Alphabet:

1.Upper (A-Z) and lower case letters (a-z) of the English alphabet

2.Decimal digits (0-9);

Lexic:

Special symbols, representing:

- operators + - * / = < <= > >= ==
- separators [ ] ; space newline
- reserved words:

  begin end integer float list in out while if bwhile ewhile bif elif eif in

Identifiers

a sequence of letters and digits, such that the first character is a letter; the rule is:

identifier ::= letter | letter{letter}{digit}

letter ::= "A" | "B" | ...| "Z" | "a" | "b" | ... | "z"

digit ::= "0" | "1" |...| "9"

Constants

1.integer - rule:

  noconst:="+"no|"-"no|no

  no:=digit{no}

2.float - rule:

floatno := "+" no "," no | "-" no "," no | no "," no

3.character

character:='letter'|'digit'

4.string

constchar:="string"

string:=char{string}

char:=letter|digit

Syntax:

program ::= "begin" vardecl newline cmpdstmt "end"

vardecl ::= typedecl | typedecl newline vardecl

typedecl ::= type identifierlist ";"

identifierlist ::= identifier | identifier ";" identifierlist

type1 ::= "integer" | "float"

listdecl ::= "list" "[" no "]" "[" type1 "]"

type ::= type1|listdecl

cmpdstmt ::= stmt | stmt ";" newline cmpdstmt

stmt ::= simplstmt | structstmt

simplstmt ::= assignstmt | iostmt

assignstmt ::= identifier "=" expression

expression ::= noconst | floatno | identifier

iostmt ::= "in" identifier | "out" identifier | "out" """ string """

structstmt ::= cmpdstmt | ifstmt | whilestmt | forstmt

ifstmt ::= "if" condition ":" newline "bif" stmt "elif" stmt "eif" | "if" condition ":" newline "bif" stmt "eif"

whilestmt ::= "while" condition ":" newline "bwhile" stmt "ewhile"

condition ::= expression RELATION expression

RELATION ::= "<" | "<=" | "==" | "!=" | ">=" | ">"