CS 161B: Programming and Problem Solving II

Assignment 3: Parallel Arrays

Grade Level A

Academic Integrity

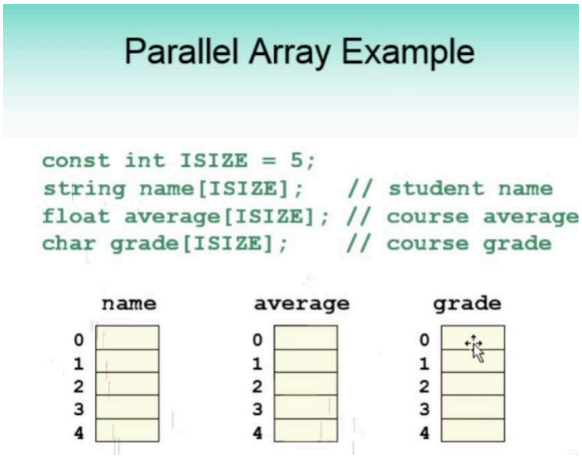
You may NOT, under any circumstances, begin a programming assignment by looking for completed code on StackOverflow or Chegg or any such website, which you can claim as your own. Please check out the Student Code of Conduct at PCC.

The only way to learn to code is to do it yourself. The assignments will be built from examples during the lectures, so ask for clarification during class if something seems confusing. If you start with code from another source and just change the variable names or other content to make it look original, you will receive a zero on the assignment.

I may ask you to explain your assignment verbally. If you cannot satisfactorily explain what your code does, and answer questions about why you wrote it in a particular way, then you should also expect a zero.

Arrays store data contiguously by representing data with a common name and distinguishing it by its index. This is like a parking garage: the garage stores vehicles; all parking spaces have a common name (the name of the garage), and a specific parking space number identifies each parked vehicle. This is also true of arrays. Think of arrays as parking garages for our data!

Multiple arrays are also called parallel arrays. For your assignment, you will be storing related information in multiple arrays.



Purpose

In this assignment, you will be writing a program that stores a list of scores and the corresponding letter grades in two parallel arrays. The scores will be double data type and the letter grade will be a char data type.

After completing this assignment you will be able to:

- Define an array to hold multiple pieces of information of the same data type
- Define multiple/parallel arrays to store related information
- Use an appropriate loop to read input from the user into the array
- Use a for loop to manipulate the array without going out of bounds
- Pass multiple arrays and their size to a function and manipulate the arrays
- Do input data validation and use the correct exit condition to exit out of the loop
- Convert an algorithm to code

Task - This is the Grade A version.

What grade would you like?

We understand that your time and money is valuable, and we want to ensure that this programming assignment is tailored to your needs and circumstances. Whether you are a student juggling multiple commitments, a student dealing with personal responsibilities, or a student facing time constraints, or other issues in life, this assignment aims to provide programming skills at different levels that will meet the objectives of this week. With that in mind this assignment has been broken down into 3 different grade levels - Grades A (Exceeds), B (Meets) and C (Approaching). ***This is the Grade A version.***

Tasklist for getting a grade A:

The below set of tasks, if completed correctly per the rubric will get you a grade of **Exceeds** and help master the concepts required to learn the objectives for this week's topic. You can improve your grade by resubmitting the assignment once.

- ❏ Before you get started:
  - ❏ Check out Sample Assignment A03 - Algorithmic Design document
  - ❏ Check out Sample Assignment A03 - SampleA03.cpp
- ❏ Your job is to design and create a program to calculate the letter grade based on a numerical score. You will first read the numerical scores from the user as input into a double array, and then calculate the letter grade and fill another parallel array and print the two arrays as output.
- ❏ The numerical score must be between 0 and 4 inclusive. The user should not be allowed to enter a score less than 0 or greater than 4.

