**UNEXPECTED-EOF = -2,**

**INVALID-LEX = -1,**

**END-OF-FILE = 0,**

**VAR = 1, ----------------- (none)**

**ASSIGN = 2, -------------- =**

**INPUT = 3, -------------- input**

**PAR\_OPEN= 4,-------------- (**

**PAR\_CLOSE = 5, ----------- )**

**STRING = 6(Entre aspas),-- (none)**

**POT\_COMMA = 7, ----------- ;**

**BRA\_OPEN = 8, ------------ [**

**BRA\_CLOSE = 9, ----------- ]**

**DOT = 10, ---------------- .**

**RAND = 11, --------------- rand**

**COMMA = 12, -------------- ,**

**SHOW = 13, --------------- show**

**NUMBER = 14, ------------- (none)**

**FOR = 15, ---------------- for**

**SEQ = 16, ---------------- seq**

**MINUS = 17, -------------- -**

**PLUS = 18, --------------- +**

**TIMES = 19, -------------- \***

**DIV = 20, ---------------- /**

**MOD = 21, ---------------- %**

**VALUE = 22, -------------- value**

**END = 23, ---------------- end**

**ISEQ = 24, --------------- iseq**

**EQUAL = 25, -------------- ==**

**DIFF = 26, --------------- !=**

**LOWER = 27, -------------- <**

**LOWER-EQUAL = 28, -------- <=**

**GREATER = 29, ------------ >**

**GREATER-EQUAL = 30, ------ >=**

**AND = 31, ---------------- &**

**OR = 32, ----------------- |**

**IF = 33, ----------------- if**

**ELSE = 34, --------------- else**

**WHILE = 35, -------------- while**

**OPPOSED = 36, ------------ opposed**

**TRANSPOSED = 37, --------- transposed**

**SUM = 38, ---------------- sum**

**MUL = 39, ---------------- mul**

**NULL = 40, --------------- null**

**FILL = 41, --------------- fill**

**ID = 42, ----------------- id**

**SIZE = 43,---------------- size**

**ROWS 44, ----------------- rows**

**COLS = 45. --------------- cols**

**class SymbolTable{**

**private Map<String,Symbol> st = new HashMap<>();**

**public Symbol getSymbol(String lex){**

**if( st.contain(lex)) return st.get(lex);**

**else return Symbols.INVALID-LEX;**

**}**

**public void addSymbol(String lex, Symbol s){**

**st.put (lex,s);**

**}**

**public bool haveLex(String lex){**

**if(st.contain(lex)) return true**

**return false**

**}**

**public SymbolTable(){**

**addSymbol("=",Symbol.ASSIGN);**

**addSymbol("input",Symbol.INPUT);**

**addSymbol("(",Symbol.PAR\_OPEN);**

**addSymbol(")",Symbol.PAR\_CLOSE);**

**.**

**.**

**.**

**addSymbol("opposed",Symbol.OPPOSED);**

**addSymbol("transposed",Symbol.TRANSPOSED);**

**addSymbol("sum",Symbol.SUM);**

**addSymbol("mul",Symbol.MUL);**

**addSymbol("null",Symbol.NULL);**

**addSymbol("fill",Symbol.FILL);**

**addSymbol("id",Symbol.ID);**

**addSymbol("size",Symbol.SIZE);**

**addSymbol("rows",Symbol.ROWS);**

**addSymbol("cols",Symbol.COLS);**

**}**

**}**

**Lexima = (Token, Symbol);**

**("dim", Symbol.VAR)**

**("=", Symbol.ASSIGN)**

**("input", Symbol.INPUT)**

**("(", Symbol.PAR\_OPEN)**

**("Entre com ... ", Symbol.STRING)**

**( ")", Symbol.PAR\_CLOSE)**

**(";", Symbol.DOT\_COMMA)**

**("matriz", Symbol.VAR)**

**("=", Symbol.ASSIGN)**

**("[", Symbol.BRA\_OPEN)**

**("]", Symbol.BRA\_CLOSE)**

**(".", Symbol.DOT)**

**class Lexema{**

**public String token;**

**public Symbol symbol;**

**}**

**class LexicalAnalysis{**

**private InputStream input;**

**private SymbolTable st;**

**public LexicalAnalysis(String fileName){**

**input = ...(fileName);**

**st = new SymbolTable();**

**}**

**public Lexema nextToken(){**

**}**

**}**