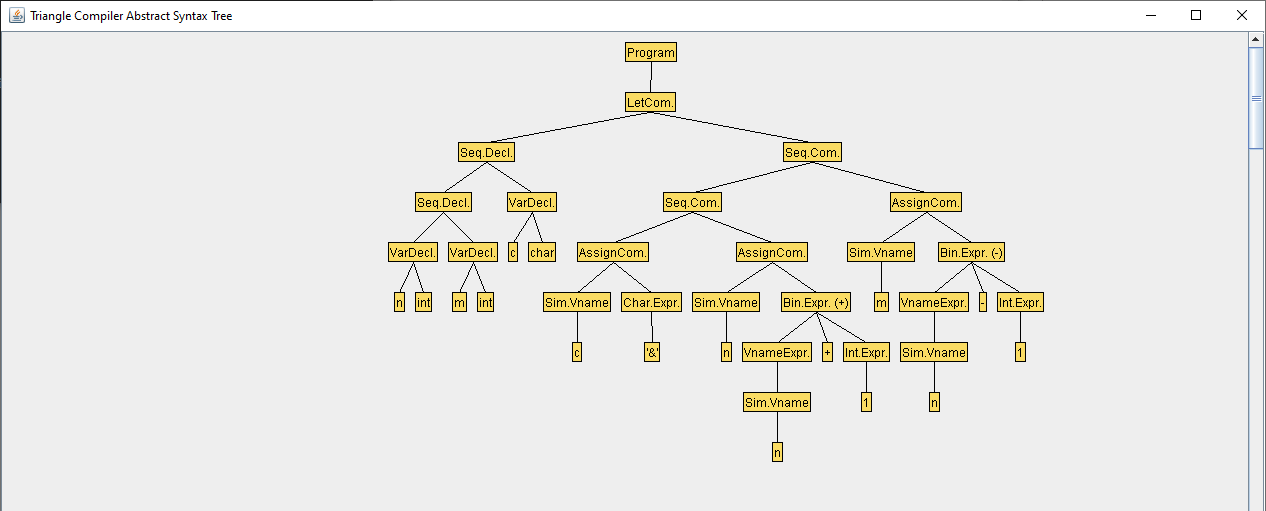
CSCU9A5 Assignment Report

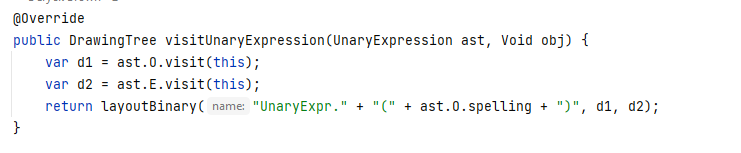
1.a) Gradle.Build was modified in the Triangle.Compiler directory and the Triangle-Tools directory to include a dependency block that calls the cli-parser library. A call to the maven repository is added also. Finally, the jar block is added.

1b.) ParseArgs method was modified to make use of the cli-parser library. This entailed creating a constructor of compiler and feeding it to the args.parseorexit method from the cli library. Code was added also as @argument functions. These allowed the code to recognise the “tree” and “o” arguments.

2a.) LayoutVisitor class was modified, mainly the visitBinaryExpression and visitUnaryExpression methods. The return method is altered to include “ast.O.spelling”. This returns the value of the operator being used. (This was added to unaryExpression but I could not find a program that would display as a UnaryExpression).



2b.)



3.) Code was added into the Parser class and within the parseSingleCommand method. In the method, the code for the Token.IDENTIFIER case was modified. In this case, two if statements were added to check if the operator tokens where either “++” or “--". This then lets the code run the set expressions that these represent, they take in the variable given and then add/subtract one from it (Eg. N = 5; N++ = N = N + 1 = N = 6).

4.) Code was added to the parseSingleMethod again in the Parser class to make use of the Token.LCURLY/.RCURLY that was defined. The code was utilized from the BEGIN/END tokens but where modified to make use of the LCURLY/RCURLY tokens instead.

5.) Code was added to many classes in this case. First I added a LOOP token to be parsed in the Parser class. This uses 4 different inputs, the first command (c1AST), the expression for the while loop (eAST), the second command after the expression condition is met (c2AST) and the command position.

A LoopCommand was also then added, this contained the constructor that defines the inputs needed for the LoopWhile to run.

Code was added to the LayoutVisitor class to import the LoopCommand and to write a visitLoopCommand method. This is used to display the arguments in the AST.

Code was added to the Encoder and Checker classes as well. These where mostly modified from the visitWhile and visitIf command methods. As such, I am not completely sure how they function.

6a.) I could create a class that can be used to count how often a visitor method is called or to count how often a node is printed, maybe counting how many times using an if statement.

6b.) Another @argument should be added to be available to intake a “stats” word to be parsed by the cli parser.