<https://github.com/AntonioSuciu/FLCD>

Alphabet:

a. Uppercase (A-Z) and lowercase (a-z) letters of the English alphabet

b. Decimal digits (0-9)

c. Decimal point (.)

a.Special symbols, representing:

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

- separators [ ] { } ( ) : ; space '' ""

- reserved words:

array char const declare do else for if integer main of read then var while write

b. Identifiers

- a sequence of characters, where

- first character is a letter

- the next characters may be 0 or more letters or digits, in no particular order

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

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

letter

identifier = letter {(letter|digit)}

c. Constants

1. Integers:

integer = [sign] nzd {digit} | "0"

sign = "+" | "-"

nzd = "1" | ... | "9"

numconst = nzd {digit}

2. Characters:

char = letter | digit

charconst = "'" char "'"

string = ``"`` string ``"``

(\*\*) i used `` `` to escape the double quote

stringconst = "char{string}"

3. Floats:

float = (integer "." ( "0" | number) ) | "0" "." number

number = digit {digit}

4. Array

array = identifier "[" identifier "]" | identifier "[" numconst "]"

The words - predefined tokens are specified between double quotes (ex: "predefined", "for")

Syntactic rules:

(\*\*) initialization, declaration..

program = "MAIN" cmpdstmt

IDdecl = "DECLARE" IDENTIFIER type

ARRdecl = "DECLARE" array "array" "[" type "]"

type = "integer" | "char" | "float"

declaration = IDdecl | ARRdecl

(\*\*) repetitive & conditional statements

forstmt = "FOR" IDENTIFIER "IN" forcondition cmpdstmt

forcondition = "(" expression ":" expression ")"

whilestmt = "WHILE" condition cmpdstmt

ifstmt = "IF" condition cmpdstmt ["ELSE" cmpdstmt]

condition = "(" expression relation expression ")"

(\*\*) expression-related

expression = expression "+" term | expression "-" term | term

term = term "\*" factor | term "/" factor | factor

factor = "(" expression ")" | IDENTIFIER | integer | float

relation = "<" | "<=" | "==" | ">=" | ">" | "!="

(\*\*) general statements

assignstmt = IDENTIFIER "=" (expression | charconst | stringconst )

(\*\*) | ARRAY "=" (expression | charconst | stringconst )

iostmt = "READ" "(" identifier ")" | "WRITE" "(" identifier ")"

| "READ" "(" array")" | "WRITE" "(" array ")"

(\*\*) | "WRITE" "(" const ")"

cmpdstmt = "{" stmtlist "}"

stmtlist = stmt | stmt ";" stmtlist

stmt = simplestmt | structstmt

simplstmt = declaration | assignstmt | iostmt

structstmt = cmpdstmt | ifstmt | forstmt | whilestmt

Tokens:

+

-

\*

/

=

<

<=

==

>=

!=

+=

-=

\*=

/=

[

]

{

}

(

)

:

;

space

' '

" "

array

char

const

declare

do

else

for

if

integer

main

of

read

then

var

while

write