Github link: https://github.com/pelotazos123/FLCD-erasmus/tree/main/Lab8

Lexical analyzer implemented in C using FLEX

Lang.lxi used:

```
%{
#include <math.h>
#include <string.h>
%}
%option noyywrap

DIGIT [0-9]
NON_ZERO_DIGIT [1-9]
ZERO_DIGIT [0]
LETTER [a-zA-Z]
COMMA [,]
SIGN [+]|[-]

CHAR ({DIGIT}|{LETTER})
NUMBER {ZERO_DIGIT}|{NON_ZERO_DIGIT}{DIGIT}|{SIGN}{DIGIT}
WORD {LETTER}*
CHARACTER "'"{CHAR}"'"
STRING [\"]{CHAR}*[\"]


%%

"break"  {printf( "Reserved word: %s\n", yytext ); }
"case"  {printf( "Reserved word: %s\n", yytext ); }
"char"  {printf( "Reserved word: %s\n", yytext ); }
"const"  {printf( "Reserved word: %s\n", yytext ); }
"final"  {printf( "Reserved word: %s\n", yytext ); }
"default"  {printf( "Reserved word: %s\n", yytext ); }
"do"  {printf( "Reserved word: %s\n", yytext ); }
"while"  {printf( "Reserved word: %s\n", yytext ); }
"if"  {printf( "Reserved word: %s\n", yytext ); }
"else"  {printf( "Reserved word: %s\n", yytext ); }
"double"  {printf( "Reserved word: %s\n", yytext ); }
"float"  {printf( "Reserved word: %s\n", yytext ); }
"int"  {printf( "Reserved word: %s\n", yytext ); }
"long"  {printf( "Reserved word: %s\n", yytext ); }
"short"  {printf( "Reserved word: %s\n", yytext ); }
"for"  {printf( "Reserved word: %s\n", yytext ); }
"printf"  {printf( "Reserved word: %s\n", yytext ); }
"return"  {printf( "Reserved word: %s\n", yytext ); }
"switch"  {printf( "Reserved word: %s\n", yytext ); }
"void"  {printf( "Reserved word: %s\n", yytext ); }
"try"  {printf( "Reserved word: %s\n", yytext ); }
"catch"  {printf( "Reserved word: %s\n", yytext ); }
"var"  {printf( "Reserved word: %s\n", yytext ); }

{STRING} {printf( "String: %s\n", yytext ); }
{WORD} {printf( "Word: %s\n", yytext); }
{CHAR} {printf( "Char: %s\n", yytext ); }
{NUMBER} {printf( "Number: %s\n", yytext ); }
{SIGN} {printf( "Sign: %s\n", yytext ); }


":" {printf( "Operator: %s\n", yytext ); }
```

```
"\\+" {printf( "Operator: %s\n", yytext ); }
"\\-" {printf( "Operator: %s\n", yytext ); }
"*" {printf( "Operator: %s\n", yytext ); }
"/" {printf( "Operator: %s\n", yytext ); }
"%" {printf( "Operator: %s\n", yytext ); }
"~" {printf( "Operator: %s\n", yytext ); }
"&" {printf( "Operator: %s\n", yytext ); }
"\\|" {printf( "Operator: %s\n", yytext ); }
"^" {printf( "Operator: %s\n", yytext ); }
"<<" {printf( "Operator: %s\n", yytext ); }
">>" {printf( "Operator: %s\n", yytext ); }
"!" {printf( "Operator: %s\n", yytext ); }
"&&" {printf( "Operator: %s\n", yytext ); }
"||" {printf( "Operator: %s\n", yytext ); }
"?" {printf( "Operator: %s\n", yytext ); }
"==" {printf( "Operator: %s\n", yytext ); }
"!=" {printf( "Operator: %s\n", yytext ); }
"++" {printf( "Operator: %s\n", yytext ); }
"--" {printf( "Operator: %s\n", yytext ); }
"<=" {printf( "Operator: %s\n", yytext ); }
">=" {printf( "Operator: %s\n", yytext ); }
"<" {printf( "Operator: %s\n", yytext ); }
">" {printf( "Operator: %s\n", yytext ); }
{COMMA} {printf( "Comma: %s\n", yytext ); }
"(" {printf( "Separator: %s\n", yytext ); }
")" {printf( "Separator: %s\n", yytext ); }
"{" {printf( "Separator: %s\n", yytext ); }
"}" {printf( "Separator: %s\n", yytext ); }
"[" {printf( "Separator: %s\n", yytext ); }
"]" {printf( "Separator: %s\n", yytext ); }
";" {printf( "Separator: %s\n", yytext ); }
" " {printf( "Separator: %s\n", yytext ); }
"\t" {printf( "Separator: %s\n", yytext ); }
"=" {printf( "Set: %s\n", yytext ); }

{SIGN}0 {printf("Illegal number. Cannot start with 0 "); return -1; }

.           {printf( "Lexical error. Unrecognized character"); return -1;}


%%
main( argc, argv )
int argc;
char **argv;
{
   ++argv, --argc;
   if ( argc > 0 )
            yyin = fopen( argv[0], "r" );
   else
            yyin = stdin;
   yylex();
}
```

P1 used:

```
int max():

        int a

        int b

        int c
        if (a > b)
                if (a > c)
                        return a
                return c
        if (b > c)
                return b
        return c

int min():
    int a
    int b
    int c

    if (a < b)
        if (a < c)
            return a
        return c

    if (b < c)
        return b
    return c

bool prime(int n):
    for (int i = n; i >= 0; i--)
        if (n%i==0)
            return false
    return true
P2 used:
int gcd():
    int a
    int b
    int x = 1

    for (int l = 1; l <= a & l <= b; i++)
        if (a%i == 0 & n2 % i == 0)
            x = i
    return x

int equation():
    int a
    int b

    int c

    int x1

    int x2

    x1 = (-b + (b^2 – 4 * a * c) ^(1/2))/2

    x2 = (-b - (b^2 – 4 * a * c) ^(1/2))/2

    print(“x1 = “, x1)

    print(“x2 = “, x2)
```