GRAMATICA LSI-132 – com ações semânticas

```
<programa> ::= programa id #01 ';' <bloco> '.' ;

#01 – Efetua inicializações:
      Nível Atual (NA:=1)), Deslocamento (:=0), Variáveis
      de Contexto, etc..
      Insere id na Tabela de Símbolos (TS) juntamente com
      seus atributos (categoria = id-programa e nível = NA)

<bloco> ::= <dcl_const> <dcl_var> <dcl_procs> <listacomando> ;

<dcl_const> ::= const id #2 "=" <constante> #3 ";" <dcl_const> | î ;

#02 – Se id já está declarado no NA,
      então ERRO("Id já declarado")
      senão insere id na TS, junto com seus atributos
            (categoria = constante e nível = NA)
#03 – Insere atributos Tipo-Const e Val-Const na TS

<dcl-var> ::= var #04 <lid> #05':' <tipo> #06 ';' <dcl_var>
            | î ;

#04 – seta contextoLID para "decl"
      Marca pos. do primeiro id da lista (relativa a TS)
#05 – Marca pos. do último id da lista (relativa a TS)
#06 – Preenche atributos na TS dos id's da lista,
      considerando categoria "variável" e TipoAtual

<dcl-procs> ::= <dcl-proc> ';' <dcl-procs> | î ;

<dcl-proc>::= proc id #07 <par-formais> #09 ';' <bloco> #11
         | funcao id #08 <par-formais>  #09 ':'
                  <tipo-pre-definido> #10';' <bloco> #11 ;

#07 – Se id do procedimento já está declarado no NA,
      então ERRO("Id já declarado")
      senão insere id na TS, junto com seus atributos
            zera número de parâmetros Formais (NPF)
            incrementa nível atual (NA := NA + 1)
```

#08 – idem a ação #07, para função.

#09 – Atualiza num. de par. Formais (NPF) na TS

#10 – Atualiza tipo do resultado da função na TS

#11 – Retira da TS as variáveis declaradas localmente
Atualiza nível atual (NA := NA – 1)

<par-formais> ::= '(' <mp_par> #12 <lid> #13 ":"
<tipo_pre_definido> #14 <rep_par> ")" | î

<rep_par> ::= ";" <mp_par> #12 <lid> #13 ":" <tipo_pre_definido> #14
<rep_par> | î ;

<mp_par> ::= ref #15 | val #16 ;

#12 - Seta contextoLID para "par-formal"
Marca pos. do primeiro id da lista (relativa a TS)

#13 – Marca pos. do último id da lista (relativa a TS)

#14 – Preenche atributos dos id's da lista de parâmetros, considerando categ. = "Parâmetro", TipoAtual e MPP. Insere todos os parâmetros na ListaPar (para ser usada na chamada de procedimentos)

#15 – Seta MPP para "referência"

#16 – Seta MPP para "valor"

<lid> ::= id #17 <rep_lid> ;

<rep_lid> ::= "," <lid> | î ;

#17 – Se contextoLID = "decl"
entao se id já declarado no NA
então ERRO("Id já declarado")
senão insere id na TS

