Productions:

<Program> 🡪 {<ConstantDef> |<FunctionDef> | <VariableDec> | <Assignment>} <MainFunction>

<ConstantDef> CONST\_DEF IDENT <Expression>

<FunctionDef> DEF <Type> IDENT LPAREN <Parameters> RPAREN LCBRACKET <CodeBlock> RCBRACKET

<Parameters> 🡪 [<Type> IDENT {COMMA <Type> IDENT}]

<MainFunction> MAIN LPAREN RPAREN LCBRACKET <CodeBlock> RCBRACKET

<CodeBlock> 🡪 {<Statement>}

<Statement> 🡪 (<Jump> | <Selection>) | (<Assignment> | <VariableDec> | <FunctionCall>) END

<Assignment> 🡪 IDENT ASSIGN <Expression>

<VariableDec> 🡪 VAR\_DEC <Type> IDENT

<Type> 🡪 PRIMITIVE\_TYPE (might add user-defined types, still TBD)

<Selection> 🡪 IF LPAREN <BoolExpression> RPAREN THEN LCBRACKET <CodeBlock> RCRACKET [ELSE LCBRACKET <CodeBlock> RCBRACKET]

<Jump> 🡪 RETURN <Value>| <Loop>

<Loop> 🡪 LOOP LPAREN <BooleanExpression> RPAREN LCBRACKET <CodeBlock> RCBRACKET

<FunctionalCall> 🡪 FUNC\_CALL IDENT LPAREN <Arguments> RPAREN

<Arguments> 🡪 [<Expression>{COMMA <Expression>}]

<BoolExpression> 🡪 <BoolTerm> {OR\_OP <BoolExpression>}

<BoolTerm> 🡪 <BoolFactor> {AND\_OP <BoolTerm>}

<BoolFactor> 🡪 NOT\_OP <BoolFactor> | LPAREN <BoolExpression> RPAREN | <Comp>

<Comp> 🡪 <Expression> [COMP\_OP <Expression>]

<Expression> 🡪 <Term> {ADD\_OP <Expression>}

<Term> 🡪 <Factor> {MULT\_OP <Term>}

<Factor> 🡪 LPAREN <Expression> RPAREN | <Value>

<Value> 🡪 NUM\_LIT | FRAC\_LIT | IDENT | <FunctionCall>

Abstract Rules (Static Semantics):

* Addition/Subtraction:
  + Operator: Expression.1
  + Operands: Expression.0 and Expression.2
* Multiplication/Division
  + Operator: Term.1
  + Operands: Term.0 and Term.2
* Comparison
  + Operator: Comp.1
  + Operands: Comp.0 and Comp.2
* OR operation
  + Operator: BoolExpression.1
  + Operands: BoolExpression.0 and BoolExpression.2
* AND operation
  + Operator: BoolTerm.1
  + Operands: BoolTerm.0 and BoolTerm.2
* NOT operation
  + Operator: BoolFactor.0 (if equals to NOT)
  + Operands: BoolFactor.2
* Assignment:
  + Operator: Assignment.1
  + Operands: Assignment.0 and Assignment.2
* Variable Declaration
  + Operator: VarDec.0
  + Operands: VarDec.2
  + Rules:
    - Add entry to symbol table
    - Set scope to current scope
    - Set type to VarDec.1
* Selection Statement
  + Operator: Selection.0
  + Operands: Selection.2 (condition), Selection.6.Child (Body), [Optional] Selection.10 (Else Body)
  + Rules:
    - Change scope to that of the current statement
* Return
  + Operator: Return.0
  + Operand: Return.1
* Loop
  + Operator: Loop.0
  + Operands: Loop.2 (condition) and Loop.5.Child (Body)
* Function Call
  + Operator: FunctionCall.0
  + Operands: FunctionCall.1 (name), FunctionCall.3.Child (arguments list)
* Main Function
  + Operator: MainFunction.0
  + Operands: MainFunction.5.Child
* Function Definition
  + Operator: FunctionDef.0
  + Operands: FunctionDef.1 (type), FunctionDef2 (name), FunctionDef.4.Child (parameters), and FunctionDef.7.Child(Body)
* Constant Definition
  + Operator: ConstantDef.0
  + Operands: ConstantDef.1 (name), ConstantDef.2 (value/expression)