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: <- + +<- - -<- < <= = > >= / % $
  - separators: | ( ) [ ] # ... ; ,
  - reserved words: let int char read check write for ret else const while

b. Identifiers
  - a seq. of letters that can end with digits such that the first character is always a letter

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

c. Constants

1. integer

```
        intconst = ["-"|"+"]no | "0"
        no = digit1{digit}
        digit1 = "1"|"2"|...|"9"
```

2. character

```
        character = 'letter'|'digit'
```

3. string

```
        strconst = "string"
        string = char{string}
        char = letter|digit|"_"
```

```
program = "f" "int" "main" "(" ")" "#" statement "ret" intconst "#"
statement = decl | write | read | loop | check | assign | compoundstmt
compoundstmt = statement ";" statement
decl = "let" type (declaree) {","(declaree)}
declaree = identifier["[" intconst "]"]  | identifier[<- (intconst | strconst)]
type = "int" | "char"
write = "write" "(" printable ")"
printable = strconst | intconst | identifier | expression | printable printable
read = "read" "(" identifier["[" expression "]"] ")"
loop = ("for" "(" ( assign | decl ) "|" expression "|" assign ")" "#" statement "#")
| ("while" "(" expression ")" "#" statement "#")
check = "check" "(" expression ")" "#" statement "#" {"else" "check" "(" expression
")" "#" statement "#"}
assign = identifier["[" expression "]"] ("<-" | "-<-" | "+<-") expression
expression = expression ("+"|"-"|"or") term | term
term = term ("*"|"/"|"%","and") factor | factor
factor = "(" expression ")" | relational | identifier["[" expression "]"] | intconst
| strconst
relational = expression comp expression
comp = "<" | "<=" | "=" | ">=" | ">"
```

```
let
int
char
read
check
write
for
ret
else
const
while
and
or
(
)
[
]
#
...
,
<-
+<-
-<-
=
/
%
+
-
*
|
```