Gramática Livre de Contexto

* S é o ponto de início da execução

```
S = DeclId S
S = FunDecl S
S = ProcDecl S
S = &

DeclId = Type LId ‘;’
DeclId = ‘const’ Type LId ';'

Type = ‘int’ | ‘float’| ‘bool’ | ‘char’ | ‘string’

LId = LId ‘,’ ‘id’ ‘=’ Ec
LId = LId ‘,’ ‘id’
LId = LId ‘,’ ‘id’ ‘[‘ Ea ‘]’
LId = LId ‘,’ ‘id’ ‘[‘ Ea ‘]’ = Ec
LId = ‘id’ ‘=’ Ec
LId = ‘id’ ‘[‘ Ea ‘]’ = Ec
LId = ‘id’ ‘[‘ Ea ‘]’
LId = ‘id’

LParamCall = LParamCall ‘,’ Ec
LParamCall = Ec
LParamCall = &

FunDecl = ‘fun’ Type FunName ‘(‘ LParamDecl ‘)’ Body
FunName = ‘id’ | ‘main’

LParamDecl = LParamDecl ‘,’ Type 'id' '[' Ea ']'
LParamDecl = LParamDecl ‘,’ Type 'id'
LParamDecl = Type 'id' '[' Ea ']'
LParamDecl = Type 'id'
LParamDecl = &

IdOrFunCall = ‘id’ ‘[‘ Ea ‘]’
IdOrFunCall = 'id' '(' LParamCall ')'
IdOrFunCall = ‘id’

Id = ‘id’ ‘[‘ Ea ‘]’
Id = ‘id’

ProcDecl = ‘proc’ FunName ‘(‘ LParamDecl ‘)’ Body
```

```
      Body = ‘{‘ BodyPart ‘}’

      BodyPart = DeclId BodyPart
      BodyPart = Command BodyPart
      BodyPart = ‘id’ ‘(‘ LParamCall ‘)’ ‘;’ BodyPart
      BodyPart = ‘return’ Return ‘;’
      BodyPart = LIdAttr ‘;’ BodyPart
      BodyPart = &

      LIdAttr = LIdAttr ‘,’ ‘id’ ‘=’ Ec
      LIdAttr = LIdAttr ‘,’ ‘id’ ‘[‘ Ea ‘]’ '=' Ec
      LIdAttr = 'id' '=' Ec
      LIdAttr = ‘id’ ‘[‘ Ea ‘]’ '=' Ec

      Return = Ec //somente admissível se for função
      Return = & //somente admissível se for procedimento

      Command = ‘print’ ‘(‘ ‘constStr’ PrintLParam ‘)’ ‘;’
      Command = ‘scan’ ‘(‘ ScanLParam ‘)’ ‘;’
      Command = ‘whileLoop’ ‘(‘ Eb ‘)’ Body
      Command = ‘forLoop’ ‘(‘ ‘typeInt’ ‘id’ ‘:’ ‘(‘ Ea ‘,’ Ea ‘,’
Ea ‘)’ ’)’ Body
      Command = ‘forLoop’ ‘(‘ ‘typeInt’ ‘id’ ‘:’ ‘(‘ Ea ‘,’ Ea‘)’
’)’ Body
      Command = ‘if’ ‘(‘ Eb ‘)’ Body
      Command = ‘if’ ‘(‘ Eb ‘)’ Body LElseIf
      Command = ‘if’ ‘(‘ Eb ‘)’ Body LElseIf ‘else’ Body
      Command = ‘if’ ‘(‘ Eb ‘)’ Body ‘else’ Body

      LElseIf = LElseIf ‘ceif’ Body
      LElsefF = ‘ceif’ Body

      PrintLParam = ‘,’ Eb PrintLParam
      PrintLParam = &

      ScanLParam = ScanLParam ‘,’ Id
      ScanLParam = Id
```

Expressões:

```
      Ec = Ec ‘opConcat’ Eb
      Ec = Eb
      Eb = Eb ‘opOr’ Tb
      Eb = Tb
      Tb = Tb ‘opAnd’ Fb
      Tb = Fb
      Fb = Fb OpRel Ra
      Fb = ‘opNot’ Fb
      Fb = Ra
      Ra = Ra OpRel Ea
      Ra = Ea
      Ea = Ea ‘opAdd’ Ta
      Ea = Ea ‘opSub’ Ta
      Ea = Ta
      Ta = Ta ‘opMult’ Pa
      Ta = Ta ‘opDiv’ Pa
      Ta = Pa
      Pa = Pa ‘opPow’ Fa
      Pa = Fa
      Fa = ‘(‘ Ec ‘)’
      Fa = ‘opSub’ Fa
      Fa = IdOrFunCall | ‘cteInt’ | ‘cteFloat’ | ‘cteBool’ |
‘cteString’ | ‘cteChar’
      OpRel = ‘opGreater’ | ‘opLesser’ | ‘opGreq’ | ‘opLeq’
```