Laboratory 2
2020

1. Language Specification
1.1 Alphabet:
	1.1.a. Upper (A-Z) and lower case letters (a-z)
	 1.1.b Decimal digits (0-9)
Lexic:
	a.Special symbols, representing:
		-operators: + - * / = < > <= == >= != && ||
		-separators: ; {} [] () , space ‘\n’
		-reserved words: int, char, if, else, while, read, print, list, return

	b.identifiers
		-sequence of letters and digits such that the first character is a letter
		-Rule:
			identifier = letter [{(letter | digit)}]
			letter = “A” | “B” | … | “Z” | “a” | “b” | … | “z”
			digit = “0” | “1” | … | “9”

	c.constants
		1.integer
			integerConstant = [“+” | “-”] nonzero_digit {digit} | “0”
			nonzero_digit = “1” | “2” | … | “9”
			digit = “0” | nonzero_digit
		2.character
			character = ‘letter‘ | ‘digit’
			-these are defined at b
		3.list of integers
			list_of_integer = “[“ elements “]”
			elements = element {“, ” element}
			element = integer

2. Syntax:
	a. Sintactical rules:
		program = “int main () { “ statementList ” } “
		statementList = statement ; { statement ; }
		statement = simpleDeclaration | simpleAssignmentStatement | listAssignmentStatement | ioStatement | whileStatement | ifStatement

		type = “int” | “char”
		simpleDeclaration = type identifier “=” expression
		listDeclaration = “list” identifier “=” list_of_integer
		simpleAssignmentStatement = identifier “=” expression
		listIdentifier = identifier "[" identifier | integerConstant "]"
		constant = integerConstant | character

		ioStatement = “print “ (identifier | constant | listIdentifier) | “read ” identifier
		ifStatement = “if (” condition “) {” statementList “}” { “else {” statementList “}” }
		whileStatement = “while (“ expression “) {“ statementList “}”

		expression = constantExpression | booleanExpression | arithmeticExpression | identifier | listIdentifier
		constantExpression = integerConstant | character
		arithmeticExpression = expression operator expression
		operator = “+” | “-” | “*” | “/”

booleanExpression = expression booleanOperator expression
booleanOperator = “==” | “<=” | “>=” | “<” | “>” | “!=” | “&&” | “||”

b. Lexical rules:
identifier = letter [{(letter | digit)}]
letter = “A” | “B” | … | “Z” | “a” | “b” | … | “z”
digit = “0” | “1” | … | “9”

Codification:
- reserved words: each word has its own code
- operators: each operator has its own code
- separators: each separator has its own code

Token type → Code
identifier → 0
constant → 1
+ → 2
- → 3
* → 4
/ → 5
= → 6
< → 7
> → 8
<= → 9
== → 10
>= → 11
!= → 12
&& → 13
|| → 14
; → 15
{ → 16
} → 17
(→ 18
) → 19
‘ ‘ → 20 **space not used**
'\n' → 21
[→ 22
] → 23
, → 24
int → 25
char → 26
if → 27
else → 28
while → 29
read → 30
print → 31
list → 32
int_main → 33