



# CSCI 2270 – Data Structures Fall 2021

*Instructor: Asa Ashraf*

## Assignment 1

Due Sunday, September 5, 11:59 PM MST

## C++ FUNDAMENTALS

### OBJECTIVES

1. Read-in command line arguments
2. Read a file
3. Loop through an array
4. Split a string
5. Create an array of struct types
6. Read and Write to a file

### Instructions

Write code to complete **Problems 1 and 2**. Implement each of the problems as separate programs. To receive credit for your code, you will need to paste your solution into the **Canvas autograder (CodeRunner)**. Your code needs to compile and pass the given test cases in order to receive points. Note that the autograder submissions are divided into separate parts, so that you will be required to test the individual functions prior to testing the entire program.

### Problem 1

**Overview:** You will write a program that reads up to 100 numbers from a file. As you read the numbers, insert them into an array in **ascending order**.

#### **Specifications:**

##### **1A. Write a function called `addToArrayAsc`.**

- i. It should take **three** arguments -
  - a. `sortedArray[]` : *sorted* array that should be able to hold at most 100 **floats**.
  - b. `numElements` : the number of elements inserted so far.
  - c. `newValue` : the incoming float value to be inserted into the `sortedArray`.
- ii. The **`addToArrayAsc`** function should return a count of the elements inserted so far (i.e. the current size of the array).

The function header will appear as follows:

```
int addToArrayAsc(float sortedArray[], int numElements, float newValue);
```



# CSCI 2270 – Data Structures Fall 2021

*Instructor: Asa Ashraf*

**NOTE:** You should NOT use the in-built **sort** function provided by the `<algorithm>` header. However, you may write your own implementation of a sorting algorithm, such as Bubble Sort. The pseudocode for bubble sort (ascending order) is provided below:

```
function bubbleSort(A, N):  
    for i = 0 to N - 1 exclusive  
        for j = 0 to N-i-1 exclusive  
            if (A[j] > A[j + 1])  
                swap(A[j], A[j+1])  
            end-if  
        end-for  
    end-for
```

**1B. Write a complete program to do the following :**

**i. Reading the file:** Your program should take **a single command line argument** i.e. the name of the file where the numbers are present.

- This file needs to be stored in the same directory as your program.
- The file should store up to 100 numbers on separate lines. For testing you can use the file named **"numbers.txt"** on Canvas, or create your own if you prefer.

**ii. In the main function:**

- Create an array of floats to store at most 100 floats.
- Open the file that was passed via the command line
  - If there is any error in opening the file then print the below statement and terminate execution, `std::cout << "Failed to open the file." << std::endl;`
- Use the **getline** function to read the numbers one by one.
- Store these numbers in a sorted array by passing them to the **addToArrayAsc** function (you should use the code from part 1A).
- Each time a new number is read, print out the entire array after insertion, according to the sample shown below.

The Input and Output formats are shown as follows:

Input FileContents: arr.txt	Your Output:
1	1
6	1,6
2	1,2,6
12	1,2,6,12
5	1,2,5,6,12



# CSCI 2270 – Data Structures Fall 2021

*Instructor: Asa Ashraf*

## Problem 2

**Overview:** In this question, you will write a program that:

1. Reads a “.csv” file with 5 columns and up to 10 lines containing information on student grades.
2. Stores the information in an array of structs.
3. Writes the lines into the **output.csv** file where the overall student grade is between a particular range.
4. Prints the content of the entire array.

### **Specifics:**

Create an array that holds the **studentData struct objects**. Use the following struct declaration:

```
struct studentData {  
    string studentName;  
    int homework;  
    int recitation;  
    int quiz;  
    int exam;  
    double average;  
};
```

### **2A. Write a function named addStudentData:**

- i. The **addStudentData** function has the following declaration:

```
// length: Number of items currently stored in the array  
  
void addStudentData(studentData students[], string studentName, int  
homework, int recitation, int quiz, int exam, int length);
```

- ii. Instantiate a struct and store the **studentName**, **homework**, **recitation**, **quiz**, and **exam** values in it.
- iii. Assume equal weights amongst everything that needs to be graded. Take the average of the homework, recitation, quiz, and exam for each respective student and store it in the struct.
  - a.  $\text{student.average} = (\text{homework} + \text{recitation} + \text{quiz} + \text{exam}) / 4;$
- iv. Add the struct to the **students** array.