- You will also sort the arrays based on the scores, and calculate the median of the scores and output the sorted arrays and the median in main().
- Write the following functions and call them in the same order as instructed in the `main()` function to accomplish this Task.
- **`void welcome()`** - This function prints a welcome message to the user.
- **`void readScores(double scores[], int &count)`**
  - Reads a list of scores from the user. A -1 indicates the end of the input and is not stored in the array. (Use an appropriate loop!! Check out this example code - a for loop is count controlled, and a while loop is condition based.)
  - Call the **`readDouble`** function to do this. You must catch all invalid data such as characters, negative numbers other than -1 etc.
  - The variable **`count`** keeps track of the number of scores entered.
  - You must also do validation to make sure that the number is between 0 and 4 inclusive and nothing other than that. Use a while loop to do this.
- **`void readDouble(string prompt, double &num)`**
  - This function should be used any time you read any floats or doubles from the user. Use this function to read the numerical scores from the user.
  - It takes a string prompt, outputs it, reads a number from the user, validates and returns the num by reference.
  - Write it exactly like the **`readInt`** from assignment 1 but declare a double or float instead of an int. See Samplea01.cpp for the **`readInt`** function.
  - You must catch all invalid data such as characters, negative numbers etc.
- **`void calcGrade(double scores[], char grade[], int count)`**
  - Use a loop to process each array element and calculate the letter grade for each score.
  - Use the table below to assign grades A to F to the corresponding numerical score.

Letter Grade	4.0 Score	Level
A	>3.3 <= 4.0	Exceeds
B	>2.7 <= 3.3	Meets
C	>1.9 <= 2.7	Approaching
D	>1.1 <= 1.9	Not Yet
F	>0.0 <= 1.1	No Evidence

- **`void printList(double scores[], char grades[], int count)`**

- Go through a for loop and print the scores and the corresponding grades for each item.

- **`void sort(double scores[], char grade[], int count)`**
  - Sort the arrays using the given sorting algorithm. This is called Selection Sort. Use only this algorithm to sort your list. To see how the Selection Card Sort Algorithm works, watch this video from Virginia Tech.
  - Be careful and make sure you sort based on the scores array and swap the corresponding element in the **`grade`** array to maintain the correspondence between the two arrays. Meaning if you swap the scores in index 0 and 5, you must also swap the corresponding grades in index 0 and 5.
  - Watch this Python Video to help you with the sorting algorithm. Try this Selection Sort Animation by Y. Daniel Liang.

```
procedure selection sort
   list  : array of items
   count : size of list
   for i = 0 to count - 1
   /* set current element as minimum*/
     min = i
     /*go through the list and find the smallest
element*/
     for j = i+1 to count
          if list[j] < list[min] then
               min = j;
          end if
     end for
     /* swap the minimum element with the current
element
     If they are not the same element*/
     if min != i  then
          swap list[min] and list[i]
     end if
   end for
end procedure
```

- **`double median(double scores[], int count)(***new to this level***)`**
  - After sorting, write this function to identify the median score. The median is located in the middle of the array if the array's size is odd. Otherwise, the median is the average of the middle two values.
  - Return the median to `main()` and output in `main()` with two decimal places.

- **`int main()`**

- ❏ Declare all variables needed. The 2 arrays (double scores[], and char grades[]) must be declared in main().
- ❏ Call **`readScores()`** and send scores and count to it. This will fill the scores array from the user. You should call the **`readDouble()`** function to read and validate the scores before adding them to the scores array.
- ❏ count will have the number of values read.
- ❏ Call **`calcGrade()`** function that takes the scores array and an empty grades array and fills the grades array with letter grades.
- ❏ Call the **`printList`** function to print the lists.
- ❏ Call the **`sort`** function to sort the list based on scores.
- ❏ Call the **`printList`** function to print the lists.
- ❏ Call the **`median`** function to find the median and print it in **`main()`**.

