lexic.txt

```
Alphabet:
 - Upper (A-Z) and lower case letters (a-z) of the English alphabet
 - Decimal digits
Lexic:
    - special symbols:
                - operators: + - * / < <= = >= == <>
          - separators: [ ] { } , ; : space newline " ’ |
          - reserved words: program var if else print for while arr input

    - identifiers: a sequence of letters and digits, such that the first character is
 a letter with the rule being:
          identifier ::= ("" | letter) { letter | digit | "" }
          letter := "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"
          digit := "0" | "1" | "2" | ... | "9"

    - constants:
                intconst ::= "0" | ["+" | "-"] nz_digit { digit }
                nz_digit ::= "1" | "2" | ... | "9"
                strconst ::= ’|’ { letter | digit | "_" | " " } ’|’
                charconst ::= "’" (letter | digit | special_char) "’"
                special_char ::= "+" | "-" | "*" | "<" | ">" | ...
```

syntax.in

```
program ::= "program" IDENTIFIER "[" stmtlist "]"

stmtlist ::= stmt | stmt stmtlist | stmt ";" stmt ";"...

stmt ::= declaration | assignment | inputstmt | forstmt | ifstmt | printstmt

declaration ::= "var" IDENTIFIER "=" NUM_CONSTANT

assignment ::= "var" IDENTIFIER "=" expression

expression ::= IDENTIFIER | NUM_CONST | "(" expression ")" | expression "+" expression

inputstmt ::= "var" IDENTIFIER "=" "input" "{" IDENTIFIER "}"

forstmt ::= "for" "{" stmtlist "}"

ifstmt ::= "if" "{" stmtlist "}" ["else" "{" stmtlist "}"]

printstmt ::= "print" "{" STR_CONSTANT "}"
```

token.in

```
A...Z
a...z
+
-
*
```

```
/
<
>
=
==
>=
<=
<>
[
]
{
}
,
;
:
|
space
newline
’
"
_
0...9
program
var
if
else
print
for
while
arr
input
```