Grammar for MINI-L Language

Program

start -> functions
functions -> Function functions | *empty*

Function

Function -> “function” “identifier” “;” “beginparams” **Dec** “endparams” “beginlocals” **Dec** “endlocals” “beginbody” **Statements** “endbody”
Dec -> Declaration “;” **Dec** | *empty*

Declaration

Declaration -> Id “:” **Assign**
Id -> “identifier” | “identifier” “,” **Id**
Assign -> “integer” | “array” “[” “number” “]” “of” “integer”

Statement

Statements -> A|B|C|D|E|F|G|H
A-> Var “:=” **Expression**
B-> “if” **Bool-Exp** “then” **Statement** “endif” | “if” **Bool-Exp** “then” **Statement** “else” **Statement** “endif”
C-> “while” **Bool-Exp** “beginloop” **Statement** “endloop”
D-> “do” “beginloop” **Statement** “endloop” “while” **Bool-Exp**
E-> “read” **Var E’**
E’-> “,” **Var E’** | *empty*
F-> “write” **Var E’**
G-> “continue”
H-> “return” **Expression**

Bool-Expr

Bool-Expr -> Relation-And-Expr | **Bool-Expr** “or” **Relation-And-Expr**

Relation-And-Expr

Relation-And-Expr -> Relation-Expr | **Relation-And-Expr** “and” **Relation-Expr**

Relation-Expr

Relation-Expr -> RExpr | “not” **RExpr**
RExpr-> Expression Comp Expression | “true” | “false” | “(” **Bool-Expr** “)”

Comp

Comp -> “==” | ”<>” | “<” | “>”|”<=”|”>=”

Expression

Expression -> Multiplicative-Expr ExprAdd
ExprAdd -> “+” **Multiplicative-Expr ExprAdd** | “*” **Multiplicative-Expr Expr-Add** | *empty*

Multiplicative-Expr

Multiplicative-Expr -> Term Multi-Term
Multi-Term -> “*” **Term Multi-Term** | “/” **Term Multi-Term** | “%” **Term Multi-Term** | *empty*

Term

Term -> PosTerm | “-” **PosTerm** | “identifier” **term-Identifier**
PosTerm -> Var | “number” | “(“ **Expression** “)”
term-Identifier -> “(” **term-Expression** “)” | “(” “)”
term-Expression -> Expression | **Expression** “,” **term-Expression**

Var

Var -> “identifier” | “identifier” “[“ **Expression** “]”