Abraham Meza

Christos Papachristos

CS 202 – 1001

February 13, 2019

Project 3 Report

**Description:**

This project will expand Project 2 by adding additional functionality, using pointers, and implementing abstract data types (ADTs) through classes. **Pointers must be used for all array manipulation**, including arrays with ADTs (structs, classes) e.g, rental cars, rental agencies. **Pointers must be used in function prototypes and function parameter lists** - not square brackets. Make sure all your C-string functions (e.g. string copy, string compare, etc.) work with pointers (parameters list and function implementation). Square brackets are to be used only when declaring an array variable. **Remember: pointer arithmetic** (e.g., ++ , +=, - -, -=) and **setting the pointer back to the base address** using the array name **can be used to move through arrays**. All pointers must be passed by value. (*Note*: Try to also get accustomed to using the arrow operator (->) with Class Object pointers for member access if you use such in your code.)

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 class objects that would be accessed and manipulated by utilizing pointer arithmetic. Unlike the previous projects, we are not allowed to scan through array elements without the use of pointers and are to use class operators to access each object. Students had to write a program that extracts car data of multiple agencies from a text file and utilizes its content to create a Car Rental menu that allows the user to see the car data of each agency when renting. The menu options would include rental estimations, to which car is the most expensive to rent. This project is an add on from our coding techniques in our previous projects, while also learning the properties of classes and pointer arithmetic. It challenges students to utilize their knowledge in pointers to transfer data between objects and structures, by utilizing loops and functions. In addition, the project continues to polish the students’ use of switch statements when creating a user menu. As students tackle on classes and pointers, they are instructed to use several functions for each menu option to decrease the complexity of the main function.

For my design, I began with copying my string functions from previous projects and labeling any constant variables as a define argument in order to make the code easier to change and navigate. From there, I began to create my class and structure skeletons for the agencies and their corresponding car data. Given the instructions, I declared all of the agency information within the structure and placed the categories of the car data within the private group of the class. From there I created a default and parameterized constructor that initialized all of the variables to 0 or null, depending on its data type, and connected the private data to public. In order to link the private data to the changeable public, I needed to assigned them temporary placeholders that would override their information with functions such as myStringCopy. For every variable I had in my private class, I made a getter and a setter that either returned the private variable for its value or assigned it a value through another when storing data. Once I finished making the basic skeleton of the class, I integrated a print and estimated function that would manipulate the class with ease when needing to obtain certain information.

After I had completed the basic foundation of the code through the instructions provided, I began to plan out my menu selections and the role each case would have. As a result, I added five different function prototypes that would organize the data based on the user’s preference. In order to make it easier, I created a menu selection first in order to have a reference of what each function definition should include. Once I finished, I began working on the function definitions. Due to my dull skills in pointer arithmetic and inexperience with classes, I found it very hard to write an algorithm to scan data without utilizing bracket notation and array scanning. However, once I learned how to utilize the arrow operator, the process of accessing class objects became much easier. Just as the previous projects, the file implementation stayed the same; however, the “for” loops used to scan through the file information changed as the parameter of the functions became pointers. Instead of scanning through with brackets, I had to utilize a pointer variable (aptr) to keep track of my agencies, while another one was used (rptr) to track the car options. In order to bypass the private barrier that held the cars’ data, I created temp variables within the function to pass the file’s value. For every variable, there was a temporary placeholder and a setter that was called for variable assignment. After some trial and error of pointer arithmetic, I was finally able to create a functional loop that scanned the text file and assigned the data accordingly. After the input function was created, the others were easy to structure as they all utilized the data in similar ways. For my print function, I simply scanned through the data the same way and indexed every agency car for convenience. The difference between the read function and the print function was: one was utilizing “inputStream” to read the data and the other “cout” to print the data read. I created my estimate function to utilize the same scanning process as it took the user input into the amount of times the loop scanned for their desired car/agency. However, once I printed the results, they were somehow ahead of the desired index by one, so I simply added a decrement argument to the pointers beforehand to fix the issue. Next I worked on my expensive function. Though it held the same process of scanning through the data, I had to figure out how to always keep track of the most expensive as the data changes. In order to keep track of the price and the data grouped with it, I created another pointer that would keep the position and a temporary expensive float that would change to the scanned price if was less than the value. Once the loop iterations were complete, the most expensive car was printed to the screen. Finally, I worked on the availability function. The function was relatively similar to the expensive function as it relied on an if statement to execute the correct output. For this function, I simply utilized the same output-file layout from the previous projects and modified it to match the input function. In order to prevent all the data from transferring, I simply integrated an if statement to check the bool flag for availability in order to indicate its true/false. If the car was available, its data would be copied to a new file.

After the functions were finished and compiled correctly, I simply added the appropriate function calls to my menu within the main section of the program. Once the program functioned correctly, I simply went back and made the code cleaner through comments and spacing (e.g. “endl”, etc.). Generally speaking, I ran into little complications and errors. Though my design is identical to the previous projects we’ve had this semester, the implementation of classes and pointer arithmetic made me think harder about the algorithms needed for proper functionality. Usually I would stick in my comfort zone of bracket manipulation when conducting these kinds of projects; however, this allowed me to gain more expertise in other ways of implementation. Even though I take my projects step-by-step, I learned that planning is very important when it comes to coding since it lays out a pseudocode that can help you devise an algorithm, while also saving time/effort.

If I were to have more time with the project, I would improve its functionality by adding more elements of convenience and by adding conditional statements to the file functions to prevent users from inputting a non-existent file.

All in all, the project was very efficient in teaching students about class manipulation and pointer arithmetic. Personally, I enjoyed being challenged to step outside of my comfort zone since it involved more time and an in-depth thought process.