https://github.com/ElinaBarabas/FLCD

```
%{
#include <stdio.h>
#include <string.h>
int lines = 0;
%}

%option noyywrap
%option caseless

DIGIT           [0-9]
WORD            \"[a-zA-Z0-9]*\"
NUMBER                  [+-]?[1-9][0-9]*|0$
CHARACTER       \'[a-zA-Z0-9]\'
CONST           {WORD}|{NUMBER}|{CHARACTER}
ID              [a-zA-Z][a-zA-Z0-9_]{0,7}

%%

and {printf("Reserved word: %s\n", yytext);}
ARRAY  {printf( "Reserved word: %s\n", yytext);}
ELSE    {printf( "Reserved word: %s\n", yytext);}
FOR     {printf( "Reserved word: %s\n", yytext);}
IF      {printf( "Reserved word: %s\n", yytext);}
INT     {printf( "Reserved word: %s\n", yytext);}
OR      {printf( "Reserved word: %s\n", yytext);}
READ    {printf( "Reserved word: %s\n", yytext);}
INPUT   {printf( "Reserved word: %s\n", yytext);}
PRINT   {printf( "Reserved word: %s\n", yytext);}
STRING {printf( "Reserved word: %s\n", yytext);}
```

```
WHILE   {printf( "Reserved word: %s\n", yytext);}

{ID}       {printf( "Identifier: %s\n", yytext );}

{CONST}            {printf( "Constant: %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 );}
"["        {printf( "Separator: %s\n", yytext );}
"]"        {printf( "Separator: %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( "Separator: %s\n", yytext );}
"!"        {printf( "Operator: %s\n", yytext );}
"%"        {printf( "Operator: %s\n", yytext );}
```

```
[ \t]+     {}
[\n]+ {lines++;}

[+-]?0[0-9]* {printf("Illegal constant at line %d\n", lines);}

[a-zA-Z][a-zA-Z0-9]{8,}   {printf("Illegal size of the identifier at line %d\n", lines);}

[0-9~@#$%^][a-zA-Z0-9]{0,7}     {printf("Illegal identifier at line %d\n", lines);}

\'[a-zA-Z0-9]{2,}\' {printf("Character of length >=2 at line %d\n", lines);}

%%


void main(argc, argv)
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();

}
```

```
C:\Users\User\Desktop\Facultate\Anul 3 Semestru 1\FLCD - with repo\Laboratories\Lab 8>gcc lex.yy.c

C:\Users\User\Desktop\Facultate\Anul 3 Semestru 1\FLCD - with repo\Laboratories\Lab 8>a.exe p1.txt
Identifier: a
Separator: =
Constant: 16
Identifier: b
Separator: =
Constant: 21
Reserved word: PRINT
Separator: (
Reserved word: INT
Separator: (
Identifier: a
Operator: *
Identifier: b
Separator: )
Separator: )

C:\Users\User\Desktop\Facultate\Anul 3 Semestru 1\FLCD - with repo\Laboratories\Lab 8>a.exe perr.txt
Identifier: a
Separator: =
Constant: 16
Separator: .
Identifier: b
Separator: =
Reserved word: INT
Separator: (
Constant: 2
Illegal identifier at line 1
Separator: )
Separator: .
Reserved word: PRINT
Separator: (
Reserved word: INT
Separator: (
Identifier: a
Operator: *
Identifier: b
Separator: )
```