Lexic.txt:
Alphabet:
a. Lower case letters (a-z) of the English alphabet
b. Decimal digits (0-9)
c. Underscore ('_')

1. Lexic:
a. Special symbols:
  - Operators: + - * / = == && || != ++ -- < <= > >=
  - Separators: space : () [] {} \n
  - Reserved words: integer, char, string, return, next, if, elsif, else, end, FOR, WHILE,
    puts, break
b. Identifiers:
- A sequence of letters and digits, such tat the first character is a letter; the rule is:
 - identifier = "_" letter {letter | digit} "_"
 - letter = "a" | "b" | ... | "z"
 - digit = "0" | "1" | ... | "9"
c. Constants:
1. Integer - rule:
 const_nr = 0 | ["+"|"-"] nz_digit {digit}
 digit = "0" | "1" | ... | "9"
 nz_digit = "1" | "2" | ... | "9"
2. Character - rule:
 char = 'letter' | 'digit' | 'special_char'
 special_char = ! | ? | " | : | ; | # | $ | % | ^ | & | ...
3. String - rule:
 string = "{char}"

Token.in:
+
-
*
/
=
==
&&
||
!=
++
==
<
<=
>
>=
\n
space
:
()
[]
{}
integer
char
string
return
next

if
elsif
else
end
FOR
WHILE
puts
break

Syntax.in
2. Syntax:
The words - predefined tokens are specified between " and ":
a) Syntactical rules:
- program = "statements_seq"
- statements_seq = statement {statement}
- statement = declaration | simple_statement | struct_statement
- declaration = type identifier
- type = ("integer" | "char" | "string") | array_type
- array_type = ("integer" | "char" | "string") "[" number "]"
- simple_statement = assignment_statement | io_statement
- struct_statement = if_statement | for_statement | while_statement
- compound_statement = "START" statement_seq "END"
- assignment_statement = identifier "=" expression
- expression = term | expression op expression
- term = identifier | number | string
- op = "+" | "-" | "*" | "/"
- io_statement = "puts" "(" identifier ")" | "puts" "(" const ")"
- if_statement = "if" condition statement {elsif condition statement} [else statement]
- for_statement = "for" assignment_statement; condition; statement
- while_statement = "while" condition compound_statement
- condition = expression relation expression
- relation = "<" | "<=" | "==" | "!=" | ">=" | ">"