Lab 1b

Token.in

|  |
| --- |
| declare |
| int |
| string |
| boolean |
| char |
| { |
| } |
| [ |
| ] |
| ( |
| ) |
| , |
| ; |
| > |
| <= |
| >= |
| < |
| > |
| = |
| == |
| + |
| - |
| \* |
| / |
| div |
| mod |
| identifier |
| print |
| read |
| if |
| elif |
| True |
| False |
| for |
| increase |
| by |
| while |
| “” |
| ‘’ |

**Lexic.txt**

a.Special symbols, representing:

**-** operators: + - \* / = < > <= >=

**-** separators [ ] { }  : ; “” ‘’ ( ) ,

- reserved words: declare int string boolean char div mod print read if elif else True False for while increase by

    b.identifiers

  -a sequence of letters and  digits, such that the first character is a letter; the rule is:

    identifier ::= letter | letter{letter}{digit}

    letter ::= "A" | "B" | . ..| "Z"

    digit ::= "0" | "1" |...| "9"

c. constants

1. int : noconst = [‘+’| ‘-‘] no |no

      no = digit{no}

2.character

    character = 'letter' | 'digit'

3.string

      constchar= "string"

      string= char{string}

      char= letter|digit

4. boolean

bool = “True”|”False”

**Syntax.in**

The words - predefined tokens are specified between " and ":

Sintactical rules:

program ::= "DECLARE" LISTdecl ";" CMPDstmt "."

LISTdecl::= declaration | declaration ";" LISTdecl

declaration ::= IDENTIFIER ":" type

type1 ::= "BOOLEAN" | "CHAR" | "INT" | “STRING”

ARRAYdecl ::= identifier "[" integer "]" "OF" type1

type  ::= type1|ARRAYdecl

CMPDstmt ::= "{" LISTstmt "}"

LISTstmt ::= stmt | stmt ";" LISTstmt

stmt ::= SIMPLEstmt | STRUCTstmt

SIMPLEstmt ::= ASSIGNstmt | IOstmt

ASSIGNstmt ::= identifier "=" expression

expression ::= expression ["+"|”-”] term | term

term ::= term ["\*"|”/”] factor | factor

factor ::= "(" expression ")" | identifier

IOstmt ::= ["READ" | "PRINT"] "(" identifier ")"

STRUCTstmt ::= CMPDstmt | IFstmt | WHILEstmt | FORstmt

IFstmtt ::= "IF(" condition "){" stmt “}” ["ELSE{" stmt “}”| “ELIF(" condition "){" stmt “}”]

WHILEstmt ::= "WHILE(" condition "){" stmt “}”

FORstmt ::= “FOR(” “increase by” integer “;” identifier “=” integer “,” identifier “){” stmt “}”

condition ::= expression RELATION expression

RELATION ::= "<" | "<=" | "=" | "<>" | ">=" | ">"