CS 22510: "Where Have I Been?" - Languages Comparison

Due on Friday, April 11, 2014

James Euesden - jee22

Contents

Introduction

This document is the second part (Assignment 2[**?**]) of the "Where have I been?" assignment. In this report, I will discuss the languages and my experiences with them. While the first part (Assignment 1) was about the programming for GPS Processing, this is about the differences and similarities between them. What I found useful, or difficult, between the languages, any libraries I may have chosen to use, and what language I felt was best suited to the task.

Language Comparisons

Brief

I chose to begin programming in Java, mainly because it is the language I have written in the most and so felt the most comfortable with tackling a new problem, and the NMEA 0183 format, which I was unfamiliar with. Java has enough support from the Java API that it was relatively simple to create the application once I had come to grips with NMEA 0183 and formed the structure of the application in my mind.

Using C was also relatively simple, and the use of pointers to the data I was manipulating (using a struct to represent a stream), quite literally streamlined my application's data flow (if you can ignore the pun). The simplicity of writing in C, when it comes to passing items to a function, helped me consider the problem in a way that actually made me think of the simplest way of writing the application. This was to the point that I refactored some of my Java code after writing the C application because of better design choices, brought on by how simplistic writing in C was. I could say that this is a hinderance of Java. In that, while there are a lot of great features from the API and with Object Oriented programs, I found my original program structure and design to be much more than was necessary.

Writing in C++ was an interesting experience, as it combined the simplicity and control over the data that C has, with the benefits of Object Oriented programming. Making items into Objects, in both Java and C++, made creating my methods and variables a lot cleaner than in C due to the inherant ideaology behind keeping things as 'Objects'. This lead to taking some of the abstraction out of the functional type programming in C, and into encapsulating data and methods that logically make sense together.

File Input/Output

Accessing Data

Date & Time

For both C and C++ I used the 'time' library[**?**], and for Java used the Calendar class. I found the Calendar class cumbersome to use and in some ways hindered my programming. On the otherhand, using the time library was perfect for checking the timestamps between sentences, comparing them, creating new timestamps from sentence data and even updating a timestamp with new sentence data without a date.

When programming in Java and using Calendar, making a new time with the date and time provided by a sentence was fine. When it came to updating a time without a provided date, it became much more difficult to do. My method for handling this in all three languages was similar. Just use the most recent date provided (potential pitfall for the application's future use where the time passes 00:00. This could be rectified in a future edition).

This was simple to do in C and C++, with simply turning the time_t struct back into a tm struct, updating the values and then using mktime() to get it back to time_t. While the process with Calendar is similar, I found that I had trouble getting the time to be correct, with milliseconds assigned randomly, and needing to use .clone() to get correct values for the date.

Libraries

Boost
: NMEA parsing

Suitability

All good. C++ best.

Conclusion

References