Shc27 – Assignment 2 – Building Occupancy Grids – Language Comparison

Introduction

In this essay I will be comparing the differences and similarities between C, C++ and Java in the context of the Occupancy Grid assignment.

Compare and Contrast

Header Files and #includes

C++

The first thing found in my main.cpp is a collection of #include statements. These includes tell the pre-processor to include the contents of those text files in the place of the #include. Here I #include various libraries and the necessary header files. The header files tell the rest of the code that certain functions and member functions do exist and can be used. It doesn't need to know how they work, just that they can be accessed.

C

C is exactly the same as C++ when it comes to using #include with libraries. There would be no difference in the way this is implemented for this assignment between C and C++.

Java

In Java, here you would have import statements. These import statements basically act as namespaces for the libraries. Java uses packages that dictate the file path and filename for all program definitions instead of using #include

Object creation

C++

Next in my program is the main. In here I create a robot object. This is done by simply calling its constructor as such:

robot my_robot;

This creates an object with "Automatic storage". In the context of the assignment, this means that it gets put on the stack, and only lasts as long as we are in scope.

C++ can have multiple inheritance. This means that if we wanted to expand the functionality of the robot object, you would have it implement various different types of sensors in an easy way.

Java

In java this would be done by writing something like this using the "new" operator. Java only stores primitives on the stack, and keeps the stack small this way, allowing for more nesting on the stack for example. So to create the object in Java it would look like

Robot robot = new Robot();

And so this is placed onto the heap. However, Java has built in garbage collection, and as such will remove this from the heap once there is no reference to the object.

If I were to do it this way in my C++ assignment, I would be able to use, and have to use the deconstructor to free up the memory when I was done with the object, as C++ does not come with automatic, self-contained memory management such as Java's garbage collector. This means if not handled properly, C++ would have memory leaks making our assignment waste resources. Good, safe programming practise can negate this, but it does rely on the human not making a mistake.

Java does not support multiple inheritance, instead allowing for multiple interfaces to be implemented. If we wished to extend the functionality of the assignment to include many different sensors, it would be possible in Java, although would not be as "elegant" as implementing this in C++ would be.

C

While C is not classically regarded as an object orientated language, it IS possible to do object orientation in C. If I were to do this assignment in C I would do it procedurally. As object orientated C is rather convoluted and the language is not designed with object orientation in mind.

If you were to do object orientation in C to create a robot object, you would have a separate C file with a struct, which contains function pointers to private functions in that robot class. So to access the class functions you would use the function pointers fond in the robot struct.

Storing doubles

C++

To store the information from the text file in my object, I use vectors. They are declared in my header file as such.

```
std::vector < std::vector <double> > ranges;
```

Vectors are nice to use in C++ as the storage is handled automatically. They are not as space efficient as static arrays as they have a buffer so that it does not need re-sizing every time an element is added, but instead only once you reach the vectors capacity.

Using arrays instead of vectors has over downsides to it, such as the fact that arrays on the stack don't have range checking, you could cause a lot of unexpected errors.

Vectors don't have this problem as they are stored on the heap. Vectors have other benefits such as being able to insert elements into the middle easier than with arrays, but that doesn't come into practise in the assignment. Storing elements on the heap however means there is more overhead for a vector than an array, but being such a small project this also did not matter too much when the context of the assignment is taken into the equation, but it's worth thinking about if we were to scale the project up in the future.

Java

In Java I would use ArrayLists to store the data. As vectors are deprecated in Java compared to ArrayLists. I would declare this like such

```
ArrayList<ArrayList< double >> ranges = new ArrayList<ArrayList< double >>();
```

When ArrayLists need to increase their size, they do "newCapacity = (oldCapacity*3)/2+1". This is more space efficient than vectors, but less time efficient assuming you may need to do multiple re-sizings. As vectors simply double the memory allocated to it.

In Java, one advantage of ArrayLists over Vectors which we would consider when choosing which data structure to use is the speed, and ArrayLists are shown to be around 20-30% faster than their vector counterparts [1].

While they have their differences they also have their similarities since ArrayLists are derived from vectors. They can both grow and shrink dynamically once they have too many element/ not enough elements, they are both ordered collection classes as they maintain the elements insertion order and they both allow duplicated and null values.

