CS22510 - Assignment 2
A Discussion on Language Choices

Samuel Jackson
`slj11@aber.ac.uk`

April 16, 2013

1 Discussion of the Choice of Languages Used

In assignment one of this module, we were tasked to create three programs to perform different tasks to do with rider events using three different languages. Each of these three programs had differing requirements and therefore will be better suited to the features offered by each languages.

I chose to use the Java programming language for the checkpoint manager application. There are several reasons why i felt this was the best choice of language for this particular application, but the biggest reason was that Java has an extensive GUI library available as standard. This is useful because there is absolutely no set-up required to begin creating GUI applications as the Swing library is already bundled with the JDK. There are also several graphical tools to help create Swing GUIs rather than having to tediously hand code it.

Another major reason why I chose to create the checkpoint manager in Java was because of its easy deployment (including the GUI) on multiple platforms. Because compiled Java code executes on a JVM, all library features are supported out of the box, so there are no worries about cross platform compatibility. This is useful as the checkpoint manager application is designed to be run in the field, potentially on many different types of operating systems. While both C and C++ are portable, it can be more difficult to ensure that code written for one platform also supports others (e.g. when working with the file system).

The wide choice of data structures also played a key part in my decision to use Java for this application. As I had already written the event manager in C, this application was the most complex to write. The standard containers, such as hash-tables and priority queues, available in Java cut down the development time and increased the efficiency of the application. This contrasts, for example, with C where I used a simpler to write linked list for dynamic storage which is less efficient for accessing elements midway in the structure.

For the event creation program I chose to use C++. The event creation program was the smallest of the three applications and being less familiar with C++ than Java and C, I decided to make this application with C++ as I had less knowledge of the facilities available with C++.

The development of the event creation program benefited from the Object Orientation of C++ by allowing me to better structure data collected from users in the underlying model. As the program allows users to have multiple events, entrants and courses instantiated at the same time, the power of classes (instead of the alternatives such as structures) was a key reason why I chose C++ for this application.

Another key point for selecting C++ was that while its standard library might not be quite as feature rich as Java's, it still provides a wide range of containers and helper functions, making development of the application substantially easier than with C. It is also easier to work with file streams in C++ compared with C, which was very important in my choice of language since a large part of the functionality for the event creation program deals with formatted file creation.

While C++ and Java share many common features, the real pivotal point in my decision of which one to use for each application was that GUI development in C++ requires more set-up and does not come as a standard part of C++. Instead an additional library, such as Qt, must be used. I also do not have any previous experience with a C++ GUI framework and therefore decided on Java for the checkpoint manager.

Finally, I chose to use C for the event manager program. The biggest reason for doing so was that the majority of the application's functionality had already been written using C for a previous assignment and therefore it was easier to keep most of the previous work and "bolt on" the new functionality (such as file locking). My choice was also heavily influenced by the relative difficulty in development of C programs compared with the more modern features of the other two languages.

2 Comparison of the Languages Used

Java, C++ and C all have very different features and characteristics that define them. My experience working with the three languages in the previous assignment has given me a good insight strengths and weaknesses of each.

While I ended up writing the most code for the Java application, comparatively I would of had to write a lot more code if I had chosen to use C or C++ instead. The reason that the Java application is the biggest is that it contains the most functionality out of all the programs in the assignment. With the other two languages (particularly C) there are less standard data structures and library functions available than with Java. A good example of this is that C++ does not have a "Date" class like Java does, but instead provides some basic functions to parse strings into dates. While the same end result can be achieved, more work (and hence more code) is involved on behalf of the programmer to get the same solution.

I have found the clarity of Java code to be extremely clear. Because of the aforementioned built in functionality of Java, it means that the programmer spends most of his/her time developing application logic code, rather than spending their time building less immediately relevant components such as having to write a linked list from scratch. With Java if you require a particular data structure there is usually a reasonably appropriate implementation already available. A good example from my code is the use of the Date class to store checkpoint time data.

