CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 1, Aug 2018

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
7. Create a class with constructors, getters and setters
8. Create header files with header guards

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 will need 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 *"assignment_1_problem_1.txt"* on Moodle, or create your own if you prefer.
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 and stores the information in a list of structs. Each line of the file corresponds to a struct element in the list. Then you will allow the user to print the contents of the list.

Specifics:

1. Your program should take a single command line argument: the name of a .csv file to read in.
    a. This file needs to be stored in the same directory as your program.
    b. Each line in the file must follow the format *"<USERNAME>,<GPA>,<AGE>".*
2. Reading from the file
    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. You can use the **stoi,** and **stof** functions to convert from strings to numbers.

CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 1, Aug 2018

3. Use the following struct declaration:

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

4. Write a function named **addUser** function to perform the following operations:
   a. Create a struct and store the USERNAME, GPA, AGE values in it.
   b. Add the struct to the vector.
   c. The addUser function has the following signature:

```
void addUser(vector<User> *users, string _username, float _gpa, int _age);
```

5. Write a function named **printList** function to perform the following operations:
   a. Loop through the populated list.
   b. Print out each element of the list in the following format:
      "<USERNAME> [<GPA>] age: <AGE>".
      For example, "Gupta [4] age: 25".
   c. printList function has the following signature:

```
void printList(const vector<User> users);
```

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

7. Only the functions **addUser** and **printList** needs to be implemented in Moodle.

8. Testcases:

Testcase 1:
File Contents: info.csv
Gupta, 4, 25
Alex, 4, 23
Your Output:
Gupta [4] age:25
Alex [4] age:23

CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 1, Aug 2018

Problem 3

Overview: You will be writing a program that defines a class to store movie information.

Specifics:

1. Define a class named **Movie**. It should store three pieces of information - the movie **title** as a string, the **year** it was released as an integer, and the **rating** from 0 to 10 as a float.
2. For each of the three members you created in step (1), write a function to *get* the value (accessor) and a function to *set* the value (mutator). The *accessor* function should take no arguments and return the member. The *mutator* function should take one argument representing the new value for that member, then set the member to be the new value without returning anything. These six *accessors* and *mutators* should look like this*:*

```
std::string Movie::getTitle();
void Movie::setTitle(std::string newTitle);

int Movie::getYear();
void Movie::setTitle(int newYear);

float Movie::getRating();
void Movie::setRating(float newRating);
```

3. Write two constructors for the class. The first one should take no arguments and set the title to “unknown”, the year to 2018, and the rating to 0.0. The second one should take three arguments representing the initial title, year, and rating, then set the class variables to be those arguments.
4. Have the program create three instances of the Movie class, then test all the methods in steps (2) and (3) at least once each.