https://github.com/911-Abrudan-Rebeca/FLCD/tree/main/Lab8

Laboratory 8

Lex file lang.lxi:

```
%{
   #include <stdio.h>
   #include <stdlib.h>
   #include <string.h>

   int lines = 1;
%}

%option noyywrap
%option caseless

DIGIT [0-9]
NON_ZERO_DIGIT [1-9]
INT_CONSTANT [+-]?{NON_ZERO_DIGIT}{DIGIT}*|0
LETTER [a-zA-Z_]
SPECIAL_CHAR [ ?:*\^+=.!]
STRING_CONSTANT (\"({LETTER}|{DIGIT}|{SPECIAL_CHAR})*\")
IDENTIFIER {LETTER}({LETTER}|{DIGIT})*
BAD_IDENTIFIER ({DIGIT})+({LETTER})+({LETTER}|{DIGIT})*

%%

"var"|"int"|"str"|"read"|"print"|"if"|"else"|"do"|"while" {printf("%s - reserved word\n", yytext);}
```

```
"+"|"-"|"*"|"/"|"=="|"<"|">"|"<="|">="|"="|"!="|"%" printf("%s - operator\n", yytext);

{IDENTIFIER} {printf("%s - identifier\n", yytext);}

{BAD_IDENTIFIER} {printf("Error at token %s at line %d\n", yytext, lines); exit(1);}

{INT_CONSTANT} {printf("%s - integer constant\n", yytext);}

{STRING_CONSTANT} {printf("%s - string constant\n", yytext);}

"["|"]"|";"|"("|")"|"{"|"}"|","|":" {printf("%s - separator\n", yytext);}

[ \t]+ {}

[\n]+ {++lines;}

. {printf("Error at token %s at line %d\n", yytext, lines); exit(1);}

%%

int main(int argc, char** argv) {
   if (argc > 1) {
      FILE *file;
      file = fopen(argv[1], "r");
      if (!file) {
         fprintf(stderr, "Could not open %s\n", argv[1]);
         exit(1);
      }
      yyin = file;
```

```
    }


    yylex();
    return 0;
}
```

This lang.lxi file consists of 3 sections:

Definitions

(c libraries)

%%

Rules

(regular expression and the c statement, both on the same line.

When the lexer finds a pattern matching the regular expression, it executes the c statement)

%%

Code

(optional, file opening + error check,

yyin=lexer reads from the file,

yylex() to do the lexical analysis and tokenization)

Command prompt:

First command: flex lang.lxi

This will generate lex.yy.c file

Second command: gcc lex.yy.c

This will generate a.exe

Third command: a.exe p1.txt

This will show us the scanner results. The table with each token and identifier/operator/separator/string constant/integer constant