- ❏ **Assume the arrays will always contain fewer than 20 values. You must not let the user enter more than 20 values.**
- ❏ Open the Algorithmic Design Document, make a copy, and follow the steps to create your algorithm.
- ❏ You must express your algorithm as **pseudocode.**
- ❏ **Check out all the coding constructs under Criteria for Success.**

Criteria for Success

- ❏ Use the below sample run as a guide to test your program.

```
Welcome to my Parallel Arrays program!
Please enter the list of scores (-1 to end input:)
Valid scores are between 0 and 4 inclusive.
>> 3.5
>> 2.7
>> 3.3
>> 2.5
>> 3.2
>> 1.5
>> 4.0
>> 3.7

Your stats are as below:

The list of scores and their grades are:
3.5 A
2.7 C
3.3 B
2.5 C
3.2 B
1.5 D
4.0 A
```

```
3.7 A

The list sorted by scores in ascending order:
1.5 D
2.5 C
2.7 C
3.2 B
3.3 B
3.5 A
3.7 A
4.0 A

The median score is 3.25


Thank you for using my Parallel Arrays program!!
```

```
Welcome to my Parallel Arrays program!
Please enter the list of scores (-1 to end input:)
Valid scores are between 0 and 4 inclusive.
>> 3.5
>> 2.7
>> 3.3
>> 4.5
Invalid score! Please try again!!
>> abc
Invalid score! Please try again!!
>> 3.2
>> 1.5
>> 4.0
>> 3.7
>> 4.0

Your stats are as below:

The list of scores and their grades are:
3.5 A
2.7 C
3.3 B
3.2 B
1.5 D
4.0 A
3.7 A
4.0 A

The list sorted by scores in ascending order:
1.5 D
2.7 C
3.2 B
3.3 B
3.5 A
```

```
3.7 A
4.0 A
4.0 A

The median score is 3.40


Thank you for using my Parallel Arrays program!!
```

- ❏ Complete zyBooks sections **11.1 to 11.11 CS161B: Arrays**, and **zyLabs 11.15 and 11.16**
- ❏ Complete all sections of your Algorithmic Design Document.
- ❏ You may use any IDE to write your code. If your Transfer University is PSU or plan to do more CS classes here at PCC, check out the Special Note For PSU Transfer section in the Syllabus.
- ❏ Please bookmark the PCC Linux and Vim Manual - this will become a frequently used reference!
- ❏ Include **pseudocode** in part d of the design document.
- ❏ **Follow these Coding Construct Requirements:**
    - ❏ Your program must have function prototypes. Place the prototypes for your functions globally, after your #includes, just before main(). All functions must be implemented after main().
    - ❏ Each function must do only what it is supposed to do. For example, the menu function will simply display the menu, the readOption will take the option from the user, and so on.
    - ❏ **You may not use a while true loop and break statements in the loop.**
    - ❏ **Your program must do all data validation mentioned in the functions.**
    - ❏ Try not to have any redundant code (repeated code) in your program. That is the purpose of functions.
    - ❏ When you display the scores, it should be formatted appropriately with one decimal place.
    - ❏ Use constants where appropriate (for discount percentages).
    - ❏ **Do not use structs or any vectors for this program. Use only the concepts and functions we have learned so far.**
    - ❏ Print a welcome and goodbye message.
- ❏ Please open and compare your work with the grading rubric before submitting.
- ❏ Remember to follow all style guidelines.
- ❏ Download your Algorithmic Design Document as a PDF (File -> Download -> PDF), rename it to `a03.pdf`, and upload it to the D2L assignment by **Wednesday**.
- ❏ Upload your `a03.cpp` C++ source file to the D2L assignment by **Sunday**.
- ❏ Do your own work. Consult the syllabus for more information about academic integrity.

Additional Support

- ❏ Post a question for the instructor in the Ask Questions! area of the Course Lobby.
- ❏ To see how the Selection Card Sort Algorithm works, watch this video from Virginia Tech.
- ❏ Try this Selection Sort Animation by Y. Daniel Liang.
- ❏ Watch this Python Video to help you with the sorting algorithm.