Evan Hedges

CIS 3190: Software for Legacy Systems

Professor Michael Wirth

This assignment demonstrated new languages are created and how they make coding easier. The assignment was updating a COBOL program which would tell you if a number was prime or not. The beginning of the assignment was very difficult. I had read over the COBOL lectures multiple times and found the program difficult to understand. This was due to the different sections and divisions within the program. I used examples from online and the text to begin the first part of the assignment. Once I understood how files in COBOL worked, it did not take very long to update the code and input the relevant environment to run the code. Afterwards, I made a copy and procced onto the second task, updating the code. Initially, understand the code seemed easy. I began by changing the formatting and changing variables names to increase readability. This was due to make tracing through the file easier. I took and outside-in approach, beginning by removing the tags in descending order. This was done to ensure that I always had working and compiling code throughout the assignment. Since you cannot have a period within a block, it made no sense to try an remove all of them at once. Stages would allow me to continuously test working code. I saw that GO TO 1 was really a loop around the entire program, so that made sense to leave to last. I realized that GO TO 3 and GO TO B2 were if statements so I began by removing and condensing that code. That was initially easy. However, GO TO 2 was difficult. I understood that it was a loop, however, it was difficult to figure out how exactly the program was supposed to exit. I proceeded to ensure that all the code involved with that loop were in if statements to prevent. Then I realized that the initial requirement to loop was the two values were not equal. Furthermore, I realized that if statement with the GO TO 2 was reverse it would create the end condition. Once this was completed, I was able to remove the GO TO B1 statement. This left me with the only GO TO 1 left. I began by rereading the code, ensuring that all conditions was inside an if statement. I then converted the remaining GO TO into a loop. However, this caused the program to double up on the last entry. I fixed this by moving the input to the end of the loop and copying the read in statement outside of the loop. This completed the updates to the code. Afterwards, I viewed Michael Wirth's, The Craft of Coding blog to implement task 3. I used his code to create the dynamic file names.

Throughout the program, I always referred back to the first task's output. This was done to ensure that if a change was made, I would be able to test against a correct set of output. This allowed me to ensure that I would always have the correct output in case I caused an error, or had some other problem.

One thing I liked bout COBOL was the file access. Although figuring out how to set up the file access was a challenge, once I understood how it functioned I realized that the actual code was too difficult. The environments were difficult to understand. In order to get the program working, you had to include the environments. This makes the initial learning curve difficult because in other languages, you have a main then program. It was also confusing to try and figure out which values were used for reading in, and how those functioned with file I/O. However, reading in the data seemed very simple compared to C. In C you require memory allocation, multiple functions, allocations and frees into order to get data in. However, with COBOL, the file was essentially treated like a variable, and it seems easy to manipulate once you have the system setup. The move commands and other outdated formats were not difficult to understand. Overall, some of the structures of COBOL was similar to assembly, which is a familiar language to me. This allowed me to easily learn COBOL once the environments were set up.

This assignment enlightened me to how different languages can be. The program structure with the different division was very difficult to understand and took multiple readings to comprehend. The use of periods was also difficult to follow. At first I did not understand that the periods did not go inside of blocks. In other languages, such as C every line ended with a semicolon. Even for other languages that do not require and end delimiter, such as Javascript, I always used one to ensure that the lines ended properly. This made programming in COBOL annoying due because either I would enter a period to end the line, or would copy a line and not remove the period. Furthermore, for this assignment the bulk of the code is inside of if statements or loops. It seems confusing to require that each line must end in periods, yet not have that for inside logic blocks.

Overall, the initial difficulties of the assignment was understanding the basics of COBOL as stated above. Afterwards, the remainder of the assignment was fairly simple. Some of the format, such as move, had similarities to assembly and other older languages. The actual body of programming was not that difficult.

The assignment would have been easier in other languages. C has the ability to create a loop that continues until the end of file. This differs from COBOL which requires a Boolean to indicate whether or not the actual file was empty. This would have made the assignment easier to do. Furthermore, built in functions such as modulo would make calculating the simpler. This would reduce the size of the program and make it easier to read. Furthermore, the ability to use the break command would have been helpful. Although not necessary for the program, it would have been more convenient to break out of the loop when done.