Juhwan Lee

Professor Karla Fant

CS202

December 10th 2020

Term Paper

Originally, my major was mechanical engineering, but I changed my major because the computer science major seemed very attractive. Not only that, it is because I thought computer science would have a better prospect than mechanical engineering in the future. As a result, I am currently working on the last assignment of CS202, and I think changing my major was a very good decision. I love my major and look forward to future subjects. Below is my computer science journey so far.

I started with CS161 because I didn't have any experience and knowledge in coding. In CS161, I learned a little bit about a programming language called Python, and I gained some experience in coding and learned basic knowledge to get started with CS162. However, the CS162 felt much more difficult than the CS161. CS161 didn't require much practice, but in CS162, we first started to learn about C++ and it was much more professional and it required a lot of coding practice. I didn't practice enough and, as a result, I got PW(pass with warning) in the first proficiency demo. After that, I felt that I had to practice harder and more, and I practiced coding really hard to get good results in the next proficiency demo. As a result of my hard work, I got P (proficient) in the second proficiency demo. When I first got PW, I thought a lot that computer science might not be the right major for me, but after I got the P, my thoughts changed. I gained confidence in coding. It wasn't smooth at first, but as time passed, things started to make sense and I was able to finish the CS162 well.

And I took CS163 right after CS162. CS162 was about basic C++ syntax, while CS163 was about data structures. In CS162 we only learned about linear linked list, but in CS163 we learned about various data structures. For example, what algorithms the data structures have, what format should be coded, what is the data abstraction, what is the client program, etc. Each data structure has different characteristics, and it was very interesting to find out which data structure to use depending on the situation. And while doing the CS163 program assignment, I was able to master the basic C++ syntax I learned in CS162. However, in CS163, Recursion was not required. At that time I still needed more practice with Recursion and most of my data structure programs used iteration. Because I used the iteration method in the programming assignments, I had to practice a lot of recursion for proficiency demo. I practiced enough, and fortunately I got a P (proficient) in both proficiency demos. Therefore, CS163 was also successful.

And I started taking CS202 this fall term. Not long after the fall term started, I faced great challenges. First of all, I felt the difference between CS202 and CS163 is somewhat similar to CS162 and CS161. In other words, it was much more difficult than CS163. The biggest difference between CS202 and CS163 is that in CS202, all the programs have to be object oriented. Object oriented programming is the biggest topic that we learn in CS202 and is the most important topic. At first, it was hard to understand what object oriented programming is and I had no idea how to make my first program to be object oriented. Moreover, in CS202, it was required to use only classes not structs. This was a challenge and it came to me really difficult especially when I implemented node class. Another challenge was that all the data structures have to be implemented recursively. After watching a few lecture videos I dived into my first programming assignment.

The first programming assignment was very difficult. It was so different from the programming approach that I've been using so far that it took a lot of time to complete the program. I thought I understood what object-oriented programming is after watching several

videos, but when I actually programmed it, I realized that I didn't fully understand it yet. I concentrated on forming a single inheritance, creating relief efforts class and three sub-classes (ex. housing class, food class, clothing class), but the node class and circular linked list class became very complex and inefficient structures. Both the circular linked list and linear linked list were programmed to use recursion, but I could not figure out how to unify a node class into one (ex. relief_item * data), so I decided to divide it into three different node classes (ex housing data, food data, clothing data), and it caused the circular linked list class to have so many functions.

The second programming assignment was about making a program that manages three different communication softwares with the ability of management of all the different types of communication software accounts. There are three communication softwares in this program: discord, slack, and zoom. Communication class is a base class and discord, slack, and zoom are derived classes. All the different types of communication software accounts are stored in one linear linked list and all the messages are stored in an array of linear linked lists. Depending on the communication software and message type, it all goes to a different index. Discord has one unique function and this function can be executed by using RTTI(run time type identification). RTTI is required when performing a unique derived class function when there is no virtual function in the base class. Another technique that was required in order to build the second program is dynamic binding. Dynamic binding is a late binding or run time binding. With dynamic binding, the compiler waits and sees what the base class pointer is actually pointing at and depending on the class object that the base class pointer is pointing, it calls the appropriate function not the base class function. With dynamic binding, I was able to store all the different types of communication software accounts in one linear linked list and I was able to call the right functions without micromanaging the different data types. The problem that I had when I was implementing the first program was that I could not find a way to store all the different derived class objects into one node class and call the right function. This was solved in the second program by using dynamic binding.

Third programming assignment was to create a program that uses operator overloading. The purpose was to create a monster battle using Operator Overloading. There are four monster options, and users can create the monster they want, and choose a monster to compete with their opponents. The types of Halloween monsters I have chosen are vampire, witch, werewolf, and devil. The user can create one of the four and as it is created it is stored in the binary search tree. And after creating a monster, the user can choose one of the monsters stored in the binary search tree and battle the opponent with the monster. Operator overloading was used in binary search tree functions (especially insert function), and in the monster battle part of the client program, and dynamic casting was used a lot in the array of doubly linked lists. For example, it was used in the part to find out the user's monster type in the wrapper function because the array to be accessed is different according to the user's monster type. Operator overloading allows us to use the operator as a built-in data type for our class object.

Unlike previous programs, program 4 was built in Java rather than C++, and there were many differences. First of all, the biggest difference is that there is no destructor. One of the most difficult parts when programming in C++ was memory management. Because Java has a garbage collector, it makes programming easier because there is no need to manage memory. However, when building the data structure, it was more difficult than building it in C++. Because there is no pass by reference in Java, there was a lot of difficulty because recursive functions had to be built using pass by value. However, with a lot of trial and effort, it was possible to find a way and recursive functions could be built using pass by value. The main feature of program 4 is that when media is uploaded, it is saved in the playlist, and the user can access the media that the user wants and perform various actions. If the user is a professor (administrator), media can be uploaded, modified, and deleted, and functions such as adding a quiz, deleting a quiz, adding a comment, deleting a comment, etc. can be performed by moving to the desired media. And if the user is a student, the user can go to

the desired media and perform functions such as adding a comment, deleting a comment, and taking a quiz. Java programming and C++ programming are similar but different. While programming, I could feel the strengths and weaknesses of each. Program 5 is an extension of Program 4.

Program 5 is an extension of program 4.Program 4 is a program for one playlist, and Program 5 has another data structure that manages multiple playlists. The data structure that manages multiple playlists is one of the balanced trees, the AVL tree. In program 3, I implemented a general binary search tree, not a balanced tree, so I had to implement a balanced tree in this program. The overall design is the same as in program 4, but the difference is that another node class has been added and the AVL tree class. Program 4 has only one data file, but program 5 has multiple data files. Because there are multiple playlists. One data file means one playlist (subject). Therefore, the first time the data file is loaded, the file for each subject is loaded. And if the user is a professor (administrator), the user can add a playlist (subject). It was easier than starting from nothing because program 5 continued on top of program 4, but it was very difficult to implement a balanced tree. The implementation is similar to the normal binary search tree, but it was difficult to keep the balance.

In conclusion, all the things I thought were challenges such as using recursion, not using struct, and being object oriented were actually very important when it comes to mastering not just C++ but also Java. Moreover, comparing the last programming assignment with the first programming assignment, I can see that my coding skills have gotten a lot better. It is more object oriented and it is more organized and clean. All in all, CS202 was full of excitement and I look forward to what to come next.