Lexic.txt

Alphabet:

a. Upper case (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:
    - + - * / % (arithmetic)
    - < <= == >= != (relational)
    - = (assignment)
- separators: [ ] ( ) { } : ; space
- reserved words: intreg caracter citeste tipareste repeta pentru daca altfel true false returneaza start

b.identifiers

- a sequence of letters, digits and "_", such that the first character is a letter; the rule is:
- identifier ::= small letter | small letter{letter | digit | "_"}
- small letter ::= "a" | "b" | ... | "z"
- letter ::= "A" | "B" | . ..| "Z" | small letter
- non zero digit ::= "1" |...| "9"
- digit ::= "0" | non zero digit

c.constants

1.integer rule for the sign:
- number const ::= "+"number | "-"number | number
- number ::= non zero digit{digit}

2.caracter
- char ::= 'letter' | 'digit'

3. string
- string ::= {char}
- string const ::= """ {char} """

Syntax.in

program ::= "start" cmpdstmt
decllist ::= declaration | declaration ";" decllist
declaration ::= type " " IDENTIFIER

type1 ::= "intreg" | "caracter"
arraydecl ::= type1 " " identifier"[" nr "]"
type ::= type1|arraydecl

stmtlist ::= stmt | stmt ";" stmtlist
stmt ::= simplstmt | structstmt
simplstmt ::= assignstmt | iostmt | declaration
assignstmt ::= IDENTIFIER "=" expression
expression ::= expression "+" term | expression "-" term | term
term ::= term "*" factor | term "/" factor | term "/" factor | factor
factor ::= "(" expression ")" | IDENTIFIER | CONST

iostmt ::= "citeste" "("IDENTIFIER")" | "scrie" "("IDENTIFIER")" | "scrie" "("CONST")"
structstmt ::= stmtlist | ifstmt | whilestmt
ifstmt ::= "daca" "("condition")" "{"stmt"}" ["altfel" "{"stmt"}"]
whilestmt ::= "repeta" "("condition")" "{"stmt"}"
condition ::= expression RELATION expression
RELATION ::= "<" | "<=" | "==" | "!=" | ">=" | ">"

Token.in

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u

v
w
x
y
z
0
1
2
3
4
5
6
7
8
9
+
-
*
/
%
<
>
=
!=
==
>=
<=
[
]
(
)
{
}
,
;
:
space
'
"
_
start
intreg
caracter
citeste
tipareste
repeta
pentru
daca
altfel
true
false

returneaza