CS225 Assignment 2 - "Where have I been ?" - languages comparison

Evdzhan Mustafa

April 11, 2014

1 Introduction

This is my report that compares the three languages used for the CS22510(2013-2014) Assignment 1. It discusses C, C++ and JAVA programming languages, comparing and contrasting the differences between them. I mainly talk about string manipulation, libraries, the used data container and some other language specific features.

2 JAVA implementation

Feeling really comfortable with JAVA, it was the language I used to develop my first program.

Data container - For a data container I used the library class LinkedList. With just one import statement I would get a handle over a really stable container class. I would add new location to the container whenever satellite fix was good. Then at the end I iterate over it, fetch the data stored and output it a GPX file.

String manipulation - The string manipulation in JAVA was fairly easy task. Just declaring a String array, and filling it with the String split method, would give me handle of every sentence I had interest in. I could easily loop through that String array and determine whether satellite fixes were good, or fetch the coordinates and the time.

Libraries - A library class, I had really good use of, was the Date class. It helped me store the times from the two streams. Furthermore I used the implemented by that class function compareTo method (comparable interface). This function helped me synchronise the two streams, as it would tell me whether two times are equal, or which is before the other.

Other features - A great feature, that I used in my JAVA implementation, was the use of nested classes. That is my Location class. It did make a lot of sense to put the Location class inside the GPSReader class. The logical relation between the two classes makes the code more understandable and clear. The reason for that is the fact that the Location had only three fields. I did not want to create a whole new class, and make setters and

getters for those fields. Rather than doing that, I decided just to put it inside the GPSReader class, thus giving access to its fields.

3 C implementation

Having my implementation done in JAVA, the next challenge was to repeat that using C. Now, that wasn't a easy task. I had to move away from the Object Oriented Paradigm. Furthermore I did not have that rich set of libraries that JAVA provides.

OO paradigm issues and memory management - Having no objects did cause me to spend a lot of time just thinking about how I would implement the algorithm I created with the JAVA implementation. It was only now that I realised how powerful the Object Oriented paradigm is. Rather than having the data structures built around the functions, I had to use functions built around my data structures. This caused each function to take as a parameter pointer to a struct, which was predefined by me. After realising how I can achieve similar functionality as the OO implementation in JAVA, procedural programming became much easier task.
A fact I want to mention here is that I could also use global variables to avoid passing the addresses of the structs. But I was aware that this is really bad practice. As long as I passed the correct address of the object manipulated, I did not have to use any kind of global variables. The fact that I had to use pointers, and pass the addresses of my structs, increased the difficulty of the task significantly. One has to be careful with the use of pointers and memory management. Although I tried to produce good and robust program, I wouldn't be surprised if there are some kind of memory leaks or dangling pointers in it.

Data container - In order to be able to store the data from the two streams, I had to implement my own container. Since we were required to treat the input files as streams, I had to implement a container that wouldn't know how many elements are going to be stored. So the linked list was the obvious choice. Now this wasn't a pleasant task, and it took me some time to implement it. The overhead of doing that hindered me a bit, but I knew how my overall design would look like, thanks to the previous implementation in JAVA.

String manipulation - String manipulation in C was really big trouble. I met few problems on the way. Initially I decided to use the strtok method which would tokenise a string upto a delimiter specified by me. But the function behaves badly when it reaches two or more consecutive delimiters. I had really big trouble debugging that. Having discovered where the problem was, I had to refactor huge deal of my program, using different method which was the POSIX method strsep. Eventually I managed to achieve the same functionality as the JAVA program, but I had to spend many hours debugging the initial problem, and then finding proper replacement for the strtok function. I found it extremely time consuming to process strings in C.

Libraries - I used few libraries(header files), such as the math.h and the time.h libraries. The math one was used when I had to round up the GPS latitude and longitude values. The time library was used to compare and record the times from the two streams. The functionality was similar to the JAVA Date class.

4 C++ implementation

Finally, I had to implement the program in C++. A great advantage was the fact that I could use all the libraries used in the C program. I was back in the OO world. So the design I used was exact copy of the JAVA implementation.

