>>>>>>>>> Lexic.txt <<<<<<<<<<
Alphabet:

1. Upper (A-Z) and lower case letters (a-z)
2. Decimals: 0-9

Lexic:

- assignment operator : <-
- arithmetic operators: +,-,*,/, mod
- relational operators: =, <, >, <=, >=, <>
- logical operators: and, or, not
- separators: , ; ( ) [ ]
- reserved words: in, out, begin, number, string, character, if, end, for, end_if, end_for, while, end_while, else
- identifiers: a sequence of letters and digits, such that the first character is a letter

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

- constants:
 1. integer:
  nzdigit = "1"|"2"|...|"9".
  digit = "0"|nzdigit.
  nzintconst = ["+"|"-"]nzDigit{digit}
  intconst = "0"|nzintconst.
 2. character:
  character = """char"""
  char = letter|digit
 3. string:
  stringconst = """{char}"""

>>>>>>>>>> Token.in <<<<<<<<<<<<
begin
end
:
<-
+
-
*
/
mod
in
out
and
or
not
>
<
=
<>
>=
<=
,

;
(
)
[
]
number
string
character
if
else
end_if
for
end_for
while
end_while

>>>>>>>>>>>>> syntax.in <<<<<<<<<<<<<<<<

Predefined tokens are specified between ` and ` .
ex: `and`

List of sintactical rules:

<simpletype> ::= `number`|`character`|`string`.
<arraydeclaration> ::= <simpletype> `[` <nzintconst> `]`.
<type> = <simpletype>|<arraydeclaration>.

<declaration> ::= <type> <identifier> `;`.
<assignmentstmt> ::= <identifier> `<-` <expression> `;`.

<expression> ::= <expression> `+` <term> | <expression> `-` <term> | <term>.
<term> ::= <term> `*` <factor> | <term> `/` <factor> | <factor>.
<factor> ::= `(` <expression> `)` | <identifier> | <constant>.
<constant> ::= <intconst>|<stringconst>.

<iostmt> ::= `in` <identifier> `;` | `out` <identifier> `;` | `out` <constant> `;`.
<condition> ::= `(`<expression><relation><expression>`)` | `not` `(` <expression> `)`.
<relation> ::= `=`|`<>`|`>=`|`<=`|`and`|`or`.

<ifstmt> ::= `if` <condition> `:` <statementlist> [`else` <statementlist>] `end_if`.
<whilestmt> ::= `while` <condition> `:` <statementlist> `end_while`.
<forstmt> ::= `for` <assignmentstmt> `,` <identifier> `,` <constant> `:` <statementlist> `end_for`.

<simplestatement> ::= <declaration>|<iostmt>|<assignmentstmt>.
<statement> ::= <simplestatement>|<ifstmt>|<whilestmt>|<forstmt>.
<statementlist> ::= <statement>|<statement><statementlist>.

<program> ::= `begin` `:` <statementlist> `end`.