Much of this is in sharp contrast to C program development. The C programming language is relatively simple when compared to a larger, more modern language such as Java. It even lacks some of the basic types which almost all other high level languages provide (such as a boolean primitive or a string data type). In sharp contrast with both Java and C++, C is a functional language, meaning that it is often more difficult to organise code than with an object orientated language. Unlike the other two languages, C incorporates virtually no support for polymorphism and has only very rudimentary encapsulation (if you class local vs. global scoped variables as encapsulation). Therefore the amount of code written in comparison to the level of functionality delivered is much larger by comparison. The clarity of code produced is often obscure and complicated because there are many "nasty" bits of C code that need to be avoided in order for a program to be maintainable, or because there many components need to be manually written. C code does not seem as readable as there object orientated counterparts and therefore good structure, organisation and documentation is paramount to producing maintainable C code.

However, there is one very important feature of the C programming language that distinguishes it as being useful in comparison to Java. The C language gives the programmer full access to manual memory management features, allowing them to allocate and free memory as they require it. While this feature was not a huge benefit to the development of my application, it has powerful ramifications for programming anything where efficiency is paramount. Another important point about the C language is that it is extremely "low level" for a high level language. This is useful because it provides the programmer with almost direct access to the underlying operating systems facilities. While this can cause portability issues, it also gives the programmer a higher degree of flexibility in some circumstances, for example when writing complex file I/O operations or hardware interfacing code.

In comparison, C++ stands out as being midway between both Java and C. It incorporates features present in both of the other languages. Being object orientated means that C++ allows the developer to create complex data structures easily, as well as utilising the power of polymorphism, encapsulation and inheritance. In fact, the C++ provides a much richer implementation of object orientation compared to Java, with features such as multiple inheritance. It also provides the manual memory management available in C with some additional improvements such as auto pointers and the new operator. This means that C++ and C, unlike Java, both do not have "garbage collection" facilitates to delete allocated memory when not being used. It is the programmers responsibility to allocation and deallocate memory. While this can be very useful, it can also be a hindrance as the developer must be careful to prevent memory leaks. Conversely, Java's garbage collection automatically handles memory deallocation removing the risk of memory leaks.

In contrast with C, C++ code seemed to require less code to be written. There is better support for basic types (i.e. string and boolean are provided) and there is better safety offered when using some potentially dangerous features (e.g. auto pointers and the new keyword) and it generally more strongly typed than C. There is also a decent set of standard containers available (such as vectors). These were incredibly useful in the event creation program as it meant that I did not have to write and manage my own dynamic storage structure (as with C). However, generally I found C++ required more code to be written when compared with Java, mostly due to the fact that the library provided by C++ is not quite so fully featured as Java's. I also found the clarity of the code written in C++ to be slightly less than compared with Java. This was partly due to the fact that there are simply less features available in C++ and therefore more code needs to be written, but also because C++ reuses operators over and over again (like C), rather than defining new keywords, to help ensure backwards compatibility. A good example of this would be the dual meaning of the ampersand operator in C++.

3 Conclusions

In conclusion, based on the discussion in the preceding sections, I would of defiantly made some different choices of language for each application given my experience with assignment one. If I had to choose to use all three of the languages (C, C++ and Java) to be used for one of the applications, I would have probably swapped the C and C++ programs around. As I mentioned earlier, I mainly chose to use C for the event management program because the majority of the code was already written in C. If I had to write the application again from scratch I would have chosen C++ for this program as this application is more complex than then event creation program and it therefore would of been easier to develop with the extra features available with C++, in particular object orientation. I would have still kept the checkpoint manager as being written in Java because of the excellent GUI support offered.

However, if I could of chosen to use any one of the three languages for each application in the assignment, I believe I would have chosen Java. This is because the applications do not really require any features Java does not have that the other languages have to offer. All three programs mostly perform fairly trivial high level operations. They do not really require the powerful memory management available in the other two languages, or the more fully featured object orientation available in C++. The "low level" operations provided by C are of no real value as most of the application's logic is more abstract. The only interfacing with the underlying system that needs to be done is reading and writing to files, which Java has some excellent classes for anyway. Furthermore both C and C++ require external libraries to create a GUI, while Swing is integrated in Java as standard.

In summary, I believe that Java is best suited for "high level" applications where performance, memory management and access to underlying hardware and operating system are not paramount, while speed and ease of development, portability and robust code are. Therefore Java would of been my language of choice for the three applications in assignment one. In contrast, if any of the programs called for high efficiency and bespoke data structures, or a great deal of interfacing with the underlying system architecture I would of chosen either C or C++. Of the two languages, I would probably pick C++ over C for the majority for applications because it provides access to the same low level operations offered by C, but with all the extra power of object orientated principles as well as being more strongly typed and hence easier to write.