Abhishek Jhoree (0986820)

Reflection Report

Synopsis of my coding experience:

Below is a synopsis of my coding experience in working with the 4 languages; namely C, Fortran, Ada and Python in re-engineering the Spigot's Algorithm for PI.

Synopsis of C:

- Since I had the most experience with the C programming language, I decided to start the re-engineering process with it. Using the re-engineered template that I would have with the C code would make it much easier for me to translate over the other legacy languages and python since the C syntax for loops is very familiar and intuitive for me.
- The first step I took in the re-engineering process was to look at the Pascal code for the Spigot's algorithm that was provided. I first identified what each variable was used for in the structure of the algorithm. I then identified each loop structure to gain a comprehensive understanding of how the algorithm worked.
- Once this was done I started porting the code over to C. This was achieved relatively easily. The algorithm was fairly simple requiring only basic loops and existing data typed in C, like integers and integer arrays. I also added the required file handling portion whereby I asked the user for the filename to store the data. Again from past experience, this was achieved fairly easily as well.
- To overcome any potential integer overflow problems, I wrote each digit one by one to the file after they were calculated.
- Extensive testing was carried out on the code and it was found to be very accurate up and until the maximum value of an integer, which N holds in the algorithm, but apart from that using a different data type would have fixed it. With this now solid template for the re-engineered code, I could now move on to the different coding languages.

Benefits and limitations of using C for this given problem:

C was a particularly good choice for performing this algorithm. C is very fast when it comes to performing computationally itensive algorithms and I found very quickly that I could produce the desired output in a fraction of a second even for large values of N as compared to another language like Python. Loops were easy to implement in a comprehensive way and C also has good file handling structure which made it easy to write each char to the file one at a time.
One drawback could be that memory allocation in C has to be handes by the programmer if you wish to have dynamic sized arrays or strings.

Synopsis of working with Fortran:

- From prior experience in working with Fortran, writing the algorithm in Fortran was not very hard to do. I made use of the re-engineered code I previously wrote in C and understanding of the program structure; i.e. how the different variables were used to compute Pi and how the loops worked to approximate the value of PI.
- The data types being used were mostly available in Fortran, for instance, integer, strings, and character arrays. Hence i could easily replicate the required data types for performing the algorithm.

- File IO in Fortran was new for me. It took me some time to get used to the file handling in Fortran. It was especially tough having to deal with unwanted spacing when writing to the file as well as string concatenation to generate a single string of Pi so as to write it as a whole to the file.
- This approach enabled me to achieve the desired format for the Pi value
- The function was extensively tested and it was again found to be accurate to at least 1000 digits.

Advantages and drawbacks of using Fortran:
The biggest advantage that Fortran has over C is that it is usually faster than C for programs that require huge mathematical calculations. Fortran was specially designed to make mathematical calculations and therefore whenever it comes to programs designed exclusively for large calculations, Fortran has the edge over C. It is a popular language for high-performance computing and is used for programs that benchmark and rank the world's fastest supercomputers
One big drawback of Fortran if the poor structure to handle strings and also the ambiguous way that the language writes to files

Synopsis of working with Ada:
- Similarly to the other legacy languages, the previous experience I had with Ada made it significantly easier to work with, since much of the syntax was already known to me.
- I was also able to easily implement the required data types for the different types of variables required to perform the algorithm. Ada already had data types like integers, strings, and integer arrays as well a good file handling structure in place.
- Similarly to the previous problems, the file handling portion was easily implemented as well as the various loops and functions calls and variable assignments that the algorithm required to operate. Thus these were implemented relatively easily.
- Once complete, the code was tested extensively to a large number of decimal places and was once again found to be very precise, differing only on the last thousandth digit. But in retrospective, it was very fun to do this problem in Ada especially given how stringent Ada is.

Advantages and drawbacks of using Ada for this algorithm:
The most useful feature of Ada is its ability to detect bugs very early on in the development process. As a programmer, the most important aspect of my job is to be able to deliver bug-free code and Ada greatly facilitates this.
The structure of Ada is also very English-like. Things like loops and if statements are very easy to understand and read to any programmer. The syntax also greatly resembles that of common languages line C for instance which makes it easy to transition to. Ada also allows specifying the range that a variable can take which made it particularly useful for this assignment so as to only allows values larger than zero
One drawback of Ada is that it takes more time to write the same code in Ada than it would in another language. Languages like Ada attempt to reach safety by restraining the programmer, enforcing extremely strict rules, disabling potentially valid/techniques because they may be incorrect in certain cases.

Abhishek Jhoree (0986820)

Synopsis of working with Python:

- Python was definitely the most fun language to work with out of all of the 3. Implementing everything was by far easier as accessing arrays or string operations are by far easier to do in python as the latter allows that.
- Loops were easily implemented as well as writing to the file as python handles that quite easily and intuitively.
- Once complete, the code was extensively tested and it was found to be very accurate as well. Since Python does not restrict array sizes like a language like Ada for instance, it makes it much easier to code and implement this algorithm

Advantages and drawbacks of using python for this algorithm:
Python is a very easy language to learn and pick up. It has a vast amount of resources available online that greatly help in helping any coder figure out anything that he might want to implement as well as having a vast array of libraries readily available.
One drawback of python is that it is relatively slow as compared to other languages. Having implemented a timer function for my code, it was found that it was about 1o times slower than C or Fortran for instance.

Analysis of results:
Below is a short analysis of the performance of the different languages. The table shows how the algorithm performs for each of the languages

C	Fortran	Ada	Python
0.049439 s	**0.0490 s**	**0.05872 s**	**0.762565 s**

We can clearly see that Fortran and C are tied for the first place with nearly identical execution times followed by Ada and Python very far behind.
In fact, Python is **15 times** slower than C or Fortran! Thus, python is not very good at computationally intensive mathematic calculations and languages like C should be favored when working with such functions.