Data container - As a data container, I used the vector class from the standard library. I had no previous experience using it, but it was pretty straight forward. The good thing is I was free of the overhead of implementing my own data structure. Much like the JAVA implementation I barely spent time thinking about the data container. I just had to include the vector header and that was all.

String manipulation - String manipulation in C++ was a huge improvement over C. First of all, I could use the C++ string class. I had similar set of functions to play with strings as I had in JAVA. The only difference is that I had to tokenise a sentence by myself. I achieved essentially the same functionality as the JAVA String.split method.

Libraries used The libraries used, were standard C++ libraries for input/output, along with the C libraries for the time and math. I used exactly the same time struct from the time.h header as in the C program.

5 Comparison

With JAVA I was able to express my thoughts as code really easily. It took me just few lines of code to express highly sophisticated ideas. For example it freed me from implementing my own container class and the string manipulations, and let me focus on the overall structure of my program. I was able to think more easily of how the program will look in the big picture, with no overheads of the low level design problems. This is a really huge advantage of JAVA when it comes to abstraction. The libraries provided right from the get go, are really powerful tool.

C on the other hand was a lot harder. I had to first implement various functions such as the linked list one, string manipulation functions and more. This required that I first design and implement those functions, and only then be able to get back to the design. At some point the design would change, which would lead to changing some of the functions. The consistent changes to my functions, would sometimes change the functions signature as well. This would lead to additional time spent on refactoring the header files as well. In JAVA I would only think about a single source file with no additional overheads. The C implementation was hardest to do, mainly because of the lack of high level abstraction, object orientation and the vast amount of standard C libraries.

Writing the last program in C++ wasn't a big challenge. I used exactly the same structure design as my in JAVA program. This led to almost similar amount of code as in my JAVA implementation. The difference was the fact that I had to also think about memory management. With C++, I could store my objects directly onto the stack. In JAVA I did not have that option, because every single object, by default, goes to the heap. I believe I managed to build my C++ classes with no resources handled by them. That is, I placed all my objects on the stack. I had to be careful so that these objects do not go out of scope, and I believe I succeeded. I would say C++ gave me the low-level control of how my program will access the memory, much like C, while keeping all the good stuff I had in JAVA, such as the Object Oriented paradigm and the libraries.

6 Final conclusion and comments

In terms of effectiveness of the code written, JAVA is the clear winner for me. In JAVA I was able to solve various task with just few lines of code. In C and C++ it would require me to write a set of functions, to to the same task, and also keep in mind all the memory I have allocated.

Having said that, we must also consider the scope of the software built and its purpose. One thing to have in mind is that the task we were given was essentially really simple. Just read some input files, process them, and produce an output file. The high level abstraction that you have with JAVA, suited the designing of my algorithm really well. For such small task JAVA can be really friendly, with its stability, and the rich APIs.

As much is it friendly, JAVA also limits your control over the hardware. I do know for example that direct memory access through pointers in C and C++ can much faster than the indirect memory access provided in JAVA. But for a small program such as the one we had to implement, it really doesn't make a difference. The simple file processing wouldn't be significantly slower in JAVA. If one doesn't have much time, and the program itself is not intended/required to be the very quick, JAVA would be an OK language to go with the development.

On the other hand, if one has implement a very complicated program that would run for days, weeks, or maybe even months, one would better choose C/C++. If the task did require continuous memory allocation and deallocation, C/C++ would outperform JAVA for sure. Writing the code in C or C++ would definitely take more time, but it is required if the nature of the software that has to be produced, demands speed. There is reason why C/C++ is used in the kernels of many operating systems.

My final remark is that all of the three languages, are must to know for every software engineer. Although JAVA is my favourite among the three, C and C++ let me appreciate the features I get in JAVA right from the get go. Furthermore the language picked when new software is built depends on various

factors concerned with the nature of the software itself - its scope, environment, time-to-develop, life cycle, requirements(e.g. speed) and many others.