Specifications

Alphabet:

a. [A-Za-z]

b. [0-9]

c. Underscore ('_')

Lexic:

a. Special symbols, representing:

Operators:

- + - * ** / % (Arithmetic operators: *addition, subtraction, multiplication, power, division, mod*)
- < <= > >= Equal NotEqual(Relational operators: *smaller, smaller or equal, greater, greater or equal, equality, inequality*)
- && || ! (Logical operators: *and, or, not*)
- = (Assignment operator)
- [ ] (Index operator)

Separators:

- { } ( ) , ; <space> <newline> <indent>

Reserved words:

- read, write, if, ifNot, for, while, do, break, number, string, char, list, return, program

b. Identifiers

- IDENTIFIER = letter {letter | digit | underscore}
- letter = "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
- digit = "0" | non_zero_digit
- non_zero_digit = "1" | ... | "9"
- underscore = "_"

c. Constants

- integer = "0" | ["+" | "-"] non_zero_digit{digit}
- character = ‘letter’ | ‘digit’ | ‘underscore’
- string = "{character}"
- CONSTANT = integer | character | string

Tokens:

(	&&	char
)	||	list
[	!	return
]	=	do
{	,	
}	;	
+	<space>	
-	<newline>	
*	<indent>	
**	read	
/	write	
%	if	
<	ifNot	
<=	for	
>	while	
>=	break	
Equal	number	
NotEqual	string	

Syntax:

- program = “program ” + [name] + “{” compound_statement + “}”

- compound_statement = “{“ statement_list “}”
- statement_list = statement | statement “;” statement_list
- statement = simple_statement | struct_statement
- simple_statement = assign_statement | io_statement | declaration
- struct_statement = compound_statement | if_statement | while_statement | for_statement
- assign_statement = (IDENTIFIER | indexed_identifier) “=“ expression “;”
- io_statement = read_statement | write_statement
- read_statement = “read” “(“ (IDENTIFIER | indexed_identifier) {“,” (IDENTIFIER| indexed_identifier)} “)” “;”
- write_statement = “write” “(“ id {“,” id} “)” “;”
- if_statement = “if” “(“ condition_statement “) => ” compound_statement
  “ifNot =>” “{“ compound_statement “}”
- for_statement = “for” “(“ “int” assign_statement “;” condition “;” assign_statement “)” compound_statement
- while_statement = “while” “(“ condition_statement “)” “do” compound_statement
- condition_statement = cond | cond LOGICAL cond

- expression = [expression("+"|"-")] term
- term = term("*" | "/") factor | factor
- factor = "(" expression ")" | id

- id = IDENTIFIER | CONSTANT | indexed_identifier
- declaration = type “ “ IDENTIFIER {“,” IDENTIFIER} “;”
- type = simple_type | array_declaration
- simple_type = “int” | “string” | “char”
- array_declaration = “list” “<” simple_type “>”
- condition = [“!”] expression RELATION expression

- indexed_identifier = IDENTIFIER "[" integer "]"
- RELATION ::= "<" | "<=" | "Equal" | "NotEqual" | ">=" | ">"
- LOGICAL :== "&&" | "||"