**Original**

PROGRAM → (STATEMENT | FUNCLIST)?

FUNCLIST → FUNCDEF FUNCLIST | FUNCDEF

FUNCDEF → def ident(PARAMLIST){STATELIST}

PARAMLIST → ((int | float | string) ident, PARAMLIST | (int | float | string) ident)?

STATEMENT → (VARDECL; | ATRIBSTAT; | PRINTSTAT; | READSTAT; | RETURNSTAT; | IFSTAT | FORSTAT | {STATELIST} | break; | ;)

VARDECL → (int | float | string) ident ([int constant]) ∗

ATRIBSTAT → LVALUE = (EXPRESSION | ALLOCEXPRESSION | FUNCCALL)

FUNCCALL → ident(PARAMLISTCALL)

PARAMLISTCALL → (ident, PARAMLISTCALL | ident)?

PRINTSTAT → print EXPRESSION

READSTAT → read LVALUE

RETURNSTAT → return

IFSTAT → if(EXPRESSION ) STATEMENT (else STATEMENT)?

FORSTAT → for(ATRIBSTAT; EXPRESSION; ATRIBSTAT) STATEMENT

STATELIST → STATEMENT (STATELIST)?

ALLOCEXPRESSION → new (int | float | string) ([ NUMEXP RESSION ]) +

EXPRESSION → NUMEXPRESSION(( < | > | <= | >= | == | ! =) NUMEXPRESSION)?

NUMEXPRESSION → TERM ((+ |−) TERM) ∗

TERM → UNARYEXPR(( ∗ | / | %) UNARYEXPR) ∗

UNARYEXPR → ((+ |−))? FACTOR

FACTOR → (int\_constant | float\_constant | string\_constant | null | LVALUE |(NUMEXP RESSION))

LVALUE → ident([NUMEXP RESSION]) ∗

**Modificações**

PROGRAM → (STATEMENT | FUNCLIST)?

FUNCLIST → FUNCDEF FUNCLIST | FUNCDEF

FUNCDEF → def ident(PARAMLIST){STATELIST}

PARAMLIST → ((int | float | string) ident, PARAMLIST | (int | float | string) ident)?

STATEMENT → (VARDECL; | ATRIBSTAT; | PRINTSTAT; | READSTAT; | RETURNSTAT; | IFSTAT | FORSTAT | WHILESTAT | {STATELIST} | break; | ;)

VARDECL → (int | float | string) ident ([int constant]) ∗

ATRIBSTAT → LVALUE([NUMEXPRESSION])? = (EXPRESSION | ALLOCEXPRESSION | FUNCCALL)

FUNCCALL → ident(PARAMLISTCALL)

PARAMLISTCALL → (ident, PARAMLISTCALL | ident)?

PRINTSTAT → print EXPRESSION

READSTAT → read LVALUE

RETURNSTAT → return (ident | EXPRESSION)?

IFSTAT → if(EXPRESSION ) STATEMENT (else STATEMENT)?

FORSTAT → for(ATRIBSTAT?; EXPRESSION?; ATRIBSTAT?) STATEMENT

WHILESTAT → while(EXPRESSION) STATEMENT

STATELIST → STATEMENT (STATELIST)?

ALLOCEXPRESSION → new (int | float | string) ([NUMEXPRESSION]) +

EXPRESSION → NUMEXPRESSION(( < | > | <= | >= | == | ! =) NUMEXPRESSION)?

NUMEXPRESSION → TERM ((+ |−) TERM) ∗

TERM → UNARYEXPR(( ∗ | / | %) UNARYEXPR) ∗

UNARYEXPR → ((+ |−))? FACTOR

FACTOR → (int\_constant | float\_constant | string\_constant | null | LVALUE | (NUMEXP RESSION))

LVALUE → ident([NUMEXP RESSION]) ∗

**Forma Convencional**

1. **Transformar para definição de gramática convencional**

PROGRAM → STATEMENT | FUNCLIST | &

FUNCLIST → FUNCDEF FUNCLIST | FUNCDEF

FUNCDEF → def ident(PARAMLIST){STATELIST}

TYPES → int | float | string

PARAMLIST → (int | float | string) ident, PARAMLIST | TYPES ident | &

STATEMENT → VARDECL; | ATRIBSTAT; | PRINTSTAT; | READSTAT; | RETURNSTAT; | IFSTAT | FORSTAT | WHILESTAT | {STATELIST} | break; | ;

VARDECL → TYPES ident VARDECL’

VARDECL’ → [int constant] VARDECL’ | &

ATRIBSTAT → LVALUE ATRIBSTAT’ = ATRIBSTAT’’

ATRIBSTAT’ → [NUMEXPRESSION] | &

ATRIBSTAT’’ → EXPRESSION | ALLOCEXPRESSION | FUNCCALL

FUNCCALL → ident(PARAMLISTCALL)

PARAMLISTCALL → ident, PARAMLISTCALL | ident | ?

PRINTSTAT → print EXPRESSION

READSTAT → read LVALUE

RETURNSTAT → return RETURNSTAT’

RETURNSTAT’ → ident | EXPRESSION | &

IFSTAT → if(EXPRESSION) STATEMENT IFSTAT’

IFSTAT’ → else STATEMENT | &

FORSTAT → for(FORSTAT'; FORSTAT’’; FORSTAT’) STATEMENT

FORSTAT’ → ATRIBSTAT | &

FORSTAT’’ → EXPRESSION | &

WHILESTAT → while(EXPRESSION) STATEMENT

STATELIST → STATEMENT STATELIST’

STATELIST’ → STATELIST | &

ALLOCEXPRESSION → new TYPES [NUMEXPRESSION] ALLOCEXPRESSION’

ALLOCEXPRESSION’ → [NUMEXPRESSION] ALLOCEXPRESSION’ | &

EXPRESSION → NUMEXPRESSION EXPRESSION’

EXPRESSION’ → COMPOPERATOR NUMEXPRESSION | &

COMPOPERATOR → < | > | <= | >= | == | ! =

NUMEXPRESSION → TERM NUMEXPRESSION’

NUMEXPRESSION’ → ADDSUB TERM | &

ADDSUB → + | −

TERM → UNARYEXPR TERM’

TERM’ → MULTDIV UNARYEXPR | &

MULTDIV → ∗ | / | %

UNARYEXPR → UNARYEXPR’ FACTOR

UNARYEXPR’ → ADDSUB | &

FACTOR → int\_constant | float\_constant | string\_constant | null | LVALUE | (NUMEXP RESSION)

LVALUE → ident LVALUE’

LVALUE’ → [NUM\_EXPRESSION] LVALUE’ | &