lexic.txt

Alphabet:

- upper (A-Z) and lower case letters (a-z) of the English alphabet

- underline character '_'

- decimal digits (0-9)

Lexic:

- special symbols:

    - operators: + - * / == < <= > >= = !=

    - separators: () {} , ; : space newline "

    - reserved words: var int str read print if else do while

- identifiers:

    - identifier := (letter|"_"){letter|digit|"_"}

    - letter := "A"|"B"|...|"Z"|"a"|"b"|...|"z"

    - digit := "0"|"1"|...|"9"

- constants:

    - intconst := ["+"|"-"]non_zero_digit{digit}|"0"

      non_zero_digit := "1"|"2"|...|"9"

    - strconst := """{letter|digit|"_"|" "}"""

token.in

```
+
-
*
/
!=
==
=
<=
>=
<
>
{
}
(
)
,
;
:
space
newline
"
_
0 - 9
A - Z
a – z
var
int
str
read
print
if
else
while
```

syntax.in

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

decllist ::= declaration | declaration ";" decllist
declaration ::= IDENTIFIER ":" type
type ::= type1 | arraydecl
type1 ::= "int" | "str"
arraydecl ::= "arr" "(" type1 "[" INTCONST "]" ")"

cmpdstmt ::= "{" stmtlist "}"
stmtlist ::= stmt | stmt ";" stmtlist
stmt ::= simplstmt | structstmt

simplstmt ::= assignstmt | iostmt
assignstmt ::= IDENTIFIER "=" expression
expression ::= expression "+" term | expression "-" term | term
term ::= term "*" factor | term "/" factor | factor
factor ::= "(" expression ")" | IDENTIFIER | INTCONST
iostmt ::= "read" "(" IDENTIFIER ")" | "print" "(" IDENTIFIER ")" | "print" "(" STRCONST ")" | "print" "("
INTCONST ")"

structstmt ::= cmpdstmt | ifstmt | whilestmt
ifstmt ::= "if" "(" condition ")" "{" stmt "}" ["else" "{" stmt "}"]
whilestmt ::= "while" "(" condition ")" "{" stmt "}"
condition ::= expression RELATION expression
RELATION ::= "<" | "<=" | "==" | "!=" | ">=" | ">"
```