CSCI 2270 – Data Structures

Instructors: Maciej Zagrodzki, Christopher Godley

Assignment 1

31, August 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 **Problems 1 and 2**. Implement each of the problems separately.

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.

Specifics:

1A. Write a function called **insertIntoSortedArray**.

i. It should take **three** arguments -

a. myArray[ ] : *sorted* array that should be able to hold at most 100 integers.

b. numEntries : the number of elements inserted so far.

c. newValue : the incoming value to be inserted into the sorted array (i.e. myArray[]).

ii. The **insertIntoSortedArray** function should return a count of the elements inserted so far (i.e. the current size of the array)

This function is defined as follows:

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

CSCI 2270 – Data Structures

Instructors: Maciej Zagrodzki, Christopher Godley

Assignment 1

31, August 2019

1 B. 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 integers are present.

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

ii. In the main function:

a. Create an array of integers to store at most 100 integers.
b. Open the file that was passed via the command line
    i. If there is any error in opening the file then print the below statement *std::cout << "Failed to open the file." << std::endl;*
c. Use the **getline** function to read the integers one by one.
d. Store these integers in a sorted array by passing them to the **insertIntoSortedArray** function (you can use the code from part 1A).
e. Each time a new number is read, print out the entire array after insertion.

The Input and Output formats are shown below:

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

CSCI 2270 – Data Structures

Instructors: Maciej Zagrodzki, Christopher Godley

Assignment 1

31, August 2019

Problem 2

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

a) Reads a **".csv"** file with up to 100 lines and columns containing information on national parks.
b) Stores the information in an array of structs.
c) Write the lines where the **area of the park** is greater than the minimum value into the **output .csv** file.
d) Prints the content of the entire array.

Specifics:

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

```
struct Park {
      string parkname;
      string state;
      int area;
};
```

2A. Write a function named **addPark:**

a. The **addPark** function has the following signature:

```
// length: Number of items currently stored in the array
void addPark(Park parks[], string parkname, string state, int area, int
length);
```

b. Instantiate a struct and store the **parkname, state**, **area** values in it.
c. Add the struct to the **parks** array.

2B. Write a function named **printList:**

b. The **printList** function has the following signature:

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

c. Loop through the **parks** array.
d. Print out each element of the **parks** array in the following format.
"<PARKNAME> [<STATE>] area: <AREA>" using the below cout statement

CSCI 2270 – Data Structures

Instructors: Maciej Zagrodzki, Christopher Godley

Assignment 1

31, August 2019

std::*cout << park.parkname <<" [" << park.state << "] area: "<< park.area << std::endl;*

Example, "Acadia National Park [ME] area: 47390"

2C. Write a **complete program** which includes the following**:**

I. The **park** struct and the **addPark, printList** functions coded above.

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

1. Your **main()** should handle three command line arguments: the name of the **input** "**.csv" file**, the name of the **output** "**.csv" file** and a **minimum area** respectively**.**
2. Input and output files need to be stored in the same directory as your program.
3. Read from the input file, "***park.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. **parkname** should be a string, **state** should be a string, and **area** should be an integer. *(Hint: Use **stoi, stof** functions to convert from strings to numbers)*
   c. Call **addPark** function to update the **parks** array.
4. Call the **printList** function after the array has been filled with data.
5. Write into **output ".csv"** file:
   a. Write the <parkname>, <state>, <area> of the parks, whose <area> is more than the **minimum_area** (read from command line) into the **output ".csv"** file.
6. Make sure you close the file when you are done.

Check next page for sample input and output.

CSCI 2270 – Data Structures

Instructors: Maciej Zagrodzki, Christopher Godley

Assignment 1

31, August 2019

Sample Input and Output:

Testcase 1:

File Contents: data.csv

Acadia National Park, ME, 47390
Arches National Park, UT, 76519
Badlands National Park, SD, 242756
Big Bend National Park, TX, 801163
Biscayne National Park, FL, 172924

Your print Output:

Acadia National Park [ME] area: 47390
Arches National Park [UT] area: 76519
Badlands National Park [SD] area: 242756
Big Bend National Park [TX] area: 801163
Biscayne National Park [FL] area: 172924

Your output.csv file with minimum area 200000 should contain the following:

Badlands National Park, SD, 242756
Big Bend National Park, TX, 801163