**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; | FUNCCALL ;| 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 | (NUMEXPRESSION))

LVALUE → ident([NUMEXP RESSION]) ∗

**Forma Convencional**

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

PROGRAM → STATEMENT | FUNCLIST | ϵ

FUNCLIST → FUNCDEF FUNCLIST | FUNCDEF

FUNCDEF → def ident ( PARAMLIST ) { STATELIST }

TYPES → int | float | string

PARAMLIST → TYPES ident , PARAMLIST | TYPES ident | ϵ

STATEMENT → VARDECL ; |

ATRIBSTAT ; |

PRINTSTAT ; |

READSTAT ; |

FUNCCALL ; |

RETURNSTAT ; |

IFSTAT |

FORSTAT |

WHILESTAT |

{ STATELIST } |

break; | ;

VARDECL → TYPES ident VARDECL’

VARDECL’ → [ int\_constant ] VARDECL’ | ϵ

ATRIBSTAT → LVALUE ATRIBSTAT’ = ATRIBSTAT’’

ATRIBSTAT’ → [ NUMEXPRESSION ] | ϵ

ATRIBSTAT’’ → EXPRESSION | ALLOCEXPRESSION | FUNCCALL

FUNCCALL → ident ( PARAMLISTCALL )

PARAMLISTCALL → ident , PARAMLISTCALL | ident | ϵ

PRINTSTAT → print EXPRESSION

READSTAT → read LVALUE

RETURNSTAT → return RETURNSTAT’

RETURNSTAT’ → ident | EXPRESSION | ϵ

IFSTAT → if ( EXPRESSION ) STATEMENT IFSTAT’

IFSTAT’ → else STATEMENT | ϵ

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

FORSTAT’ → ATRIBSTAT | ϵ

FORSTAT’’ → EXPRESSION | ϵ

WHILESTAT → while ( EXPRESSION ) STATEMENT

STATELIST → STATEMENT STATELIST’

STATELIST’ → STATELIST | ϵ

ALLOCEXPRESSION → new TYPES [ NUMEXPRESSION ] ALLOCEXPRESSION’

ALLOCEXPRESSION’ → [ NUMEXPRESSION ] ALLOCEXPRESSION’ | ϵ

EXPRESSION → NUMEXPRESSION EXPRESSION’

EXPRESSION’ → COMPOPERATOR NUMEXPRESSION | ϵ

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

NUMEXPRESSION → TERM NUMEXPRESSION’

NUMEXPRESSION’ → ADDSUB TERM | ϵ

ADDSUB → + | −

TERM → UNARYEXPR TERM’

TERM’ → MULTDIV UNARYEXPR TERM’ | ϵ

MULTDIV → ∗ | / | %

UNARYEXPR → UNARYEXPR’ FACTOR

UNARYEXPR’ → ADDSUB | ϵ

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

LVALUE → ident LVALUE’

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

1. **Remover recursão à esquerda**

PROGRAM → STATEMENT

| FUNCLIST

| ϵ

FUNCLIST → FUNCDEF FUNCLIST

| FUNCDEF

FUNCDEF → def ident ( PARAMLIST ) { STATELIST }

TYPES → int

| float

| string

PARAMLIST → int ident , PARAMLIST

| float ident , PARAMLIST

| string ident , PARAMLIST

| int ident

| float ident

| string ident

| ϵ

STATEMENT → VARDECL ;

| ATRIBSTAT ;

| PRINTSTAT ;

| READSTAT ;

| FUNCCALL ;

| RETURNSTAT ;

| IFSTAT

| FORSTAT

| WHILESTAT

| { STATELIST }

| break ;

| ;

VARDECL → int ident VARDECL’

| float ident VARDECL’

| string ident VARDECL’

VARDECL’ → [ int\_constant ] VARDECL’

| ϵ

ATRIBSTAT → LVALUE ATRIBSTAT’ = ATRIBSTAT’’

ATRIBSTAT’ → [ NUMEXPRESSION ]

| ϵ

ATRIBSTAT’’ → EXPRESSION

| ALLOCEXPRESSION

| FUNCCALL

FUNCCALL → ident ( PARAMLISTCALL )