With all this in mind I would use ArrayLists in my implementation if I were to do this assignment in Java.

C
C does not have the luxury of having ArrayLists or Vectors, so it must instead get by on static arrays, or by declaring / creating our own vector type.

If it was to use static arrays, one way to do that would be to find out how much space you need when scanning the file, and dynamically sizing it once you know using malloc. These fixed arrays have certain advantages over other dynamic arrays. They are more memory efficient, have big O of 1 insert, delete and read. It is also pretty easy to implement.

Creating your own vector type is a possibility, and would function similarly to the vector type that C++ / java has access to. If I were to do the assignment in C, I would probably implement vectors myself, as if we were to expand the functionality of the program, having access to a vector would make a lot of things easier.

Reading in Data

C++
Once I enter the constructor for the robot I want to read in the files, so I call read_poses(). In read_poses I pass in the vector of vectors to store the data in. To read from the file, I use ifstream from the std library, and pass in the name of the file, which I hardcoded. An advantage of C++ in this section is that is uses templates, which means that it can accept any type of input when reading a file. It does not have to be specified prior to reading the data.

Java
To read from a file in Java, I would first have to create a file object. This file object would contain the name and extension of the text file. In the assignment this would look like

File file = new File(“poses.txt”)

I would first of all have to import java.io.File, as well as java.util.Scanner to read from the file this way. I would then create a new scanner object, passing in the file object I just created as a parameter. Then while this scanner has a nextline I would keep putting data into the ArrayList, putting each line into its own ArrayList. The nice thing about the Java scanner, similar to the C++ ifstream, is that it is a generic, and as such can accept any type of input without having to be declared prior.

C
In C I would use fopen to open the connection to the file, and the use fscanf to process the data from the file. This would get stored in a buffer, and subsequently added to the vector I would use to hold this data which I created myself.

C does not have file objects like C++ and java, so I would have to use a file pointer when reading from the file. C has a disadvantage when compared to the other languages in this area, as the type on input must be specified before reading from the file. This would usually be done using regex. However in the context of this assignment we knew we were reading in doubles, so it wouldn't be much of a concern.

Mathematical Methods

C++

For the member functions which didn't belong to any class, and served a purely mathematical purpose, placing them into their own static class and accessing them from there would be the way I would implement this. The reason I made them static was so that I wouldn't require an instance of the class they were placed in.

Java

For these methods in Java I would place them into a static class, and access them from there just like you would if you were to use C++. Again so that I wouldn't have to create an object to use them.

C

In C I would just place the functions at the bottom of the file, or you could simply place them in a separate C file, for example called "conversion.c", and you would then #include that file.

GRID BUILD

C++

To build the grid, I need to pass in the ArrayLists of information to the gridBuild member function. I pass in my values by reference, this means in the function I would use the reference to access the actual parameter.

C

In C you would not be able to pass in parameters by reference using the ampersand, instead I would pass in the first item in the array, as when passing an array in C you just pass the pointer to the first item in that array. From that first item you have access to the whole array.

Java

In java you would pass by reference the ArrayList to the method. In this method there is the use of a "sleep" function in C++ and C to slow down the animation of the grid. This would not work on every machine and OS 100%, and so Java has the advantage in this area as java runs in its own virtual machine, it's equivalent sleep method will work universally.

Suitability

In summary, I believe Java is the language I would prefer to do the assignment in. Java has simpler object orientation, thanks to the built in garbage collector. Storing the information in ArrayLists has advantages over using vectors of C++ and arrays/self-made vectors in C. Mainly due to the space efficiency and methods of access. When reading in the data, both Java and C++ are similarly advantaged thanks to the inclusion of File types. And finally the portability of the Java sleep functionality is an advantage only Java has over C and C++. C not being (strictly) object orientated matters a lot in the context of the assignment, as in the future you may want many robots, or many sensors that do different things. The lack of ease and accessibility of C object orientation makes it the language I would least likely use.

Bibliography

[1] Kiourtzoglou, B. (2016). *Java Best Practices – Vector vs ArrayList vs HashSet*. [online] Java Code Geeks. Available at: https://www.javacodegeeks.com/2010/08/java-best-practices-vector-arraylist.html [Accessed 14 Apr. 2016].