Daniel Cody

Alan Labouseur

28 January 2014

Theory of Programming Languages

Date

1/28/14

I spent 3 hours today working on the Fortran program. At first for the first hour I spent just trying to compile the language as well as look at the basic data types. After I set up my git I tested how to manipulate strings and other types. I realized that strings are terrible in Fortran. They are just an array of characters which is not as easy as I would like to work with. Instead of index’s you need to substring them to get a character to work with. It took me about 45 minutes to work on the algorithm and such for the functions, but I struggled to insert the characters one grabbed from the string back into the string (Array of Characters). After a lot of research I figured out a pretty simple way to do it, which was frustrating because of how simple it was and how hard it was to find the answer.

Overall I liked Fortran for this assignment. I would imagine from the small parts I used it, it would be hard to write a scalable program in this language but I could see how since I feel you have much control over the language it could be easy to do certain things. It reminded me of sML I do not know if that was because of how I wrote it in subroutines and that is sort of how functions are in sML, but I did not hate Fortran once I began to learn how to use it. The readability of it is pretty simple, especially to a programmer. To a non-programmer certain things like substring might look weird. I enjoyed the start and end instead of {} because it makes it easier to read in my opinion.

1/31/14

I spent a few hours today trying to research COBOL, with very little luck. I had an extremely difficult time trying to find good sources of code to examine to help myself with trying to declare variables as well as understand what limitations the language has. I eventually got to the point of declaring variables and making a loop. I grew to understand what exactly I was doing and how I could do things. I wrote the loops needed to encode, decode, and solve the string. The next issue I faced was making it its own function or subroutine.

2/3/14

Today I spent about an hour trying to research how to tackle the subroutines, in COBOL, which I needed to make each their own function. After a lot of time, I found a good source of code I could read to help myself with this issue. I then after a few minutes separated the code into three routines, and fixed up my code. For some reason comments give me errors with the online compiler. Also I realized I had a bug in the way I declared one of my variables. I only had PIC 99 so the digits in that integer did not go into the hundreds column so I needed to add another 9 which limited the way the alphabet looped around since when you add 26 to some letters it gets above 100 which caused an issue because before this it went back to 0. Overall I did not mind the language once I got around to seeing how it worked. If there was better documentation or resources online this part would not have been so bad. The language is pretty straight forward once you read it. I like certain aspects like explicitly stating end as well as display instead of like System.out.println in java. When to use periods and when not to is very confusing as well as I do not like the MOVE keyword instead of using =. I am used to the = vs == so it does not bother me but I have begun to look into Pascal and I like the := operator to setting values to a variable.

I continued to work today and finished the Pascal. It was by far the easiest of the three I have done so far. It reminds me a lot sML. The var to declare things before the main part of the method is like sML and the way it feels when writing it feels similar as well. There was a lot of good resources online to help which made it very easy. It maybe took me over an hour to write, since I know the algorithm so well now. All it really takes is the ability declare data types and variables, loop, substring a string, change the character to ASCII then change it back and re-adding it to a string. Once I figure out how to do those tasks the rest of the program is extremely easy. This was not a bad language to learn, I really enjoyed it.

2/6/14

Today I did Scala procedural. It was not hard at all, it had a lot of documentation online as well as it was easy to compile. It is very similar to java, just a little different syntax wise. I enjoyed how easy it was and how there was type casting. It was a little bit of an issue going from chars to strings, but after figuring that out it was fairly simple. This was my favorite language that I have done so far for this assignment. I think it was because it reminded me of java, which is my favorite language.

I really like how I am going to write this project in a functional style as well because it will make me look at the same problem from a different style of coding as well as doing it in the same language another way. I am excited to try it in Scala because of how much I like the language already and that I will be able to do it in another style of coding. I am curious as to how Scala would handle a part functional program as well as a procedural program. I would like to be able to take advantage of both styles of programming as well as combine them.

There was certain things I really liked about each programming language and some things I did not like. There was not much I liked about COBOL, it was hard to read, write, and did not have a lot of built in functionality which I wish it had. I really like how in Pascal there is a difference between assigning a value and checking for equality. Instead of using =, ==, or === like in other languages the = versus := was very covenant. I enjoyed how readable Fortran was as well as the ends instead of the {}. Scala was my favorite since it was so much like java and so cleanly written. I liked how they declared the variables types and setting them equal to a value. Also I am very interested to see how it handles functional programming.