Eh, Ray?

Professor Caleb Fowler

1/10/24

# Problem.

Your friend Ray has asked you to help demonstrate the abilities of arrays. Your task is to write a program which shows off the capabilities of arrays. You will create 2 10 element arrays, one for integers and one for floats. You will load each with values in whatever way you choose.

# Requirements.

* Store the results of all computations.
* Include your speciﬁcation comments above the main portion of your code where you implement the speciﬁcation. No credit if this is missing!
* Use white-space and comments to make your code more readable.
* Put a Source File Header at the top of your source ﬁle.
* Do not use global arrays. Pass the arrays instead.
* Do not use the vector or array libraries (you will not need them).
* Use function prototypes for all functions. Functions go below main().
* Do not use c (.h) style libraries. Use C++ libraries instead.
* Your program must compile in C++ on Ubuntu.
* Your program must generate logically correct output.
* Program activities are split into logical ’chunks’ or paragraphs. I’m expecting paragraphs for input, processing (if any), and output operations.
* If there is non-integer output, force the computer to always display 3 places to the right of the decimal.
* Include a **Source File Header**, like you used in homework 1. Do not include a specification with it like you did in that assignment.
* Include a **Commented Sample Run** for this assignment like you did for homework 1. Again, do not include a specification with it.
* Include a **ProgramGreeting** for this assignment like you did for homework 1, but turn it into a function you call at the very start of main().

# Speciﬁcations.

**~~// Speciﬁcation C1 – Display Array~~**

~~Display the elements of each array on the screen.~~

**~~// Speciﬁcation C2 – Sum~~**

~~Total all the elements in each array and display the totals.~~

**~~// Speciﬁcation C3 – Min and Max~~**

~~Automatically find the smallest and largest element in each array and display them. Do not feel constrained to have your code do ‘double duty’ - write code to find the min and other code to find the max if you wish.~~

**~~// Speciﬁcation B1 – Value Hunt~~**

~~Prompt Ray to enter a value for each array and then go looking for it. Indicate if that value is or is not present in the array. Use 0.05 margin of error for the floating point numbers.~~

**~~// Speciﬁcation B2 – A Ray Reversal~~**

~~Reverse the order of the elements in both arrays (so element #1 is now element #10, element #2 is now element #9, etc. etc.). Don’t just display them reversed, actually reverse the values. Display the arrays before and after you reverse them.~~

**~~// Speciﬁcation B3 - Element Shift~~**

~~Shift all elements in the integer array 1 spot to the right. Shift all elements of the float array 1 spot to the left. Actually shift the elements - don’t just display them shifted. Display the arrays before and after you shift them.~~

**~~// Specification A1 - Function Activity to Disk~~**

~~Write a message when every function is called. Save this in a file called “log.txt”. Include a timestamp and a message string for each line you save in your log. Log when a function is called - at a minimum. If you also log the time the function ended, you can then compute the Elapsed Time (or have the program compute it!) for your Peer Review. It is also VERY helpful for debugging to log the incoming and outgoing parameter values.~~

**~~// Specification A2 - Implement a Stack~~**

~~Create a stack data structure. Use a loop to Pop() between 15 and 25 random integers on the stack. Then display the contents of the stack to the terminal.~~

**~~// Specification A3 - Random Number in Singleton Pattern~~**

~~Use a singleton design pattern to create a random number generator which is initialized from the system time~~ **~~only~~** ~~the first time it is called. Subsequent calls simply return a random number. Use this to generate random numbers between 0 to 100, inclusive. Initialize the integer array with multiple calls to this function. We saw this when we discussed static variables in the chapter on functions.~~