PARAMLISTCALL → ident , PARAMLISTCALL

| ident

| ϵ

PRINTSTAT → print EXPRESSION

READSTAT → read LVALUE

RETURNSTAT → return RETURNSTAT’

RETURNSTAT’ → ident

| EXPRESSION

| ϵ

IFSTAT → if ( EXPRESSION ) STATEMENT IFSTAT’

IFSTAT’ → else STATEMENT

| ϵ

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

FORSTAT’ → LVALUE ATRIBSTAT’ = ATRIBSTAT’’

| ϵ

FORSTAT’’ → EXPRESSION

| ϵ

WHILESTAT → while ( EXPRESSION ) STATEMENT

STATELIST → int ident VARDECL’ ; STATELIST’

| float ident VARDECL’ ; STATELIST’

| string ident VARDECL’ ; STATELIST’

| LVALUE ATRIBSTAT’ = ATRIBSTAT’’ ; STATELIST’

| print EXPRESSION ; STATELIST’

| read LVALUE ; STATELIST’

| ident ( PARAMLISTCALL ) ; STATELIST’

| return RETURNSTAT’ ; STATELIST’

| if ( EXPRESSION ) STATEMENT IFSTAT’ STATELIST’

| for ( FORSTAT' ; FORSTAT’’ ; FORSTAT’ ) STATEMENT STATELIST’

| while ( EXPRESSION ) STATEMENT STATELIST’

| { STATELIST } STATELIST’

| break ; STATELIST’

| ; STATELIST’

STATELIST’ → int ident VARDECL’ ; STATELIST’

| float ident VARDECL’ ; STATELIST’

| string ident VARDECL’ ; STATELIST’

| LVALUE ATRIBSTAT’ = ATRIBSTAT’’ ; STATELIST’

| print EXPRESSION ; STATELIST’

| read LVALUE ; STATELIST’

| ident ( PARAMLISTCALL ) ; STATELIST’

| return RETURNSTAT’ ; STATELIST’

| if ( EXPRESSION ) STATEMENT IFSTAT’ STATELIST’

| for ( FORSTAT' ; FORSTAT’’ ; FORSTAT’ ) STATEMENT STATELIST’

| while ( EXPRESSION ) STATEMENT STATELIST’

| { STATELIST } STATELIST’

| break ; STATELIST’

| ; STATELIST’

| ϵ

ALLOCEXPRESSION → new TYPES [ NUMEXPRESSION ] ALLOCEXPRESSION’

ALLOCEXPRESSION’ → [ NUMEXPRESSION ] ALLOCEXPRESSION’

| ϵ

EXPRESSION → NUMEXPRESSION EXPRESSION’

EXPRESSION’ → COMPOPERATOR NUMEXPRESSION

| ϵ

COMPOPERATOR → <

| >

| <=

| >=

| ==

| !=

NUMEXPRESSION → TERM NUMEXPRESSION’

NUMEXPRESSION’ → ADDSUB TERM

| ϵ

ADDSUB → +

| −

TERM → UNARYEXPR TERM’

TERM’ → MULTDIV UNARYEXPR TERM’

| ϵ

MULTDIV → ∗

| /

| %

UNARYEXPR → UNARYEXPR’ FACTOR

UNARYEXPR’ → +

| −

| ϵ

FACTOR → int\_constant

| float\_constant

| string\_constant

| null

| LVALUE

| ( NUMEXPRESSION )

LVALUE → ident LVALUE’

LVALUE’ → [ NUM\_EXPRESSION ] LVALUE’

| ϵ

1. **Fatorar gramática**

PROGRAM → STATEMENT

| FUNCLIST

| ϵ

FUNCLIST → FUNCDEF FUNCLIST'

FUNCLIST' → FUNCLIST

| ϵ

FUNCDEF → def ident ( PARAMLIST ) { STATELIST }

TYPES → int

| float

| string

PARAMLIST → string ident PARAMLIST'

| float ident PARAMLIST'

| int ident PARAMLIST'

| ϵ

PARAMLIST' → , PARAMLIST

| ϵ

STATEMENT → VARDECL ;

| ATRIBSTAT ;

| PRINTSTAT ;

| READSTAT ;

| FUNCCALL ;

| RETURNSTAT ;

| IFSTAT

| FORSTAT

| WHILESTAT

| { STATELIST }

| break ;

| ;

