Programming in the Past

CMPT 331 - Spring 2023 | Dr. Labouseur

Josh Seligman | joshua.seligman1@marist.edu

January 19, 2023

1 Log

1.1 Predicton

I am predicting that it will take me around **20 hours (average of 4 hours per programming language)** for me to learn Fortran, COBOL, BASIC, Pascal, and Scala and build a Caesar cipher in each language. This is an extremely rough estimate as I have never used any of these programming languages before and will have to learn each of them starting with "hello world." I am sure that I will have moments of staring at my computer screen for extended periods of time due to a weird nuance or gimmick in at least one of these programming languages that is not common in more modern languagues. However, despite my lack of familiarity with these languages, I am hoping that there will be some nice similarities between them, so my approach to writitng the Caesar cipher does not drastically change between them, and each implementation can easily be compared to each other on an even playing field.

1.2 Progress Log

Date	Hours Spent	Tasks / Accomplishments / Issues / Thoughts
January 18	3 hours	I built the entire Caesar cipher in Fortran.

1.3 Final Results and Analysis

2 Commentary

2.1 Fortran

2.1.1 My Thoughts

Building the Caesar cipher in Fortran was a challenging experience due to its strict rules regarding code organization and how functions and subroutines work. First, regarding code organization, I did not like having to declare all used variables at the top of a subroutine. In many other languages, the purpose of variables can oftentimes be inferred by the location in which they are declared. For instance, a variable declared by a loop often means it will have an important role in the loop, usually as a loop increment variable. However, in Fortran, since the variables were declared at the top of each subroutine and program, the readability of the language is hurt because the variables are all clumped together, and the language is tougher to write because you have to be careful in making sure that comments are written to explain what each variable will be used for. In the end, this organization did clean up the rest of the subroutine, but I would have rather declared the variables in more logical spots to go along with the flow of the program.

Another issue that I had with Fortran was with functions and returning values. I initially wanted to use functions to take in the orginal string and return the output string for encrypt and decrypt. Hewever, functinos and character objects were really not working nice as I received a bunch of errors, for which Google searches were not able to help me as the language has not been widely used since the inception of the internet. Some of these errors included but were not limited to an "entity with assumed character length at (1) must be a dummy argument or a PARAMETER" and functions that were explicitly marked to return a character (string) were returning a REAL and caused type mismatches at compile time. In the end, the best resource on the internet was the quick start guide on Fortran's official website, whose advanced examples provided me with some inspiration to use a subroutine and pass in the variable that would be modified and used as an output. Additionally, I wanted to make the function/subroutine support characters of an unknown length to work with all possible inputs. However, despite a lot of forums online having examples with this as the situation, the code just never compiled or worked for me. Therefore, I chose to have one of the parameters of my subroutines to be the length of the string, so the variable declarations at the top of the subroutine would be able to provide a known length.

Overall, aside from the variables being declared up top, Fortran is quite a clean programming language in terms of readability as a lot of the control structures are similar to those in more modern languages. However, the lack of clarity for how functions work with strings is extremely frustrating and significantly hurts the writability of Fortran as I had to find a way to finnesse a solution using subroutines and pass in the output variable, even though I would have preferred to have better organized code that returns a result from a function.

2.1.2 Google Search History

- fortran hello world
- function in fortran
- string type in fortran
- get length of string in fortran
- Entity at (1) has a deferred type parameter and requires either the POINTER or ALLOCATABLE attribute
- pass in character to a function fortran
- Entity with assumed character length at (1) must be a dummy argument or a PARAMETER
- return string from function fortran
- iterate through characters in a string fortran
- fortran print string to include value of variable

2.2 COBOL

2.3 BASIC

2.4 Pascal

2.5 Procedural Scala

3 Conclusion