

Alphabet:

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

b. Decimal digits (0-9);

Lexic:

a.Special symbols, representing:

- operators + - \* / < > <= >=

- separators [] ( ) ; space

- reserved words:

list char while else if then int start end read write is endif endwhile

b.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"

c.constants

1.integer - rule:

noconst = [("("+" | "-")] no | "0"

no = nonzerodig {no | "0"}

nonzerodig = "1" | ... | "9"

2.character

character = 'letter' | 'digit'

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

3.string

constchar = "string"

string = char {string}

char = 'letter | digit

---

+

-

\*

/

<

>

<=  
=  
>=  
(  
)  
[  
]  
;  
list  
char  
while  
else  
if  
then  
int  
start  
end  
read  
write  
is  
endif  
endwhile

---

The words - predefined tokens are specified between " and ":

Sintactical rules: (file Syntax.in)

program = "start" { decllist | stmtlist } "end"

decllist = declaration | declaration ";" decllist

declaration = type " " IDENTIFIER

type = type1 | listdecl

type1 = "int" | "char" | "string"

listdecl = "list" "[" type1 "]"

stmtlist = stmt | stmt ";" stmtlist

stmt = simplstmt | structstmt

simplstmt = assignstmt | iostmt

assignstmt = IDENTIFIER "is" expression

expression = term { ("+" | "-") expression }

term = factor { ("\*" | "/" ) term }

factor = const | IDENTIFIER

const = noconst | character | string

noconst = [ ("+" | "-") ] no | "0"

no = nonzerodig { no | "0" }

nonzerodig = "1" | ... | "9"

character = 'letter' | 'digit'

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

string = "string"

string1 = char {string}

char = 'letter' | digit

iostmt = "read" | "write" "(" IDENTIFIER ")"

structstmt = ifstmt | whilestmt

ifstmt = "if(" condition ") THEN" stmtlist ["else" stmtlist]

whilestmt = "while(" condition ") then" stmtlist

condition = expression RELATION expression

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

\*question: can we have symbols like <,> etc if we didn't include them in the alphabet?