Alphabet
uppercase characters of the English alphabet (A-Z)
decimal digits (0-9)
underline character '_'

Lexic
operators: +, -, *, /, %, <-, si, sau, ss, se, ee, nee, be, bb, nu
separators: {}, ', (), [], space, newline, st, "", ``, $, ,(actual comma)
reserved words: num bul lit string pt cat scr cit dc alfel, return
identifiers: a sequence of letters, digits and "_" and such that the first character is a letter
identifier = letter | letter{letter|digit|"_"}

letter = "A" | "B" | . ..| "Z"
digit = "0" | "1" |...| "9"
nonzerodigit = "1" |...| "9"

num = [-]nonzerodigit{digit}|0
string = "" | "sir" (the string is of the form "Ana")
sir = (letter | digit | _ | " "){sir}
lit = "digit" | "letter"
bul = "true" | "false"

num_array_constant = "[" [{num","}] "]"
bul_array_constant = "[" [{bul","}] "]"
lit_array_constant = "[" [{lit","}] "]"
string_array_constant = "[" [{string","}] "]"

Tokens
+
-
*
/
%
ee
nee
ss
se
bb
be
<-
si
sau
[
]
{
}
(
)
,
'
"
$
st

```
space
newline
num
string
lit
bul
cit
scr
dc
alfel
pt
return


Syntax
program = decllist "'" stmtlist "st"
decllist = declaration | declaration "'" decllist
declaration = type identifier
type1 = "bul" | "num" | "lit" | "string"
arraydecl = type1 "^" nr "^"
type = type1 | arraydecl
vardecl = type1 identifier
stmtlist = stmt | stmt stmtlist
stmt = simplstmt | structstmt
simplstmt = assignstmt | iostmt | comment | returnstmt | ostmt
comment = "$" sir "$"
assignstmt = identifier "<-" (expression | [not] compcondition |
stringexpression) "'"
expression = expression operator term | term
term = term ("*" | "/" | "%") factor | factor
factor = "(" expression ")" | identifier | num
stringexpression = stringexpression "+" string | string
iostmt = "cit" | "scr" "(" identifier ")" "'"
ostmt = "scr" "(" stringexpression ")" "'"
returnstmt = "return" expression "'"
structstmt = ifstmt | whilestmt | forstmt
ifstmt = "dc" compcondition ":" stmtlist ":" ["alfel" ":" stmt ":"]
whilestmt = "cat" compcondition ":" stmtlist ":"
forstmt = "pt" "(" vardecl "'" assignstmt "'" assignstmt ")" ":" stmtlist
":"
compcondition = ["("] condition ("si"|"sau") compcondition [")"] |
condition
condition = [not] simplcondition
simplcondition = expression relation expression
relation = "ss" | "se" | "ee" | "nee" | "be" | "bb"
not = "nu"

Review
for https://github.com/ciprianturcu/University-
Projects/tree/main/Semester5/Formal%20Languages%20and%20Compiler%20Design
/lab2
| <stmtlist> ::= <stmt>; | <stmt>;<stmtlist>
| <stmt> ::= <assignstmt> | <declrstmt> | <iostmt> | <ifstmt> | <forstmt>
| <whilestmt> | <returnstmt>
```

in the lab1 files, "if","for","while" statements do not have ";" after them

```
| while (n>0) {
|     in(x);
|     l.add(x);
| }
```