2017-02-06

CIS 3190 A1: Fortran Handout

Fortran is a legacy language made by IBM in the late 1950's, and was primarily made for scientists and for raw/heavy calculations.

The first challenge I found when re-engineering this code was understanding the f77 version of Fortran. It looked like assembler with go to statements and the syntax was also similar to low level language but at the same time it had a lot more features than assembly. I tried to get over this first by copying the code that the professor sent us into a .for file and just understand how the code worked by commenting lines out. However, that plan just failed miserabley because the weird column/special numbers and go to statements were hard to read because of how unfamiliar I was with it. So my next plan was printing out the code that the professor provided and go through the code by hand with my peers so we had multiple views on this. Also writing out the code in sequential order of how its written as pseudo code also worked, all I had problems with figuring out after that was what the variables i and j were and how they worked with randomizing the dice roll.

At first I thought this assignment was going to be very similar to a Game of Pig, a dice game that we wrote in C back in first year for CIS 1500, where the dice rolls were actually random from 1 runtime to another. But then after looking at all the sample output that the prof sent us I realized the randomizer wasn't actually random but instead it would give different scores based on the values of i and j that you give. Once, professor Wirth sent us a really helpful flowchart for the diceroll() function, I had all I needed. After that all I needed to do was copy the flowcharts from the pseudocode that it was to Fortran syntax.

This came to my next and final roadblock, I was so used to languages like C and python that I completely forgot that whitespaces mattered in older languages. I tried to format the code like I would for any other assignment and it just spat so many errors back at me. After a bit of fooling around and looking up stuff on stacked overflow, I finally got my Fortran code working and it was perfect. All that was left after that was literally copying the whole program into C syntax and having it work. Overall, it took me about 2 days to get Fortran working (including time to understand the algorithm), and half an hour for the C code.

My code isn't the prettiest though. In terms of how the code is written, it's pretty clean with comments to show what everything is so users have a better time if they ever need to re-engineer this code. The on-screen stuff is the ugly bit, as I ask users for all the values in one line just being delimited by spaces. It assumes that the user has perfect input and that could be something I can improve on next assignment.

By: Kevin Pirabaharan

Questions:

1. **What were the greatest problems faced while re-engineering the algorithm in Fortran?**

   The hardest part of re-engineering the algorithm was understanding the syntax of the .for version of Fortran. It looked similar to assembler in some ways but at the same time I had a hard time following the goto statements until someone walked me through it on paper. However, I was still kind of lost as to how different the versions of Fortran were and online resources for Fortran never specified what version the examples they had were from so it took a lot of research but with the given flowcharts, I eventually figured it out and had a working program.

2. **What particular features make Fortran a good language? (In comparison to C for instance).**

   - Fortran allows support for variable-dimension array arguments in subroutines, while C array arguments have to be stated when they are declared and can't change at all.
   - Fortran is a lot faster than C in terms of heavy calculations. A lot of scientists, particularly physicists, tend to still use Fortran because of this. Fortran doesn't use aliasing which may result in some short comings in terms of features. But this allows Fortran to be faster to run than C as it doesn't have to slow down to process or include pointers.

3. **Would it have been easier to write the program in a language such as C?**

   As someone who comes from a background of more modern languages such as C, java and python, it'd be easier to code the assignment in C. In fact, we did have to write the dice program in C. Back in CIS 1500, we wrote a small dice game called 'A Game of Pig' which was really similar to this game and that only took me around a few hours to code as a novice programmer. Fast foreword to now, writing this assignment in C was a lot easier than reengineering it in Fortran. It only took me around 35 minutes to have a fully functional program that work with the test cases given by the prof. It was kind of tedious to set up pointers as Fortran doesn't need pointers for the values of i and j to retain their values throughout the lifetime of the program. This may be partially due to the fact that I am not familiar with legacy languages like Fortran so I had to spend time learning the syntax before being able to re-engineer the .for code to f.95 code. Despite this I think that my C code looks a lot nicer and is much more efficient than my

Fortran code (though it could be a lot more efficient if we were allowed to use the srand() function to randomize the dice rolls).

4. **Given your knowledge of programming, was Fortran easy to learn?**

   I found Fortran's ease of learning to be moderate. Though this may be my very first time working with a legacy high level language, the syntax is similar to some of the languages I have worked with in the past. The .for file that the professor gave us looks very similar to the assembler language that we used in our micro computers course. So it didn't take me as long to understand how the rand functionality worked using the i and j values. In addition, the .f95 version of Fortran was pretty similar to C syntax, except that white space really did make a difference in how your code ran and calling functions took a bit of research before I figured it out. At first glance Fortran just looked like a bunch of garbage to me because of all the go to statements and special column numbers. However, the flowcharts provided by professor Wirth actually helped a lot and made this a really simple assignment.

5. **How would you feel about re-engineering a 10,000 line Fortran program?**

   If I had to re-engineer a 10,000 line Fortran program then I would have to rethink all the life choices I made to be in this program. Considering that the program we re-engineered was only 58 (so basically it would be 172 times the assignment we just did) lines of code, most of which were basic if or else if statements, a massive program would be a tremendous amount of work and would require a much more in-depth knowledge of the language. Sure, maybe reading the code would be a lot easier with practice and experience, but managing all the calls and go to statements in a TEN THOUSAND line Fortran program would be insane. A project like this would either have to pay well or require more than one person to do. At some point it would be easier/cheaper to rewrite the code from scratch in a more modern and useful language like Java or C.