a -> Upper (A-Z) and lower case letters (a-z) of the English alphabet
b -> Underline character '_'
c -> Decimal digits (0-9)

~ Lexic ~

a) special symbols, representing:

operators: + _ * // / % < <= == >= > = += -= *= /= && || nu ++ --
separators: [ ] { } ( ) : ? space ; .
reserved words: matrice
caracter
constanta
fa
altfel
daca
in treg
de
program
citeste
atunci
variabila
InTimpCe
scrie
sfarsit
bucla
pentru
start
stop
pas

b) identifiers:

identificator ::= litera | litera{subliniat_litera}{cifra}
litera ::= "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"
subliniat_litera ::= "_" | litera
cifra ::= "0" | "1" | ... | "9"

c) constants
intreg - rule:

intreg ::= "0" | ["-" | "+"] fara-zero-cifra {cifra}
fara-zero-cifra ::= "1" | .. | "9"
cifra ::= "0" | fara-zero-cifra

caracter:

caracter ::= 'litera' | 'cifra' | ' ' | '-'

sir:
sir ::= "multecaractere"
multecaractere ::= caracter | caracter{caracter}

~ Syntax ~

program ::= "var" decllist ";" cmpdstmt "."

decllist ::= declaration | declaration ";" decllist
declaration ::= "variabila" type IDENTIFICATOR

type1 ::= "adevarat_fals" | "caracter" | "numar" | "real"
arraydecl ::= type1 IDENTIFICATOR "[" nr "]"
type ::= type1 | arraydecl

cmpdstmt ::= "{" stmtlist "}"

stmtlist ::= stmt | stmt ";" stmtlist
stmt ::= simplstmt | structstmt
simplstmt ::= assignstmt | iostmt
assignstmt ::= IDENTIFICATOR "=" expression
expression ::= expression symbol term | term
symbol ::= "+" | "-" | "*" | "//" | "/" | "%"
term ::= term symbol factor | factor
factor ::= "(" expression ")" | IDENTIFICATOR | IDENTIFICATOR "[" IDENTIFICATOR | positive-number-const
ant "]"
iostmt ::= "citeste" type IDENTIFICATOR | "listeaza" IDENTIFICATOR
structstmt ::= cmpdstmt | ifstmt | whilestmt | forstmt
ifstmt ::= "daca" "(" conditionlist ")" "?" "atunci" stmtlist "sfarsit" ["altfel" stmtlist "sfarsit"]
whilestmt ::= "InTimpCe" conditionlist "fa" stmtlist "sfarsit"
forstmt ::= "pentru" IDENTIFIER "start" IDENTIFIER "stop" IDENTIFIER ["pas" IDENTIFIER] stmtlist "sfarsit"
conditionlist ::= condition | condition OPERATION condition
condition ::= "(" expression RELATION expression ")"
RELATION ::= "<" | "<=" | "==" | "<>" | ">=" | ">"
OPERATION ::= "%%" | "||"
positive-number-constant ::= "0" | fara-zero-cifra {cifra}
fara-zero-cifra ::= "1" | .. | "9"
cifra ::= "0" | fara-zero-cifra

~ If k is a prime number ~

```
pentru ( intreg i=2;i<k;i++)
 daca (k%i==0)
  cout << "nu este prim"
```