Assigment 2

**Lexic**

Alphabet:

* Uppercase letters: A - Z
* Lowercase letters: a - z
* Decimal digits: 0 - 9

Lexical:

* Special symbols, representing:
  + Operators
* Comparison: x == x <= x >= x < x >
* Arithmetic: x = x + x / x \* x % x // x
* Separators
* ()
* []
* {}
* ;
* ,
* space
* /t
* ”
* /n
* Reserved words
* read
* write
* if
* else
* while
* for
* repeat
* until
* int
* char
* var
* const
* Identifiers:
* Identifier = letter | letter {letter}{digit}
* Letter = ”A ”| ”B ”|...| ”Z ”| ”a ”| ”b ”|...| ”z ”
* Digit = ”0 ”| ”1 ”|...| ”9 ”
* NonZeroDigit = ”1 ”|...| ”9 ”

* Constants:
* NumericConstant = ”0 ”|[ ”+ ”] NonZeroDigit|[ ”- ”] NonZeroDigit
* StringConstant = ”{char} ”
* CharConstant = ”[char] ”
* Char = letter|digit|” ”

**Syntax**

* program = {statement}
* declarationList = declaration | declaration ",” declarationList
* declaration = ”var” identifier ":" type | "const" identifier ":" type

Types

* simpleType = "int" | "real" | "char" | "bool"
* arrayDeclaration = "[" dataType "]"
* type = simpleType | arrayDeclaration

Statements

* compoundStatement = "{" statementList "}"
* statementList = statement | statement "\n" statementList
* statement = simpleStatement | structStatement
* simpleStatement = assignStatement | ioStatement | declarationList
* assignStatement = identifier "=" expression ";”
* ioStatement = "read" "(" identifier ")" ";” | "write" "(" identifier ")" ";”| "write" „(” constant„)” ";” |
* structStatement = compoundStatement | ifStatement | whileStatement
* ifStatement = "if" "(” compoundCondition ")” "{” compoundStatement"}” ["else" "{” compoundStatement"{”]
* whileStatement = "while" „(” compoundCondition „)” "{” compoundStatement"}”

Expressions

* expression = expression "+" term | expression "-" term | term
* term = term "\*" factor | term "/" factor | factor
* factor = identifier | numericConstant

Conditions

* condition = expression relation expression
* compoundCondition = condition | "(" conditon ")" && "(" conditonList ")" | "(" conditon ")" || "(" conditonList ")"

Relations

* relation = "<" | "<=" | "==" | ">=" | ">" | "!="

**Tokens**

* =
* +
* -
* /
* \*
* ()
* []
* {}
* ;
* ,
* "
* %
* //
* ==
* !=
* <=
* >=
* &&
* ||
* space
* /n
* read
* write
* fread
* fwrite
* if
* else
* while