BNF – Linguagem 2021.1

```
VT = {
    program define not variable is natural real char boolean execute set get
to put loop while do true false { } [ ] , + - * / ** % %% == != < > <= >= & |
! ( ) .
}


<program>                  ::= <header-sel> program { <define> <execute> }
<identifier-sel>

<header-sel>               ::= <header>
                            |  ε

<identifier-sel>           ::= <identifier>
                            |  ε

<define>                   ::= define { <variable-block> }
                            |  ε

<variable-block>           ::= <not-variable> <variable-sel-1>
                            |  <variable> <variable-sel-2>
                            |  ε

<variable-sel-1>           ::= <variable>
                            |  ε

<variable-sel-2>           ::= <not-variable>
                            |  ε

<execute>                  ::= execute <command-block>

<not-variable>             ::= not variable <not-variable-decl> <not-
variable-sel>

<not-variable-sel>         ::= <not-variable-decl> <not-variable-sel>
                            |  ε

<not-variable-decl>        ::= <type> is <identifier-list-value> .

<identifier-list-value>    ::= <identifier> <value> <identifier-list-value-1>

<identifier-list-value-1>  ::= , <identifier-list-value>
                            |  ε

<variable>                 ::= variable <variable-decl> <variable-sel>

<variable-sel>             ::= <variable-decl> <variable-sel>
                            |  ε

<variable-decl>            ::= <type> is <identifier-list> .

<identifier-list>          ::= <identifier> <index> <identifier-list-1>

<identifier-list-1>        ::= ,  <identifier-list>
                            |  ε

<set>                      ::= set <expression> to <identifier-list>
```

```
<get>                      ::= get { <identifier-list> }

<put>                      ::= put { <put-list> }

<put-list>                 ::= <identifier-list>
                            |  <value>

<verify>                   ::= verify <expression> is <false-true-sel>

<false-true-sel>           ::= <true-block> <false-true-sel-1>
                            |  <false-block> <false-true-sel-2>

<false-true-sel-1>         ::= <false-block>
                            |  ε

<false-true-sel-2>         ::= <true-block>
                            |  ε

<true-block>               ::= true <command-block>

<false-block>              ::= false <command-block>

<loop>                     ::= loop <command-block> while <expression> is
true

<while>                    ::= while <expression> is true do <command-block>

<command-block>            ::= { <command> <command-list> }

<command-list>             ::= <command> <command-list>
                            |  ε

<command>                  ::= <command-sel> .

<command-sel>              ::= <set>
                            |  <get>
                            |  <put>
                            |  <verify>
                            |  <loop>
                            |  <while>

<expression>               ::= <expr-arith-logical> <expression-sel>

<expression-sel>           ::= == <expr-arith-logical>
                            |  != <expr-arith-logical>
                            |  < <expr-arith-logical>
                            |  > <expr-arith-logical>
                            |  <= <expr-arith-logical>
                            |  >= <expr-arith-logical>
                            |  ε

<expr-arith-logical>       ::= <term2> <less-priority>

<less-priority>            ::= + <term2> <less-priority>
                            |  - <term2> <less-priority>
                            |  | <term2> <less-priority>
                            |  ε
```

```
<term2>                    ::= <term1> <mid-priority>

<mid-priority>             ::= * <term1> <mid-priority>
                            |  / <term1> <mid-priority>
                            |  % <term1> <mid-priority>
                            |  %% <term1> <mid-priority>
                            |  & <term1> <mid-priority>
                            |  ε

<term1>                    ::= <element> <great-priority>

<great-priority>           ::= ** <element> <great-priority>
                            |  ε

<element>                  ::= <identifier> <index>
                            |  <value>
                            |  ( <expression> )
                            |  ! ( <expression> )

<index>                    ::= [ <natural> ]
                            |  ε

<value>                    ::= <natural>
                            |  <real>
                            |  <char>
                            |  <boolean>

<type>                     ::= <natural-type>
                            |  <real-type>
                            |  <char-type>
                            |  <boolean-type>
```

Mensagens Parser

1. `Unexpected token 'token' at line 'line', column 'column'. Expected: 'expected token'.`
2. `Unexpected token 'token' at line 'line', column 'column'. Expected an identifier.`
3. `Unexpected token 'token' at line 'line', column 'column'. Expected a command.`
4. `Invalid expression encountered at line 'line', column 'column'.`
5. `Unexpected token 'token' at line 'line', column 'column'. Expected a constant value (char, natural, real or boolean).`
6. `Unexpected token 'token' at line 'line', column 'column'. Expected: 'char', 'natural', 'real' or 'boolean'.`

Mensagens Lexer

1. `Lexical error at line 'line', column 'column'. The following character 'token' is invalid.`