Grammar rules:

-> Production rule | is defined as

"" encloses a literal

[] items that can be omitted

{} 0 or more times

| either or

() grouping

<> semantic qualifiers

program -> sequenceOfStatements

sequenceOfStatements -> statement { statement }

statement -> simpleStatement | compoundStatement |functionSpecification

functionSpecification -> "function" identifier ";" typeDefinition (" formalParameters "")" "{" sequenceOfStatements "}"

formalParameters -> typeDefinition identifier { "," typeDefinition identifier }

simpleStatement -> nullStatement | assignmentStatement | functionCallStatement | declarativeStatement | printStatement | returnStatement

declarativeStatement -> typeDefinition identifier { ":=" expression} | { "," identifier } "."

typeDefinition -> typeName | ArraytypeDefinition

ArraytypeDefinition -> "Array" typeName "[" range "]"

typeName -> "integer" | "boolean" | "floating"

range -> numericLiteral

nullStatement -> "null""."

assignmentStatement -> identifier := expression "."

functionCallStatement -> identifier "(" actualParameter ")""."

actualParameter -> identifier { "," identifer }

compoundStatement -> ifStatement | loopStatement

ifStatement -> "when" "(" condition ")" "{" sequenceOfStatements "}"

{ "elseWhen" "(" condition ")" "{" sequenceOfStatements"}" }

[ "else" "{" sequenceOfStatements "}"]

loopStatement -> "loop" "(" condition ")" "{" sequenceOfStatements "}"

condition -> <boolean> expression

expression -> ["NOT"] relation { "AND" relation } | { "OR" relation }

relation -> simpleExpression | [ relationalOperator simpleExpression ]

simpleExpression -> [unaryAddingOperator] term { binaryAddingOperator term}

term -> factor { multiplyingOperator factor}

factor -> numericLiteral | identifier | "(" expression ")" | "true" |"false

identifier -> stringLiteral

numericLiteral -> [0-9]\* { "."[0-9]\* }

stringLiteral -> Everything apart from relationalOperator| binaryAddingOperator |unaryAddingOperator | multiplyingOperator

numericLiteral | "AND" | "NOT" | "OR" |true" | "false" | "when" | "else" | "elseWhen" | "loop"

"Array" | typeName | "(" | ")" | "{" | "}" | "[" | "]" | "." | "null" | ";" | "," | ":="

relationalOperator -> "==" | "!=" | "<" | ">" | "<=" | ">="

binaryAddingOperator -> "+" | "-"

unaryAddingOperator -> "+" |"-"

multiplyingOperator -> "\*" | "/"