Final Project Proposals

CS3490: Programming Languages

Jacob Villemagne, Wolfe Bowman, Luke Patterson

Friday April 9th, 2021

1. **What will your program do? State precisely the problem that it will solve.**

We decided to do a project along the lines of 2.4 Translating from one programming language into another (1-3). We decided to do Java code to C code.

2. **What will the user interaction look like? How will the program take the input? How will it return the output to the user?**

In terms of the user interface, we would ask the user to provide a input then after the given input our code would provide the desired output.

The input will be taken as a input file. So after the user initiates our main.io we will prompt the user to give us a file name for example “test.txt”. If we are given a valid input (a valid input being a valid file of Java code) our program will run and begin translating it to C code.

Thus, our output will return similar to our input. If everything goes according to plan, we will output a output file that will contain the code translated from Java code to C code. The new file will be placed where our program will be, the users current directory.

3. **What datatypes will be used internally by the program? How will the inputs be parsed into internal representations?**

In our project we will be starting with a list of strings in the form of a Java program. That Java program will be converted into a list of tokens. That list of tokens will then again be converted into a list of strings, but in C code.

In summation: Java Strings → Java tokens → C tokens → C strings.

4. **Give a brief outline of the process by which the program will attain its goal. A good answer could include the datatype(s) definition, a list of functions to be implemented, a one-line description of what each function does, and pseudocode for a “main” method that will call them in sequence. (At this point, you do not have to give the pseudocode for individual functions.)**

```
Main.io
{
    filename <- getLine
    Java String <- readFile filename
    Lexer: Java String → Java Tokens
    Conversion: Java Tokens → C Tokens
    De-Lexer: C Tokens → C String
    C String → output file
}
```

Functions:

- Lexer, will allow us to parse the Java sting into tokens
- De-Lexer will allow us to parse the tokens into C strings
- Conversion will be where we break down the Java Tokens and convert them into C tokens
    - Java String: Public final int var = 0;
    - Java Token: final(jVsym “var”)
    - Would go to
    - C Token: #define (cCsym “var”)
    - C String: #define var = 0;

5. **If you intend to work in a group, list the name(s) of who you will work with.**

Jacob Villemagne (Me)

Wolfe Bowman

Luke Patterson