---------- Lexic ----------

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: [ ] { } ; : space
- reserved words: start finish var int char string print read if then else while execute exit

b. Identifiers = a sequence of letters and digits s.t. the first character is a letter

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

c. Constants

-integer:
int_no = [("+" | "-")] non_zero_digit {digits} | "0"
non_zero_digit = "1" | ... | "9"

- character:
character = 'letter'|'digit'|'symbol'
symbol = ":" | ";" | "?" | "!" | "."

- string
string = "character_for_string{character_for_string}"
character_for_string = letter | digit | symbol

---------- Syntax.in ----------

Sintactical rules:
	program = "start" compound_statement "finish"
	compound_statement = (declaration_statement | array_declaration_statement | statement) ";" [compound_statement]

declaration_statement = "var" identifier ":" primitive_type ["=" expression]
primitive_type = "int" | "char" | "string"
constant_value = int_no | character | string
array_declaration_statement = "var" identifier ":" primitive_type "[" int_no "]"
statement = assignment_statement | io_statement | if_statement | while_statement | "exit"
io_statement = read_statement | write_statement
assignment_statement = identifier "=" expression
read_statement = read "(" identifier ")"
write_statement = print "(" identifier | constant_value ")"
expression = term [("+" | "-") expression]
term = factor [("*" | "/") term]
factor = int_no | identifier | "(" expression ")"
if_statement = "if" "(" condition ")" "then" "{" compound_statement "}" ["else" "{" compound_statement "}"]
while_statement = "while" "(" condition ")" "execute" "{" compound_statement "}"
condition = expression relational_operator expression
relational_operator = "<" | ">" | "<=" | ">=" | "==" | "!="

---------- Token.in ----------

=
+
-
*
/
<
>
<=
>=
==
!=
[
]
{
}
;
:
start
finish
var
int
char
string

print
read
if
then
else
while
execute
break
exit

---------- Problems ----------

Lab 1

P1. Compute max of 3 numbers.

```
start

	var a: int;
	var b: int;
	var c: int;

	read(a);
	read(b);
	read(c);

	var max: int = a;

	if (max < b) then {
		max = b;
	}

	if (max < c) then {
		max = c;
	}

	print("The max number is: ");
	print(max);
finish
```

P1.Error. Compute max of 3 numbers

```
start
```

```
	var a: int;
	var b: int;
	var c: int;

	read(a);
	read(b);
	read(c);

	// identifier starts with a digit
	var max: int = 1a;

	if (max < b) then{
		max = b;
	}

	if (max < c) then{
		max = c;
	}

	print("min");
	print(min);
finish
```

P2. Check if a number is prime

```
start
	var n: int;
	var i: int = 3;

	read(n);

	if (n == 2) then{
		print("number is prime");
	}

	while(i * i < n) execute {
		if (n % i == 0) {
			print("number is not prime");
			exit;
		}
		i = i + 2;
	}

	print("number is prime");
```

finish

P3. Calculate the sum of all numbers of an array.

```
start
	var array: int[10];
	var n: int;

	read(n);

	var sum: int = 0;
	var i: int = 0;

	while(i < n) execute{
		read(array[i]);
		sum = sum + array[i];
		i = i + 1;
	}

	print("the sum of numbers from the array is: ");
	print(sum);
finish
```