FORTRAN Evaluation

Riley Fiske – September 24th, 2023

***Preface – Previous Experience***

Going into this exploration project, I had no prior experience with Fortran other than knowing of its existence and the extent of talking about it in our first couple weeks of lecture. This was also my first experience in having to compile a program and execute it independently afterwards by hand. This brief exploration of Fortran was very beneficial, maybe not in the respect of using it again for projects in the future, but for the increased knowledge coming from learning the basics of another language, and especially one that is as dated as it is.

[Note: to compile my program, the module also needs to be called in gfortran. I wrote a .bat file that does the compilation in this way for the user and then opens the generated .exe file after pressing any button to continue]

***Readability***

Fortran is a little clunky when it comes to readability since it does not logically read top to bottom always. For example, you must declare all your variables and their data types at the top of the program when you may not use them for 300 more lines. This causes problems when you are reading down that far and encounter a variable that you do not recall the type of. On the other hand, it is easy to know that you can look back at the top of the program to check the data type of any variable as opposed to searching throughout the application for where the variable was declared, pros and cons for this aspect of readability.

Fortran contains more key words than Python requires for a given program, but probably about the same amount as java. Fortran requires specifying the type of procedure or type you’re beginning (program, module, subroutine, etc.), then a few arguments (implicit none, for example) directly following, which is not the most readable from a quick glance if that is not impacting directly what is happening from a reading standpoint.

***Writability***

Fortran is not as easily written as Java in my opinion because of the more structured data typing required beforehand. If the goal of a programmer is to not jump around their application very much while coding, they better know exactly what variables they will need before beginning to write any steps to be executed since all variables need to be declared at the top of a program, with their types and corresponding space to allocate for them given. This means that if the programmer quickly wants to create a variable to hold a value from an array while rearranging its elements, for example, the programmer must go to the top of the program and declare this variable up there rather than spontaneously throughout the program where it is logically introduced.

The loops and conditional statements of Fortran also feel clunkier than in Java, although I know time has fixed many of these problems. All loops start with “do” and end with “end do” as opposed to incapsulating all elements of the loop with curly braces like in Java or in the correct indented column like in Python. This can throw errors very fast if a programmer forgets to end the loop or condition. Fortran also only had traditional “do” loops that would repeat infinitely for a long time before adding loops with baked-in exit conditions. This means the programmer would have to strategically add the exit condition somewhere within the loop as opposed to being forced to specify it.

Overall, the rest of the way Fortran is written was quite logical and made sense for writing without too many hiccups.

***Reliability***

Because of the stingy nature of data typing in Fortran, programmers must be very careful with the types they are using in every context. Not only through type checking are errors more prone, but also through array indexing since the amount of memory required for a such an object must be declared at the same time the array is declared, making it much easier to go out-of-bounds. The error messages given when problems arise is usually descriptive enough to point you to the right line and discover the problem, it is just much more likely to happen in Fortran than in previously explored languages.

***Cost***

At one point in time, Fortran was one of the leading languages in popularity and utility, but today not as much. Thus, it would cost much more to have a program written in Fortran since it is much harder to find a Fortran developer than it was 20 years ago or more. Given the more control a programmer has with memory in Fortran, these programs most likely run much quicker than other programs, making their run cost lower.

***Quick Comparison to Java***

Fortran is probably the furthest out there language I have worked with, using the most different variable declaration scheme, strict memory allocation, and older-functionality loops among other things. Fortran felt similar to Java in that the indentation didn’t matter as much (unlike Python), and statements were encapsulated within “program” and “end program”, for example, keywords rather than curly braces ({}) like in Java.

I definitely felt the age of Fortran while writing in it as many of the more modern Computer Science tools felt to be missing, but all the important elements of what made coding coding to me were still present. After writing in it for a little bit, I was able to grasp the syntax and structure of where things are arranged and placed quite well.

***Screenshot of Fortran Sorting Algorithm Execution:***

***A screenshot of a computer

Description automatically generated***