- Se contextoLID = "par-formal"
entao se id já declarado no NA
então ERRO ("Identificador já declarado")

```
                senão incrementa NPF; insere id na TS

      - Se contextoLID = "leitura"
        entao se id não está declarado no NA
              então ERRO ("Identificador não declarado")
              senão Se id é da categ. variável ou parâmetro
                    então se tipo <> pré-definido
                          entao ERRO("tipo de id inválido")
                          senão (* G Código para leitura *)
                   senao ERRO("apenas var. podem ser lidas")

<tipo> ::=  <tipo_pre_definido>

       | cadeia "[" <constante> #18 "]"

       | vetor  "[" <constante> #19  ".."  <constante> #20
<dimensao2> "]" de  <tipo_pre_definido> #21  ;


<dimensao2> ::= "," #22 <constante> #19 ".." <constante> #20
             |   î #23;

#18 - Se TipoConst <> "inteiro"
      então ERRO("esperava-se uma constante inteira")
      senão se ValConst > 256
            então ERRO("tam.da cadeia > que o permitido")
            senão TipoAtual := "cadeia"

#19 - Se TipoConst <> "inteiro" e <> "caracter"
      então ERRO("tipo do índice inválido")
      senão guarda Tipo e valor do limite inferior

#20 - Se TipoConst <> Tipo do limite inferior
      então ERRO("Ctes do interv. devem ser de mesmo tipo")
      senão se ValConst <= Valor do limite inferior
            então ERRO("Lim. Sup. Deve ser > que L. Inf.")
            senão guarda Tipo e Valor do L. Superior

#21 - TipoElementos  := TipoAtual
      TipoAtual := "vetor"
      Se número de dimensões = 2
      Então Registra informações sobre a segunda dimensão
```

```
#22 - Registra inf. sobre a primeira dimensão
      Seta número de dimensões para 2

#23 - Registra inf. sobre a primeira dimensão
      Seta número de dimensões para 1


<constante> ::= id #24 |  <constante_explicita> ;

#24 - Se id não está declarado
      então ERRO("Id   não declarado")
      senão se categoria de id <> constante
            entao ERRO ("Esperava-se um id de Constante")
            senão TipoConst = Tipo do id-constante
                  ValConst = Valor da constante id

<tipo-pre-definido> ::=  inteiro #25 | real #26
                       |  booleano #27 | caracter #28 ;

#25 - TipoAtual := "inteiro"
#26 - TipoAtual := "real"
#27 - TipoAtual := "booleano"
#28 - TipoAtual := "caracter"

<listacomando> ::= "{"   <comando> <replistacomando> "}"
<replistacomando> ::= ";" <comando> <replistacomando> | î ;

<comando> ::= id #29  <rcomid>
            | <listacomando>
            | se <expressao> #30  entao <comando>
                                     <senaoparte>
            | enquanto <expressao> #30 faca <comando>

            | leia '(' #31 <lid>  ')'

            | escreva '(' <expressao>  #32 <rep-lexpr> ')'

            | î ;
<senaoparte> ::= senao  <comando>  | î ;
```

```
#29 - Se id não está declarado (não esta na TS)
      então ERRO("Identificador não declarado")
      senão guarda posição de id (POSID)

#30 - Se TipoExpr <> "booleano" e <> "inteiro"
      então ERRO("Tipo inválido da expressão")
      senao (* G. Código *)

#31 - Seta contextoLID para "leitura"

#32 - Seta contextoEXPR para "impressão"
      se TipoExpr <> inteiro/real/caracter/cadeia
      então ERRO("tipo invalido para impressão")
      senão (* G. Código *)

<rcomid> ::= #33 ':=' <expressao> #34

          |'[' #35 <expressao> #36 <expressao2> #37']'
                                          ':=' <expressao> #34

          | '(' #38  <expressao> #39  <rep-lexpr> ')' #40

          | î #41;

#33 - Se categoria de id = "Variável" ou "Parâmetro"
       então se tipo de id = "vetor"
             então ERRO ("id. Deveria ser indexado")
             senão TipoLadoEsq := tipo de id
       senão se categoria de id = "função"
             então se fora do escopo da função id
                   então ERRO("fora do escopo da função")
                   senão TipoLadoEsq := tipo da função
              senão ERRO ("id. deveria ser var/par/função")

#34 - se TipoExpr não compatível com tipoLadoesq
      então ERRO ("tipos incompatíveis")
      senão (* G. Código *)

#35 - se categoria de id <> "variável"
      então ERRO ("esperava-se uma variável")
      senao se tipo de id <> vetor e <> de cadeia
```

```
            então ERRO("apenas vetores e cadeias podem
                        ser indexados")
            senão TipoVarIndexada = tipo (vetor/cadeia)

#36 - Seta número de índices para 1
      se TipoVarIndexada = vetor
      então se TipoExpr <> TipoIndice da dimensão 1
            então ERRO("tipo do índice inválido")
            senão TipoLadoEsq := TipoElementos (do vetor)
      senao (* é cadeia *)
            se TipoExpr <> "inteiro"
            então ERRO("índice deveria ser inteiro")
            senão TipoLadoEsq := caracter


#37 - se número de índices = 2
      Então se TipoVarIndexada = cadeia
            então ERRO("Cadeia só pode ter 1 índice")
            senão se num-dimensoes do vetor <> 2
                  então ERRO ("Vetor é uni-dimensional")
                  senão se TipoExpr <> TipoIndice da dim 2
                        então ERRO("tipo índice inválido")
                        senão TipoLadoEsq := TipoElementos
      senão se número de dimensões = 2
            então ERRO("Vetor é bi-dimensional")

#38 - se categoria de id <> procedure
      então ERRO("id deveria ser uma procedure")

#39 - NPA := 1 (Número de Parâmetros Atuais)
      seta contextoEXPR para "par-atual"
      Verifica se existe Parâmetro Formal correspondente e
      se o tipo e o MPP são compatíveis

#40 - se NPA = NPF
      então (* G. Código para chamada de proc*)
      senão ERRO("Erro na quantidade de parâmetros")

#41 - se categoria de id <> procedure
      então ERRO("id deveria ser uma procedure")
      senão se NPF <> 0
```

então ERRO("Erro na quant.de parâmetros")
senão (* G. Código p/ chamada de proc. *)

<expressao2> ::= "," #42 <expressão> | î ;

<rep-lexpr> ::= ',' <expressao> #43 <rep-lexpr> | î ;

#42 - Seta numero de índices para 2

#43 - se ContextoEXPR = "par-atual"
então incrementa NPA e Verifica se existe Parâmetro
Formal correspondente e se o tipo e o MPP
são compatíveis