VARDECL → int ident VARDECL’

| float ident VARDECL’

| string ident VARDECL’

VARDECL’ → [ int\_constant ] VARDECL’

| ϵ

ATRIBSTAT → LVALUE ATRIBSTAT’ = ATRIBSTAT’’

ATRIBSTAT’ → [ NUMEXPRESSION ]

| ϵ

ATRIBSTAT’’ → EXPRESSION

| ALLOCEXPRESSION

| FUNCCALL

FUNCCALL → ident ( PARAMLISTCALL )

PARAMLISTCALL → ident PARAMLISTCALL'

| ϵ

PARAMLISTCALL' → , PARAMLISTCALL

| ϵ

PRINTSTAT → print EXPRESSION

READSTAT → read LVALUE

RETURNSTAT → return RETURNSTAT’

RETURNSTAT’ → ident

| EXPRESSION

| ϵ

IFSTAT → if ( EXPRESSION ) STATEMENT IFSTAT’

IFSTAT’ → else STATEMENT

| ϵ

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

FORSTAT’ → LVALUE ATRIBSTAT’ = ATRIBSTAT’’

| ϵ

FORSTAT’’ → EXPRESSION

| ϵ

WHILESTAT → while ( EXPRESSION ) STATEMENT

STATELIST → int ident VARDECL’ ; STATELIST’

| float ident VARDECL’ ; STATELIST’

| string ident VARDECL’ ; STATELIST’

| LVALUE ATRIBSTAT’ = ATRIBSTAT’’ ; STATELIST’

| print EXPRESSION ; STATELIST’

| read LVALUE ; STATELIST’

| ident ( PARAMLISTCALL ) ; STATELIST’

| return RETURNSTAT’ ; STATELIST’

| if ( EXPRESSION ) STATEMENT IFSTAT’ STATELIST’

| for ( FORSTAT' ; FORSTAT’’ ; FORSTAT’ ) STATEMENT STATELIST’

| while ( EXPRESSION ) STATEMENT STATELIST’

| { STATELIST } STATELIST’

| break ; STATELIST’

| ; STATELIST’

STATELIST’ → int ident VARDECL’ ; STATELIST’

| float ident VARDECL’ ; STATELIST’

| string ident VARDECL’ ; STATELIST’

| LVALUE ATRIBSTAT’ = ATRIBSTAT’’ ; STATELIST’

| print EXPRESSION ; STATELIST’

| read LVALUE ; STATELIST’

| ident ( PARAMLISTCALL ) ; STATELIST’

| return RETURNSTAT’ ; STATELIST’

| if ( EXPRESSION ) STATEMENT IFSTAT’ STATELIST’

| for ( FORSTAT' ; FORSTAT’’ ; FORSTAT’ ) STATEMENT STATELIST’

| while ( EXPRESSION ) STATEMENT STATELIST’

| { STATELIST } STATELIST’

| break ; STATELIST’

| ; STATELIST’

| ϵ

ALLOCEXPRESSION → new TYPES [ NUMEXPRESSION ] ALLOCEXPRESSION’

ALLOCEXPRESSION’ → [ NUMEXPRESSION ] ALLOCEXPRESSION’

| ϵ

EXPRESSION → NUMEXPRESSION EXPRESSION’

EXPRESSION’ → COMPOPERATOR NUMEXPRESSION

| ϵ

COMPOPERATOR → <

| >

| <=

| >=

| ==

| !=

NUMEXPRESSION → TERM NUMEXPRESSION’

NUMEXPRESSION’ → ADDSUB TERM

| ϵ

ADDSUB → +

| −

TERM → UNARYEXPR TERM’

TERM’ → MULTDIV UNARYEXPR TERM’

| ϵ

MULTDIV → ∗

| /

| %

UNARYEXPR → UNARYEXPR’ FACTOR

UNARYEXPR’ → +

| −

| ϵ

FACTOR → int\_constant

| float\_constant

| string\_constant

| null

| LVALUE

| ( NUMEXPRESSION )

LVALUE → ident LVALUE’

LVALUE’ → [ NUM\_EXPRESSION ] LVALUE’

| ϵ