- **Gramática AVI LL1**
    - S = funDec Type FunName ‘[‘ Params ‘]’ Body S
    - S = Decl S
    - S = ε

- **Function Declaration**
    - FunName = ‘id’
    - FunName = ‘main’
    - Params = Type ‘id’ ArrayOpt Paramsr
    - Params = ε
    - Paramsr = ‘,’ Type ‘id’ ArrayOpt Paramsr
    - Paramsr = ε
    - FunCall = ‘[‘ Lec ‘]’ ‘;’
    - Return = ‘return’ Ec ‘;’

- **Variable Declaration**
    - Decl = Type LI
    - LI = ‘id’ ArrayOpt Inst
    - LIr LIr = ‘,’ ‘id’ ArrayOpt Inst LIr
    - LIr = ‘;’

- **Instantiating Variables:**
    - Inst = ‘atrib’ Inr
    - Inst = ε
    - Inr = ArrayOpt
    - Inr = Fc

- **Id**
    - Id = ‘id’ Idr
    - Idr = ArrayOpt
    - Idr = FunCall

- **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 IdLr
  - IdLr = ‘,’ ‘id’ ArrayAccess IdLr
  - IdLr = ε

- **Body**
  - Body = ‘{‘ BodyScope ‘}’
  - BodyScope = Decl BodyScope
  - BodyScope = Atr ‘;’ BodyScope
  - BodyScope = Command BodyScope
  - BodyScope = Return Atr ‘;’
  - BodyScope = ε

- **List of Expressions**
  - Lec = Fc Lecr
  - Lec = ε
  - Lecr = ‘,’ Ec Lecr

- **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’