- se ContextoEXPR = "impressão"
entao se TipoExpr <> inteiro/real/caracter/cadeia
então ERRO("tipo invalido para impressão")
senão (* G. Código para impressão *)

<expressao> ::= <expsimp> #44 <resto-expressao> ;

<resto-expressao> ::= <oprel> <expsimp> #45 | î ;

#44 - TipoExpr := TipoExpSimples

#45 - Se TipoExpSimples incompatível com TipoExpr
então ERRO ("Operandos incompatíveis")
senão TipoExpr := "booleano"

<oprel> ::= '=' #46 | '<' #47 | '>' #48

| '>=' #49 | '<=' #50 | '<>' #51 ;

#46 a #51 - Guarda Op. Rel. para futura Geração de Código

<expsimp> ::= #71 <termo> #52 <repexpsimp> ;

<repexpsimp>::= <opadd> #53 <termo> #54 <repexpsimp> | î;

#52 - TipoExpSimples := TipoTermo

#53 - Se operador não se aplica a TipoExpSimples

```
        então ERRO("Operador e Operando incompatíveis")

 #54 - Se TipoTermo incompatível com TipoExpSimples
       então ERRO ("Operandos incompatíveis")
       senão TipoExpSimples := tipo do res. da operação
             (* G. Código de acordo com oppad *)


<opadd> ::= '+' #55 | '-' #56  | ou #57 ;

#55 a #57 - guarda operador para futura G. código


<termo> ::= <fator> #58 <reptermo> ;

<reptermo> ::= <opmult> #59 <fator> #60 <reptermo> | î ;

#58 - TipoTermo := TipoFator

#59 - Se operador não se aplica a TipoTermo
      então ERRO("Operador e Operando incompatíveis")

#60 - Se TipoFator incompatível com TipoTermo
       então ERRO ("Operandos incompatíveis")
       senão TipoTermo := tipo do res. da operação
             (* G. Código de acordo com opmult *)


<opmult> ::= '*' #64 | '/' #65 | e #66  ;

#64 a #66 - guarda operador para futura G. Código


<fator> ::= nao  #67 <fator> #68

         | "-" #69 <fator> #70

         |  "(" #71 <expressao> ")" #72

         | id #29 <rvar> #73

         | <constante_explicita>  #74;
```

```
#67 - se OpNega
      então ERRO("Operadores "não" consecutivos")
      Senão OpNega := true

#68 - Se TipoFator <> "booleano"
      então ERRO("Op. 'não' exige operando booleano")

#69 - se OpUnario
      então ERRO("Ops. "unario" consecutivos")
      Senão OpUnario := true

#70 - Se TipoFator <> "inteiro" ou de "real"
      então ERRO("Op. '-/+' exige operando numérico")

#71 - OpNega := OpUnario := false

#72 - TipoFator := TipoExpr

#73 - TipoFator := TipoVar

#74 - TipoFator := TipoCte


<rvar> ::= '(' #75  <expressao> #39 <rep-lexpr> ')' #76

       | '[' #35 <expressao> #77 <expressao2> #78 ']'

       | î #79 ;

#75 - se categoria de id <> função
      então ERRO("id deveria ser uma função")

#76 - se NPA = NPF
      então TipoVar := Tipo do resultado da função
            (* G. Código chamada de função *)
      senão ERRO("Erro na quantidade de parâmetros")

#77 - Seta número de índices para 1
      se TipoVarIndexada = vetor
      então se TipoExpr <> TipoIndice da dimensão 1
            então ERRO("tipo do índice inválido")
            senão TipoVar := TipoElementos do vetor
      senao (* é cadeia *)
            se TipoExpr <> "inteiro"
            então ERRO("índice deveria ser inteiro")
            senão TipoVar := caracter
```

```
#78 - se número de índices = 2
      Então se TipoVarIndexada = cadeia
            então ERRO("Cadeia só pode ter 1 índice")
            senão se num-dimensoes do vetor <> 2
                  então ERRO ("Vetor é uni-dimensional")
                  senão se TipoExpr <> TipoIndice da dim 2
                        então ERRO("tipo índice inválido")
                        senão TipoVar:=TipoElem. do vet
      senão se num. De dimensões = 2
            então ERRO("Vetor é bi-dimensional")

 #79 - se categoria de id = "variável" ou "Parâmetro"
       então se tipo de id = "vetor"
             então ERRO("vetor deve ser indexado")
             senão TipoVar := Tipo de id
       senão se categoria de id = "função"
             então se NPF <> 0
                   então ERRO("Erro na quantidade de par.")
                   senão (* G. Código *)
                         TipoVar := Tipo res. da função
             Senão se categoria de id = "constante"
                   então TipoVar:= Tipo do id de Constante
                   Senão ERRO("esperava-se var,
                               id-função ou constante")

<constante_explicita>::= num-int #80 | num-real #81
                       | falso #82   | verdadeiro #83
                       | literal #84  ;

 #80 a #84 - TipoCte := tipo da constante
             ValCte := valor da constante
```