# CSCI 2270 – Data Structures Fall 2021

*Instructor: Asa Ashraf*

## 2B. Write a function named **calcLetter**:

- i. The **calcLetter** function has the following declaration:

```
char calcLetter(double avg);
```

- ii. Write IF-ELSE conditions to assign letter grades based on the following

- a.  $\geq 90$ : A
- b. 80-89.9: B
- c. 70-79.9: C
- d. 60-69.9: D
- e.  $< 60$ : F

- iii. Return the letter grade.

## 2C. Write a function named **printList**:

- i. The **printList** function has the following signature:

```
// length: Number of items in the array  
void printList(const studentData students[], int length);
```

- ii. Loop through the **students** array.

- iii. Print out each element of the **student** array in the following format.

```
"<student_name> earned <student_average> which is a(n) <letter_grade>"  
cout << students[index].studentName << " earned " << students[index].average << "  
which is a(n) " << calcLetter(students[index].average) << endl;
```

### Example:

"John Doe earned 95 which is a(n) A"

## 2D. Write a complete program which includes the following:

- I. The **struct** and **functions** listed above.

- II. A **main()** function defined as below:

1. Your **main()** should handle **four command line arguments**

- 1. the name of the **input** ".csv" file
- 2. the name of the **output** ".csv" file
- 3. a low bound letter grade
- 4. an upper bound letter grade

2. Input and output files need to be stored in the same directory as your program.



# CSCI 2270 – Data Structures Fall 2021

*Instructor: Asa Ashraf*

3. Read from the input file, “**students.csv**”:
  - a. Each line can be read and decomposed using *any* of the following:
    - **getline** function with the optional delimiter character.
    - **stringstream** object
    - **split()** function *you* wrote in CSCI 1300.
  - b. Following the decomposition, convert each entry into its appropriate data type. **Keep in mind that you must use the exact data types that are specified in the struct.** (HINT: If needed, use `stoi`, `stof` functions to convert from strings to numbers)
  - c. Call **addStudentData** function to update the **students** array.
4. Call the **printList** function after the array has been filled with data.
5. Writing out to external files in C++ can be done very similarly as reading in. Instead of using an object of the `ifstream` class, as is done with input streams, use an object of the `ofstream` class. A “csv” stands for comma separated values. Write into **output “.csv”** file:
  - a. Write the `<student_name>`, `<student_average>`, `<letter_grade>` (as shown in the sample output) of the students whose `<letter_grade>` is `>= lower_bound` and `<letter_grade> <= upper_bound` into the **output “.csv”** file. Here, **lower\_bound** and **upper\_bound** represent the letter grades passed as command-line arguments to the program.
  - b. **You should not sort them while writing to the output file.** The relative order will be the same as the input file. Only those students with overall letter grade within the bounds should be written to the output file.
6. Make sure your program closes the output file when it is finished writing.

## Sample Input and Output:

```
./a.out students.csv output.csv C A
```

**File Contents: students.csv**

Tim Thomas,90,92,95,100

Bob Brown,80,90,88,81

Jenny Jackson,69,79,90,73



# CSCI 2270 – Data Structures Fall 2021

*Instructor: Asa Ashraf*

Samantha Smith,65,57,72,59

Ralph Rogers,55,59,62,48

**Your print Output:**

Tim Thomas earned 94.25 which is a(n) A

Bob Brown earned 84.75 which is a(n) B

Jenny Jackson earned 77.75 which is a(n) C

Samantha Smith earned 63.25 which is a(n) D

Ralph Rogers earned 56 which is a(n) F

**Your output.csv file with grades between C and A should contain the following:**

Tim Thomas,94.25,A

Bob Brown,84.75,B

Jenny Jackson,77.75,C