Braelyn Rotman
ID: 1006740

Assignment 3: Cobol Re-Engineering

Reflection Report

My Re-engineering Process:

1. I spent the most time reading through the original program to figure out what was happening. I had to use a lot of display statements to help me trace through the program

2. The first thing I changed was the go to statements. Since these were the major feature of the original algorithm I thought it was important to eliminate them first.

3. After the go to statements were removed and the program functioned correctly it did not take long to update the calculation statements and add ends to all of the structures.

4. The last thing I did was remove the file input to take input from a user. The program will calculate the square root of numbers until the user enters 0. For this step I also had to change the output formatting.

Modernizing Structures and Features:

Instead of using go to statements to call other functions I used perform. To repeat the same function, instead of using go to I created a loop that would originally exit when the end of the file was reached, and later when a 0 was entered by the user. Instead of using write statements, I opted for display statements as I found focussing on formatting one thing at a time was a bit easier.

Algorithmic History:

The Babylonian method is one of the oldest algorithms for finding a square root. The idea of this method is to start with a random positive number and then repeatedly calculate a new number closer and closer to the square root each time. This converging process is repeated until the required accuracy is reached. This method was discovered in modern day from clay tablets that were used to calculate the square root of 2. On the tablet used to determine the formula was a square with sides of length 0.5. Side note that they used a base 60 number system so 0.5 would have looked like 30. The square shown was cut into triangles and the diagonal side length was calculated by multiplying the side length by the square root of 2 to get the length of the diagonal side. This is similar to the Pythagorean theorem. Its amazing how accurate these calculations were done without the simple calculators we have today.

What was my overall experience with Cobol?

Overall, I had a poor experience with Cobol. For such an old and widely used language there was so little information online for those new to the language. Cobol felt foreign and I don't think my previous programming experience helped much other than with logical program flow.

Was Cobol easy to learn?

I thought Cobol was very hard to learn. I had a hard time finding tutorials online and most of the examples I could find were buried in very old forums. This really surprised me because even though it's a legacy language it is still commonly maintained on older software. I also found trying to read through and learn then older code very confusing. The storage and output formatting was unlike any other language I've seen and took me a long time to figure out. Reading through the variable declarations was actually harder than the algorithm itself.

What did I find challenging?

One thing I really struggled with was trying to format my output. When you output text it is always left justified and numbers are always right justified. There is no simple way to change this and makes creating clean and readable output very time consuming. I would say I spent 80% of this project was trying to figure out how data was stored and formatted. It felt very inefficient. The other thing I struggled with was understanding the passing of control between functions. Since I had to remove the go to statements and change the program to use perform instead, I had trouble keeping track of which function was running.

Cobol likes and dislikes?

I found it extremely frustrating trying to format the output of strings and numbers. The formatting isn't overly readable and I think it would be very hard to go back to and maintain at a later date. I also don't like how Cobol programs are designed to be human readable. The older program contains many long lines of full English sentences. I think this would be helpful for someone learning Cobol as a first language but for anyone coming from another language it clogs up the screen and makes for 'messy' code. Unfortunately, I didn't find any benefits to using Cobol for this assignment and I wouldn't be surprised if the only use for Cobol today is maintaining old code rather than writing new software.