Assignment 3 – Roman Numeral Converter Mar. 24, 2018

Reflection Report
Mohammadamin Sheikhtaheri
0930853

Working with COBOL was a very new and interesting experience for me. COBOL is very different from the programming languages of the likes of C or Java which I am used to, writing code in this language was very weird for me because the syntax is much closer language to the English language. For example, in C we usually do “variable != 0” if we want to check whether or not the variable is 0, however in COBOL, we have to write “variable is not equal to zero.” This made it quite difficult for me because it felt like I was writing an essay, which I hate.

Just like the Fortran or Ada assignments, one of the biggest challenges was getting used to the syntax of COBOL, but even after that, I am still not comfortable writing code in COBOL. The way the records are set up and how the program is separated into different sections was quite confusing and difficult to re-engineer. Re-structuring the program to exclude go to statements was the easiest part of the re-engineering process since we have already done this in the past two assignments, all that needed to be done was making the program go from top to bottom without any unconditional jumps.

To separate the program into two portions (file input, keyboard input), I had to use evaluate statements, and performed different sections of the program depending on the input of the user. For the keyboard input section, I modified the program so that it would send the length of the inputted roman number to the “conv” function instead of just calculating the first letter of the input. This allowed me to index through the inputted roman number without the user having to press Enter every single time. For the file input section, I opened the file with a dynamic name that the user inputs, and run the same algorithm until the end of the file.

Overall, I believe COBOL is a very interesting and different legacy language and this was a very great opportunity to learn more about it. The compiler was very similar to the gcc that we are all used to, which made it easy to debug, unfortunately it also had the hated “segmentation fault” equivalent which does not tell you anything and you have to find the error on your own. COBOL is a very difficult language to learn because of how different it is compared to other languages, which is why I disliked it more than I liked it, but this assignment was still an awesome experience.

STEP BY STEP PROCESS & INSTRUCTIONS ON NEXT PAGE

Step by Step Process of Algorithm Design and Re-Engineering:

1. Researched the language of COBOL in general, specifically the syntax and program structure

2. Studied how the existing conversion algorithm worked and how the 2 files communicated with one another

3. Researched what each division of the COBOL program structure was used for

4. Modified the algorithm to work with lowercase letters as well as uppercase letters

5. Removed all of the unconditional jump statements that made the program very unreadable

6. Used evaluate statements to run different portions of the programs based on the input of the user

7. Modified the program so it wouldn't require the user to press Enter after every letter of a Roman number

8. Studied how files with user defined names were accessed (thanks to Professor Wirth's blog)

9. Made the program more usable by prompting the user for what they should input, and also made the output more clear

HOW TO RUN THE PROGRAM:

Compilation: **cobc -x -free -Wall romanA3_1.cob conv.cob**

Execution: **./romanA3_1**

Runtime Use:

File Input: **1)** Press '1' when prompted
2) Enter the file name of the file of roman numerals you wish to convert

Keyboard Input: **1)** Press '2' when prompted
2) Input the roman number you wish to convert using your keyboard
(You may use lowercase letters, uppercase letters, or both if you wish)