Alphabet:

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
  + - * / = < <= == >= and or

- separators
  () [] {} , ;

- reserved words:
  let bool int string read write if then for while do true false

b.identifiers

-a sequence of letters and digits, such that the first character is a letter; the rule is:

identifier = (letter | "_") {character}
character = letter | digit | "_"
letter = "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"
digit = "0" | nonzero
nonzero = "1" | "2" | ... | "9"

c.constants

1.integer:

integer = "0" | ["+" | "-"] nonzero {digit}

2.boolean:

boolean = "true" | "false"

3.string:

string = """ {character} """

```
#
+
-
*
/
%
=
<
<=
==
>=
and
or
(
)
[
]
{
}
,
;
let
bool
int
string
read
write
if
then
for
while
do
true
false
```

```
program = "#" block "#"
block = statement ";"

statement = compoundStatement |  declarationStatement | assignmentStatement | ioStatement | ifStatement | loopStatement
compoundStatement = statement ";" statement

declarationStatement = "let" declaration
declaration = type declaree {"," declaree} | declaration "," declaration
declaree = identifier | declaree "[" integer "]" | declaree "=" expression
type = ("int" | "bool" | "string")

assignmentStatement = address "=" expression
address = identifier | address "[" expression "]"

expression = listExpression | expression ("+" | "-" | "or") term | term
listExpression = "{" "}" | "{" expression {"," expression} "}"
term = term ("*" | "/" | "%" | "and") factor | factor
factor = "(" expression ")"  | relational | address | constant
relational = expression comparator expression
comparator = "<" | "<=" | "==" | ">=" | ">"
constant = integer | boolean | string

ioStatement = readStatement | writeStatement
readStatement = "read" "(" address ")"
writeStatement = "write" "(" expression ")"

ifStatement = "if" "(" expression ")" "then" "{" block "}" ["else" "{" block "}"]

loopStatement = whileStatement | forStatement
forStatement = "for" "(" (declarationStatement | assignmentStatement) ";" expression ";" assignmentStatement ")" "do" "{" block "}"
whileStatement = "while" "(" expression ")" "do" "{" block "}"

integer = "0" | ["+" | "-"] nonzero {digit}
boolean = "true" | "false"
string = """ {character} """

identifier = (letter | "_") {character}
character = letter | digit | "_"
letter = "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"
digit = "0" | nonzero
nonzero = "1" | "2" | ... | "9"
```