Wednesday February 25, by 4:05 pm (lab end time)

Objectives

- Learn about declaring, accessing and processing array elements
- Practice passing arrays to and from functions.
- Process arrays element by element to calculate statistical properties of the data
- Learn about summarizing the results from processing arrays of data

Description

In this lab you will be given a function, and an example of how to call the function, that will read in an array of values from a file. You will be processing arrays of floating point values. You will write two function to process the elements in an array. In the first function, you will pass in an array, and a begin and end range. This function will display the values in the array, formatted on standard output as specified below. Your other function will take an array, and will calculate the average of the values in the array. This function will return this average as its result. This second functions expects an array of floats as input, and returns the average as a floating point result.

Perform the following tasks:

1. Start with the given template code for this lab, which contains one function already written for you, and an example of calling that function. This function reads in an array of floating point values from a file. As we learned in class, since arrays are passed by reference by default, specifying the array to be read in to the function, causes the values to be filled in and returned to the caller in the given array.

2. Write a function called `displayArrayValues`. This function will take an array of floats as its first parameter, and a `beginRange` and `endRange` as its second and third parameters. These parameters are of type int, and they will be used to specify the beginning index and end index of values to be displayed from the array. These values should be $>= 0$, and the `endRange` should be less than the size of the input array. If `beginRange` == `endRange` then only 1 value will be displayed. If `beginRange` > `endRange` then no values will be output. Your function does not return anything, so it is a void function. Instead your function displays its output to standard out, and it should be formatted to look exactly like this output:

3. Write a function called `calculateArrayAverage`. This function will take an array of floats as its first parameter, and an int `size` which is the number of elements in the array. This function will return a floating point result. The function should calculate the average of all of the values in the array in the indexes from 0 to size - 1 and return this average as the result of calling the function.

4. In your main program, you should leave the code alone that reads in the array values from the file. After the array is read in, you should first prompt the user for a begin and end range, and call the `displayArrayValues` function to display the values in the array in that range. Then you should use the `calculateArrayAverage` function to calculate the average of the values in the array, and display this result.

Your program output should look something close to the following when I run your program:

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.

Lab Submission

An eCollege dropbox has been created for this lab. You should upload your version of the lab by the end of lab time to the eCollege dropbox named `Lab 06 Processing Arrays`. Work submitted by the end of lab will be considered, but after the lab ends you may no longer submit work, so make sure you submit your best effort by the lab end time in order to receive credit.

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 have the 2 required named functions, that accept the required input parameters and return the required values (if any).

3. 20+ pts. Your `displayArrayValues` function must correctly format the displayed output on standard output. Your program should work if the begin and end range are equal, and should show now output when begin is greater than the end specified.

4. 20+ pts. Your `calculateArrayAverage` function must calculate the average of all of the values specified correctly, and return this result.

5. 20+ pts. Your main function must prompt the user as specified, and display the output formatted correctly 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 lab 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 lab 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.