Gramática ROP GLC

S = funDec Type FunName ‘[‘ Params ‘]’ Body S
S = Decl S
S = ε

function declaration:

FunName = ‘id’
FunName = ‘main’
Params = Params, Type ‘id’ ArrayOpt
Params = Type ‘id’ ArrayOpt
Params =   ε
FunCall = ‘[‘ Lec ‘]’ ‘;’
Return = ‘return’ Ec ‘;’

variable declaration:

Decl = Type LI
LI = ‘id’ ArrayOpt Inst
LI = LI, ‘id’ ArrayOpt Inst

instantiating variables:

Inst = ‘atrib’ Inr
Inst = ε
Inr = ArrayOpt
Inr = Fc

array:

ArrayOpt = ‘(‘ ArrayAccess
ArrayOpt = ε
ArrayAccess = ‘)’
ArrayAccess = ‘intConst’ ‘)’

variable type:

Type = ‘intType’
Type = ‘floatType’
Type = ‘boolType’
Type = ‘stringType’
Type = ‘reVoid’

commands:

Command = ‘reFor’ ‘[‘ Atr ‘;’ Eb ‘;’ Inc‘]’ Body
Command = ‘reWhile’ ‘[‘ Eb ‘]’ Body
Command = ‘reIf’ ‘[‘ Eb ‘]’ Body Ifr
Ifr = ‘reElseIf’ ‘[‘ Eb ‘]’ Body Ifr
Ifr = ‘reElse’ Body
Ifr = ε
Inc = ‘constInt’
Inc = ‘id’

id list:

IdL = ‘id’ ArrayAccess
IdL =IdL ‘,’ ‘id’ ArrayAccess
IdLr = ε

body:
Body = ‘{‘ BodyScope ‘}’
BodyScope = Decl BodyScope
BodyScope = Atr ‘;’ BodyScope
BodyScope = Command BodyScope
BodyScope = Return Atr ‘;’
BodyScope = ε

list of expressions:

Lec = Fc
Lec = Lec ‘,’ Fc
Lec = ε

expression:
Atr = ‘id’ AtrR
AtrR = ‘decreOp’ ‘;’
AtrR = ‘increOp’ ‘;’
AtrR = ArrayOpt ‘atrib’ Fc ‘;’
AtrR = FunCall
Fc = ‘StringConst’
Fc = Eb
Eb = Tb Ebr
Ebr = ‘orOpLog’ Tb Ebr // or
Ebr = ε
Tb = Fb Tbr
Tbr = ‘andOpLog’ Fb Tbr // and
Tbr = ε
Fb = ‘negOp’ Fb // not

```
Fb = ‘boolConst’
Fb = Ra Fbr
Fbr = Comp Ra Fbr       // low/great/eq
Fbr = ε
Ra = Ea Rar
Rar = ‘eqRl’ Ea Rar         // equal
Rar = ‘notEqRel’ ’Ea Rar      // not equal
Rar = ε
Ea = Ta Ear
Ear =  ‘addOp’ Ta Ear
Ear = ‘subOp’ Ta Ear’
Ear = ε
Ta = Fa Tar
Tar = ‘divOp’ Fa Tar
Tar = ‘multOp’ Fa Tar
Tar = ε
Fa = ‘(‘ Eb ‘)’
Fa = ‘subOp’ Far
Fa = Far
Far = ‘Id’
Far = ‘intConst’
Far = ‘floatConst’
Far = ε



Comp = ‘greRel’
Comp = ‘lowRel’
Comp = ‘greEqRel’
Comp = ‘lowEqRel’
```