LEXIC

Alphabet:
- Uppercase (A-Z) and lowercase (a-z) English letters
- Underscore character '_'
- Decimal digits (0-9)

Lexical Elements:
1. Identifiers: (An identifier is a sequence of letters and digits, ALWAYS starting with a letter)
 - identifier = letter { letter | digit }
 Identifier: [A-Za-z_][A-Za-z_0-9]*

2. Constants:
 - Constants can be integers, characters, or strings.

 a. Integer Constants: (An integer constant can start with an optional '+' or '-' sign, but it should not start with 0.)
 - intconst ="0" | ["+" | "-"] ("1" | ... | "9") {digit}
 Integer Constant: 0
 Integer Constant: [+-]?[1-9][0-9]*

 b. Character Constants:
 - character = { letter | digit }
 Character Constant: '[A-Za-z0-9]+'

 c. String Constants: (A string constant contain multiple characters.)
 - constchar = { char }
 - char = letter | digit | special_character
 - special_character = any character that is not a letter or digit
  String Constant: "[A-Za-z0-9]+"

3. Operators and Separators:
 - Operators: +, -, *, /, %, :=, <, <=, =, >=
 - Separators: [,], {, }, :, ;, space

4. Reserved Words:
 - array char const do else if int of program read then var while write,endif,endwhile

Notes:
- letter = "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"
- digit = "0" | "1" | ... | "9"

Syntax

program = "VAR" decllist ";" cmpdstmt "." // A program starts with "VAR,"lists declarations, contains statements, and ends with a period.

decllist = declaration | declaration ";" decllist // Declarations can be oneor more, separated by semicolons.

declaration = IDENTIFIER ":" type // A declaration names a variable using anidentifier and specifies its type.

type = type1 | arraydecl // A variable can have a basic type like BOOLEAN,CHAR, INTEGER, or REAL, or it can be an array.

type1 = "BOOLEAN" | "CHAR" | "INTEGER" | "REAL" // Basic data types forvariables.

arraydecl = "ARRAY" "[" nr "]" "OF" type1 // Describes an array byspecifying its size and element type.

cmpdstmt = "BEGIN" stmtlist "END" // A block of statements starts with"BEGIN," contains multiple statements, and ends with "END"

stmtlist = stmt | stmt ";" stmtlist // Lists statements within a block,separated by semicolons.

stmt = simplstmt | structstmt // Statements can be simple (like assignmentor I/O) or structured (like IF or WHILE).

simplstmt = assignstmt | iostmt // Simple statements include assignments andinput/output operations.

assignstmt = IDENTIFIER ":=" expression // Assignment sets the value of avariable to the result of an expression.

expression = term | term operator term

operator = "+","-","*","/","%"

term = "IDENTIFIER" | INTCONST

iostmt = readstmt | writestmt // Input and output statements for reading/writing values from/to variables.

readstmt = "READ" "(IDENTIFIER)"

writestmt = "WRITE" "(IDENTIFIER)" | "CONSTCHAR"

structstmt = cmpdstmt | ifstmt | whilestmt // Structured statements includecompound statements, IF conditions, and WHILE loops.

ifstmt = "IF" condition "THEN" stmtlist ["ELSE" stmtlist] "ENDIF" // Conditional statementfor making decisions with an optional else part.

whilestmt = "WHILE" condition "DO" stmtlist "ENDWHILE" // A loop statement that repeatswhile a condition is true.

conditionlist = condition | condition "AND" | "OR" conditionlist

condition = expression RELATION expression // Conditions compare twoexpressions using relational operators.

```
RELATION = "<" | "<=" | "=" | "<>" | ">=" | ">" // Relational operators
usedfor comparing expressions.
```