CS22510 – Assignment 2

Languages Comparison.

Lukasz Wrzolek – luw19

1. *Introduction*

The second assignment for cs22510 module is a report based on my experience of Java, C and C++ languages. The task is to compare and contrast this three languages, discuss their aspects and which of those languages will be most suitable for previous task.

1. *Differences*

The main thing in every language is design goal. Each of languages above has different design goals. C is the procedural function oriented language when C++ and Java are object oriented languages.

The next difference which I noticed is compilation so portability of the compiled code. In Java I can compile code on one machine and run it anywhere, but It is not that easy when we are using C++ or C. In that case really helpful is compiling code on the server which using specified standards so from the beginning we are pushed to using specified standard and portability of our code is bigger than before i.e. (When I was doing my C assignment I had to compile my program on central.aber.ac.uk server using NetBeans IDE), namely program compiled on that server was working for marking people. When I was doing C++ assignment I had to show the command which I used to compile my .cpp files to build the solution, create *a.out* file and run the program, although the results can be different for each machine. If we are talking about compilation the files is worth to add that single command in C or C++ have compile every dependent classes together i.e. *gcc main.c first.c second.c*  following in C++  *g++ main.cpp first.cpp second.cpp,* when in java all we need to compile is main class and all supportive classes are re-compiled if needed i.e. *javac main.java*. There is also a second part of compiling when we need to link math library, so we are using *“-lm”* part during compiling namely: *gcc –lm calculate.c.*

I have been writing about compilation and portability so now it is time for next difference which is simple thing like arrays and size of the types. I am not gonna talk about declaration of the arrays because in every language it is the same procedure – type *name* [size]; There is more important thing to observe, namely sizes. In objective programming we have simple function *nameOfTheArray.length* which returns us a size of wanted array. Although in C that function does not exist so the best possible option is declaring sizes of the arrays by the pre-processor and use it later in for loops or in array searching when the size of that array is needed so we can avoid mistakes. We have different types in programming like int, long, char, string etc., but every of that types has their own size in memory. For example we can bring integer int type. This type of variable reserves 32 bit size. It is exactly the same like in Java, but second integer type long is different. It takes 32 bit in C, but in Java it takes 64 bit so it mean that we can store bigger number in java than we can store in C. Next difference in types is present in char type, namely in C and C++ it usually has 8 bit when in Java the same type has 16 bits. The last and the most interesting type is Boolean type. In java it is separate type of the variable and returns true or false depends of the result which can be positive or negative. In C that type return 0 if the result is negative and 1 if the result is positive.

Next interesting thing in that languages comparison is accessing a libraries, printing, memory addresses and data structures. Let’s define simple access to the library or header files. In C or C++ we are using #include statement and defining in following brackets header files (“ ”) or libraries (< >), although in Java we are using i.e. *import java.Utils.\** which means that we are importing every library from *Utils* without specification. Standard printing is also different Java uses *System.out.println(“To print out something”),* but it also can use *printf* statement when we import a library so during compilation it will be treated as default print statement. In C as I mentioned before we have *print(“function which allows us to send message from brackets to the user”).* C++ is a different story and it is much more complicated than printing in Java or C. Actually to print something we have to include library *<iostream>*  and use a namespace *std* which contains a *cout* definition. We can do in by two ways. Before our main function we can declare that we are *using namespace std* so it will be easier for following code in main or we can declare from which namespace that variable comes and by every statement we have to add *std::cout*, of course C++ provides us a header file and which letting us to use *printf* statement, but comparing that two statements *cout* is much more easier and clearer to use. Memory address is the most painful thing in C and C++. We need to learn a lot about pointers, how to manipulate pointers and finally how to pass values and which value is wanted value or if the variable still contain address of wanted variable. In Java is a lot easier because that language uses references so we are not involved of using that addresses. Finally we reached the most interesting part of that point which are the data structures, namely defining the data structures. In C we have clearly situation with structures, all we have to do is use keyword *struct* give the name for that structure and define values in brackets. That structure creates new type of featuring variables and can store values or strings of that variables. In Java we have different situation, because the class which is created is already our data structure, so all we have to do is define public and private variables or methods of that class and pass it through. What happen if that two so different data structures will be connected together? It is simple answer actually – C++ will happen. We are able to use that two different data structures in objected programming in cpp. We can mix that structures too, namely once defined structure in separate header file can be used in function prototypes in other .cpp file. We cannot use it in header file describing class but it can be used in functions prototypes which that class using. Accessing a data structures is also important thing in comparison but it pretty much the same in every language. In C and C++ we are using “ . ” to access data structure, in java we are doing the same thing i.e. *a.numerator* . It is more complicated when we passing the pointers in C and C++ we using arrows *“->”* in java we still using dots “ . ”.

1. *Discussion*

I was writing before about differences in those languages and in this part I will be speaking about my previous assignment which was about building an occupancy grid in C++ and reading poses and ranges from txt files. Now I will bring this aspects here and discuss any part of three languages which would have supported or hindered the representation of my solution in code.

I have to start from C++ because I was writing my solution in that language. At the beginning I need to say that it was firs time when I was learning about C++ and now I know why people saying about it an extended version of C. To be honest I don’t see any aspect which would support my solution if I would have used C language, although I’m sure that using structures in header files and function prototypes are really useful. Why I would choose C++ than C if I will write again my solution is because cpp is way easier to understand than C and it supports classes as objects when C offers a structures as data structures. It is time for compare Java as a solution of described code. The main difference between Java and previous two languages is the level of it. I mean that Java is high level language when C and C++ are procedural language. Java is really advanced language and it offers many solutions which are easier i.e. Java provides JPanel library for a graphic interface and every IDE has extensions to support programmer with setting up graphical interfaces, so graphic output can be easily modified by user during program running. When I would be writing it in Java I would do everything similar, but structures will be replaced by array lists and I would get rid of header files which are useless. Functions and their declarations would be replaced by class methods which are easier to define and printing methods wouldn’t be that long and complicated like they are in C++, namely It is easier to build toString method than writing long cout line with every variables which have to be printed, even if we create the printing function and we are passing variables directly to printing statements. This two languages (Cpp and Java) are objective languages but C++ is function oriented when java is object oriented. This mean that program passing objects as a parameters when Cpp using pointers to pass the address of the memory where the wanted variable is located so writing solution in java it will be much more complicated by creating object for each position and after passing that object to print it on the grid.

1. *Which language is the most suitable*

When I started doing previous assignment I was sure that the most suitable language for the task is Java, after I wasn’t sure about that and now when I have done comparison of this three languages I am sure that the most suitable language for the task was C++. I am sure about that because files given form Neil and Fred was clearly built. Three or five values separated by white symbol (space) so it was easy to shift current value to the local variable and after to the structure. In Java I would probably create an Array List and store the values of the ranges and positions in the list, after I would read the values from the end of the list and put it on the grid. Although, java provides an array lists using structures was much more easier to implement for me.

After describing and comparing differences between 3 languages I have knowledge which part of this language is the best for previous task and which was the worst. I think that everything is circling around the knowledge not around the language which suppose to be used in current task so it is hard to say which is most suitable, but for me and my knowledge I would choose C++ again.