CSCI 2270 – Data Structures

Instructors: Shayon Gupta, Ashutosh Trivedi, Maciej Zagrodzki

Assignment 1, January 2019

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. Pass by reference

Write code to complete the following problems. Each problem should be completed in its own file, as they will be graded individually on Moodle.

Problem 1

Overview: In this question you will write a program that reads up to 100 numbers from a file and stores them in a sorted array, then allow the user to see if a specific number has been stored.

Specifics:

1. Your program should take a single command line argument: the name of a file to read in.
    a. This file needs to be stored in the same directory as your program.
    b. The file should store up to 100 integers on their own lines, with no other text. You can use the file named *"numbers.txt"* on Moodle, or create your own if you prefer
    c. If there is any error in opening the file then print the below statement
    std::cout << "Failed to open the file." << std::endl;
2. Create an array of integers to store at least 100 integers.
3. Write a function called **insertIntoSortedArray**. It should take three arguments - a *sorted* array of numbers at least 100 long, the actual number of filled entries in that array, and a new value to be added to that array. It should then insert the new value into the correct spot so that the array remains sorted. It should return the new size of the array. This function can be declared the following way:

```
int insertIntoSortedArray(int myArray[], int numEntries, int newValue);
```

4. Back in the main function, open the file that was passed via the command line and use the **getline** function to read the integers one by one. Store these integers in a sorted array by passing them to the **insertIntoSortedArray** function described above.
5. Each time a number is read in, print out the whole array separated by commas. For example if the file contained the numbers 1, 6, 2, 12, and 5 in that order, it should output:

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

Problem 2

Overview: In this question you will write a program that reads a .csv file with up to 100 lines and stores the information in an array of structs and also writes the lines whose gpa is greater than a minimum value to a output .csv file. Each line of the input file corresponds to a struct element in the array. Then you will allow the user to print the contents of the array.

Specifics:

1. Your program should handle up to three command line arguments: the name of a .csv file to read, name of the output .csv file and a minimum gpa
    a. Input and output files need to be stored in the same directory as your program
    b. You can use the file named *"data.csv"* as an input file, on Moodle, or create your own if you prefer
    c. Each line in the files must follow the format *"<USERNAME>,<GPA>,<AGE>"*

2. Create an array that holds the User struct objects. Use the following struct declaration:

CSCI 2270 – Data Structures

Instructors: Shayon Gupta, Ashutosh Trivedi, Maciej Zagrodzki

Assignment 1, January 2019

```
struct User {
      string username;
      float gpa;
      int age;
};
```

3. Reading from the input file, "*data.csv*"
    a. Each line of the file can be read using **getline** function
    b. Parse each line using **stringstream** and convert each entry into its appropriate data type. USERNAME should be a string, GPA should be a float, and AGE should be an int. (Hint: Use **stoi, stof** functions to convert from strings to numbers)
4. Writing into a file
    a. Write the <USERNAME>,<GPA>,<AGE> of the students, whose <GPA> is more than the minimum gpa (read as a command line argument) into a .csv file

5. Write a function named **addUser** to perform the following operations:

    a. The addUser function has the following signature:

```
// length: Number of items currently stored in the array
Void addUser(User users[], string _username, float _gpa, int _age, int
length);
```

    b. Instantiate a struct and store the USERNAME, GPA, AGE values in it
    c. Add the struct to the users array
6. Write a function named **printList** to perform the following operations:

    a. printList function has the following signature:

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

    b. Loop through the populated array
    c. Print out each element of the list in the following format "<USERNAME> [<GPA>] age: <AGE>" using the below cout statement

std::*cout* << *user.username* << " [" << *user.gpa* << "] age: " <<*user.age* << *std::endl;*

Example, "Gupta [4] age: 25"

d. Call the **printList** function in main after the array has been filled with data

7. Be sure to close your file when you are done.

8. Functions **addUser**, **printList** and **main** need to be implemented in Moodle.

Testcase 1:

File Contents: data.csv

Acton,2.79,28
Gretchen,3.23,27
Regan,3.75,23
Rinah,2.78,22

Your Output:

Acton [2.79] age: 28
Gretchen [3.23] age: 27
Regan [3.75] age: 23
Rinah [2.78] age: 22