Andrei Barbulescu 40208635

**Attribute Grammar**

START -> PROG .

APARAMS -> EXPR REPTAPARAMS1.
APARAMS ->.

APARAMSTAIL -> comma EXPR.

ADDOP -> plus.
ADDOP -> minus.
ADDOP -> or.

ARITHEXPR -> TERM RIGHTRECARITHEXPR.

ARRAYSIZE -> lsqbr ARRAYSIZE2.
ARRAYSIZE2 -> intLit rsqbr.
ARRAYSIZE2 -> rsqbr.

ASSIGNOP -> equal.

EXPR -> ARITHEXPR EXPR2.
EXPR2 -> RELOP ARITHEXPR.
EXPR2 ->.

FPARAMS -> id colon TYPE REPTFPARAMS3 REPTFPARAMS4.
FPARAMS ->.

FPARAMSTAIL -> comma id colon TYPE REPTFPARAMSTAIL4.

FACTOR -> id FACTOR2 REPTVARIABLEORFUNCTIONCALL.
FACTOR -> intLit.
FACTOR -> floatLit.
FACTOR -> lpar ARITHEXPR rpar.
FACTOR -> not FACTOR.
FACTOR -> SIGN FACTOR.

FACTOR2 -> lpar APARAMS rpar.
FACTOR2 -> REPTIDNEST1.

REPTVARIABLEORFUNCTIONCALL -> IDNEST REPTVARIABLEORFUNCTIONCALL.
REPTVARIABLEORFUNCTIONCALL ->.

```
FUNCBODY -> lcurbr REPTFUNCBODY1 rcurbr.

FUNCDECL -> FUNCHEAD semi.

FUNCDEF -> FUNCHEAD FUNCBODY.

FUNCHEAD -> func id lpar FPARAMS rpar arrow RETURNTYPE.

IDNEST -> dot id IDNEST2.
IDNEST2 -> lpar APARAMS rpar.
IDNEST2 -> REPTIDNEST1.

IMPLDEF -> impl id lcurbr REPTIMPLDEF3 rcurbr.

INDICE -> lsqbr ARITHEXPR rsqbr.

MEMBERDECL -> FUNCDECL.
MEMBERDECL -> VARDECL.

MULTOP -> mult.
MULTOP -> div.
MULTOP -> and.

OPTSTRUCTDECL2 -> inherits id REPTOPTSTRUCTDECL22.
OPTSTRUCTDECL2 ->.

PROG -> REPTPROG0.

RELEXPR -> ARITHEXPR RELOP ARITHEXPR.

RELOP -> eq.
RELOP -> neq.
RELOP -> lt.
RELOP -> gt.
RELOP -> leq.
RELOP -> geq.

REPTAPARAMS1 -> APARAMSTAIL REPTAPARAMS1.
REPTAPARAMS1 ->.

REPTFPARAMS3 -> ARRAYSIZE REPTFPARAMS3.
REPTFPARAMS3 ->.
```

```
REPTFPARAMS4 -> FPARAMSTAIL REPTFPARAMS4.
REPTFPARAMS4 ->.

REPTFPARAMSTAIL4 -> ARRAYSIZE REPTFPARAMSTAIL4.
REPTFPARAMSTAIL4 ->.

REPTFUNCBODY1 -> VARDECLORSTAT REPTFUNCBODY1.
REPTFUNCBODY1 ->.

REPTIDNEST1 -> INDICE REPTIDNEST1.
REPTIDNEST1 ->.

REPTIMPLDEF3 -> FUNCDEF REPTIMPLDEF3.
REPTIMPLDEF3 ->.

REPTOPTSTRUCTDECL22 -> comma id REPTOPTSTRUCTDECL22.
REPTOPTSTRUCTDECL22 ->.

REPTPROG0 -> STRUCTORIMPLORFUNC REPTPROG0.
REPTPROG0 ->.

REPTSTATBLOCK1 -> STATEMENT REPTSTATBLOCK1.
REPTSTATBLOCK1 ->.

REPTSTRUCTDECL4 -> VISIBILITY MEMBERDECL REPTSTRUCTDECL4.
REPTSTRUCTDECL4 ->.

REPTVARDECL4 -> ARRAYSIZE REPTVARDECL4.
REPTVARDECL4 ->.

RETURNTYPE -> TYPE.
RETURNTYPE -> void.

RIGHTRECARITHEXPR ->.
RIGHTRECARITHEXPR -> ADDOP TERM RIGHTRECARITHEXPR.

RIGHTRECTERM ->.
RIGHTRECTERM -> MULTOP FACTOR RIGHTRECTERM.

SIGN -> plus.
SIGN -> minus.

STATBLOCK -> lcurbr REPTSTATBLOCK1 rcurbr.
STATBLOCK -> STATEMENT.
```

STATBLOCK ->.

STATEMENT -> id ASSIGNSTATORFUNCCALL semi .
STATEMENT -> if lpar RELEXPR rpar then STATBLOCK else STATBLOCK semi.
STATEMENT -> while lpar RELEXPR rpar STATBLOCK semi.
STATEMENT -> read lpar VARIABLE rpar semi.
STATEMENT -> write lpar EXPR rpar semi.
STATEMENT -> return lpar EXPR rpar semi.

ASSIGNSTATORFUNCCALL -> REPTIDNEST1 ASSIGNSTATORFUNCCALL2.
ASSIGNSTATORFUNCCALL -> lpar APARAMS rpar ASSIGNSTATORFUNCCALL3.

ASSIGNSTATORFUNCCALL2 -> dot id ASSIGNSTATORFUNCCALL.
ASSIGNSTATORFUNCCALL2 -> ASSIGNOP EXPR .
ASSIGNSTATORFUNCCALL3 -> dot id ASSIGNSTATORFUNCCALL.
ASSIGNSTATORFUNCCALL3 -> .

STRUCTDECL -> struct id OPTSTRUCTDECL2 lcurbr REPTSTRUCTDECL4 rcurbr semi.

STRUCTORIMPLORFUNC -> STRUCTDECL.
STRUCTORIMPLORFUNC -> IMPLDEF.
STRUCTORIMPLORFUNC -> FUNCDEF.

TERM -> FACTOR RIGHTRECTERM.

TYPE -> integer.
TYPE -> float.
TYPE -> id.

VARDECL -> let id colon TYPE REPTVARDECL4 semi.

VARDECLORSTAT -> VARDECL.
VARDECLORSTAT -> STATEMENT.

VARIABLE -> id VARIABLE2.

VARIABLE2 -> REPTIDNEST1 REPTVARIABLE.
VARIABLE2 -> lpar APARAMS rpar VARIDNEST.

REPTVARIABLE -> VARIDNEST REPTVARIABLE.
REPTVARIABLE ->.

VARIDNEST -> dot id VARIDNEST2.
VARIDNEST2 -> lpar APARAMS rpar VARIDNEST.

VARIDNEST2 -> REPTIDNEST1.

VISIBILITY -> public.
VISIBILITY -> private.

**Design**

Since I have a recursive descent predictive parser, I am passing around an AST reference and I build my tree from the bottom up . I have a node class which has a reference to parent, siblings, and child. I also have an enum which keeps track of the type of the node that we are currently parsing. I am using the function stack to pass the AST between different production rules.

**Tools**

For this project, I have made use of DOT files to show the representation of my ast tree. I am using a graphing website to show the representation of the tree.