Gramática LL(1)

Ponto de Início de 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 = Id AttrOpt LIdr
      LIdr = ',' Id AttrOpt LIdr
      LIdr = &

      Id = ‘id’ ArrayOpt

      ArrayOpt = ‘[‘ Ea ‘]’
      ArrayOpt = &

      AttrOpt = '=' Ec
      AttrOpt = &

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

      LParamCall = Ec LParamCallr
      LParamCall = &
      LParamCallr = ',' Ec LParamCallr
      LParamCallr = &

      LParamDecl = Type 'id' ArrayOpt LParamDeclr
      LParamDecl = &
      LParamDeclr = ‘,’ Type ‘id’ ArrayOpt LParamDeclr
      LParamDeclr = &

      ProdDecl = ‘proc’ FunName ‘(‘ LParamDecl ‘)’ Body

      Body = ‘{‘ BodyPart ‘}’

      BodyPart = DeclId BodyPart
      BodyPart = Command BodyPart
      BodyPart = BodyPartr ‘;’ BodyPart
```

```
	BodyPart = ‘return’ Return ‘;’
	BodyPart = &

	BodyPartr = ‘id’ ParamAttr
	ParamAttr = ‘(‘ LParamCall ‘)’
	ParamAttr = ‘[‘ Ea ‘]’ ‘=’ Ec LAttr
	ParamAttr = ‘=’ Ec LAttr
	LAttr = ‘,’ Id ‘=’ Ec LAttr
	LAttr = &

	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’ ForParams
	Command = ‘if’ ‘(‘ Eb ‘)’ Body Ifr

	PrintLParam = ‘,’ Ec PrintLParam
	PrintLParam = &

	ScanLParam = ‘id’ ArrayOpt ScanLParamr
	ScanLParamr = ‘,’ ‘id’ ArrayOpt ScanLParamr
	ScanLParamr = &

	ForParams = ‘(‘ ‘typeInt’ ‘id’ ‘:’ ‘(‘ Ea ‘,’ Ea ForStep ‘)’
‘)’ Body
	ForStep = ‘,’ Ea
	ForStep = &

	Ifr = ‘ceif’ ‘(‘ Eb ‘)’ Body Ifr
	Ifr = ‘else’ Body
	Ifr = &

	Ec = Eb Ecr
	Ecr = ‘opConcat’ Eb Ecr
	Ecr = &
	Eb = Tb Ebr
	Ebr = ‘opOr’ Tb Ebr
	Ebr = &
	Tb = Fb Tbr
	Tbr = ‘opAnd’ Fb Tbr
	Tbr = &
	Fb = ‘opNot’ Fb
	Fb = Ra Fbr
```

```
      Fbr = ‘opGreater’ Ra Fbr
      Fbr = ‘opLesser’ Ra Fbr
      Fbr = ‘opGreq’ Ra Fbr
      Fbr = ‘opLeq’ Ra Fbr
      Fbr = &
      Ra = Ea Rar
      Rar = ‘opEquals’ Ea Rar
      Rar = ‘opNotEqual’ Ea Rar
      Rar = &
      Ea = Ta Ear
      Ear = ‘opAdd’ Ta Ear
      Ear = ‘opSub’ Ta Ear
      Ear = &
      Ta = Pa Tar
      Tar = ‘opMult’ Pa Tar
      Tar = ‘opDiv’ Pa Tar
      Tar = &
      Pa = Fa Par
      Par = ‘opPow’ Fa Par
      Par = &
      Fa = ‘(‘ Ec ‘)’
      Fa = ‘opSub’ Fa
      Fa = IdOrFunCall | ‘cteInt’ | ‘cteFloat’ | ‘cteBool’ |
‘cteString’ | ‘cteChar’
      IdOrFunCall = 'id' IdOrFunCallr
      IdOrFunCallr = '(' LParamCall ')'
      IdOrFunCallr = '[' Ea ']'
```