Abraham Meza

Christos Papachristos

CS 202 – 1001

February 05, 2019

Project 2 Report

**Description:**For this project, you are to create a program that will assist users who want to rent a car. You are given a **datafile with 5 different cars file** (the file is a priori known to have exactly 5 entries, each following the same data layout), and you must **read in all** of the **car data** from the **file** and **store** it in an **array of structs**. You must also **create a User Menu** with the functionality defined below. Although an example file is provided (Cars.txt), for grading purposes your project will be tested against a different test file that will not be provided to you beforehand. Our test file will be in the same format as the example file.

Continuing through Computer Science II (202), our instructor assigned us a project that demonstrates our abilities to access file contents and store the information in an array of structures that would be changed by the user in a specific way. Students had to write a program that extracts car data from a text file and utilize its content to create a Car Rental menu that allows the user to see the data of each car when renting. The menu options would include sorting the cars in ascending price, checking the availability of cars for renting, and exporting the changes to an output file. This project is an add on from our coding techniques from project 1 and allows the students to build off their previous work. Even so, it challenges students to create specified functions that would be compatible with structures, while also allowing students to review the concept of switch statements. As students tackle structures and switch statements, they are instructed to use several functions for each menu option to decrease the complexity of the main function.

For my design, I began to compile and code the given data for the project. First, I declared all the given information such as the string library prototypes, header files, and defined variables within the menu selection. Then I established a global structure and declared the array of structures that labeled and organized the data contained in the given input file. From there, I worked on what I knew. From my previous project, I copied and pasted my old string functions that fulfilled the role of comparison and copy (length, copy, compare). I then worked on the string concatenation function. With my experiences with pointers, it was not difficult to create a concatenation between strings since it was simply adding the characters from one string after the other. I simply scanned through each array element and copied the string until a “null” character was present and replaced the “null” character of the other string with the copied string. As I experimented with pointers, I realized that the copy function had to be compatible with pointers, so I simply converted my function by adding a few pointer variables as placeholders which would be returned instead of the established parameters.

After I had completed my string library functions, I brainstormed the menu options that would be available for the user. For each menu option, I created a separate function prototype to work on, separately, to decrease the complexity of the code and make it clean and organized for future tracing. After I established all menu options, I worked on each function definition. Considering the project was a more advanced version of the previous, I simply took my input and output file algorithms from the last project and separated them into their own functions. Despite having the same objective, I had to modify both of the file functions to incorporate an array of structures instead of a two dimensional array. In order to switch the formatting, I simply referenced the structure during each array element to match the data to their designated category. As a result of my past coding project, I automatically had 3 of the 7 options completed. After I finished my file functions, I began to work on my sorting algorithm. Since the menu should display the car data in ascending order based on price, I simply referenced the element of price within my array of structures and compared it to another through the use of a for loop. Depending on its relationship (a < b), it would execute a swap algorithm between the strings of data. A swap algorithm will always be the same as it involves the same three variables (temp, element1, element 2); however, it may be longer depending on how much information is needed to be swapped. In this case, my swap algorithm was longer than usual as it swapped every structure element (year, make, model, price, availability, etc.). After finishing my swap function, I utilized the same concept of checking a certain element for its condition in my options function. This function displayed the available cars for rent in ascending order based on price. In order to execute the task, I scanned through the array of structures and accessed the element of availability to check the condition of each car; if the condition was true, the car data would print to screen. In order to ensure the cars printed in ascending order, I had to call my sort function. I simply modified the data to only display the car, its availability and the estimated cost (cost of the car per day multiplied by the days indicated by the user). After finishing my options function, I worked on the final function by utilizing the same concepts in the previous functions: I simply scanned through the availability elements of the data and allowed the user to rent it if the condition was true. If the rent was successful, it displayed the message on the terminal and the total cost of the car (car price per day multiplied the days indicated by the user). Then I created a modification within the array that changed the availability condition to false for future access.

After the functions were finished and compiled correctly, I began to work on the main section of the program. Since all of the work and program functionality was within the multitude of function definitions, I simply had to create the menu interface. I simply created a key that indicated which option the user should choose for a specified task and connected it to a switch statement. Depending on the users input, it would call the function related to the task in their designated cases. As a result of working on the important components outside of the main, the main was short and simple. Another benefit of having several call functions, is the ability to debug easily. If an option would not work while running the program, we could easily pinpoint where the error is occurring. To end off the program, I added another switch case that would return 0 when indicated, in order to exit the program. Afterwards, I modified the program by adding confirmation statements to each function and made it cleaner through “endl” spacing.

Throughout the project, I ran into little complications. My design is identical to the previous project since abilities required within the project were very similar. The errors that I would frequently encounter were simple mistakes such as semicolons and loop organization. Though the project was not too complicated, I was confused when attempting to tie the string library functions to the performance of the project. The program could achieve proper functionality with only the string copy function. I did not have to use any of the other functions to display the correct data: the function of myStringCompare returns a comparison between characters, while myStringLength returns an integer value of the length of the string. Even myStringCat was not needed in this project since it added a string to another, which was not what the program asked for. Generally speaking, the only needed function was myStringCopy since we had to sort the data in a certain manner.

In the projects current state, the user has to be aware of their choices since the changes are not displayed automatically. After every change, the user must print the new data by accessing option 2 every time. If I were to have more time with the project, I would improve the assignment by always calling a print function after every change to allow the user save time, energy, and to prevent confusion.

All in all, the project was very efficient in teaching students about structures and how to access/change the data within the structure. In addition, the assignment allowed the students to review their knowledge of pointers and switch statement. Personally, I enjoyed how the project extended previous knowledge since it shows us how different codes rely on one another for specific executions.