Github: https://github.com/Alin-St/FLCD (https://github.com/Alin-St/FLCD)

This is the content of my lxi file (specifications for flex program):

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

%option noyywrap
%option caseless

DIGIT [0-9]
NON_ZERO_DIGIT [1-9]
INTEGER [+-]?{NON_ZERO_DIGIT}{DIGIT}*|0
LETTER [a-zA-Z_]
SIGNS [ !#%^*+-/<=>_.,:;]
STRING_CONSTANT (\"({LETTER}|{DIGIT}|_|{SIGNS})*\")
IDENTIFIER {LETTER}({LETTER}|{DIGIT})*
%%

"int"|"bool"|"int_list"|"if"|"while"|"read"|"write" {printf("RESERVED WORD: %s\n", yytext);}

"<-"|"+"|"-"|"*"|"/"|"%"|"<"|"<="|"="|">"|">="|"and"|"or"|".add"|".get" printf("OPERATOR: %s\n", yytext);

"{"|"}"|"("|")"|";"|"end"|"begin"|"endl" printf("SEPARATORS: %s\n", yytext);

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

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

{STRING_CONSTANT} {printf("STRING CONSTANT: %s\n", yytext);}

[ \t]+ {}

"//"(.)*[\n]+ {++yylineno;}


[\n]+ {++yylineno;}

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

%%

int main(int argc, char** argv) {
    if (argc > 1)
        yyin = fopen(argv[1], "r");
    else
        yyin = stdin;
    yylex();
}
```

How to use it:

1. Create scanner program by running `flex scanner.lxi`. This will generate lex.yy.c file.
2. Compile the result with `gcc -o scanner lex.yy.c`
3. Run the scanner on a given input with `./scanner p1.txt` (replace p1 with the input)

Example

Input:

```
// Compute the minimum of three numbers (a, b, c)
int a;
int b;
int c;

read a;
read b;
read c;

int min;
min <- a;

if (b < min)
{
    min <- b;
}

if (c < min)
{
    min <- c;
}

write min;
```

Output:

```
RESERVED WORD: int
IDENTIFIER: a
SEPARATORS: ;
RESERVED WORD: int
IDENTIFIER: b
SEPARATORS: ;
RESERVED WORD: int
IDENTIFIER: c
SEPARATORS: ;
RESERVED WORD: read
IDENTIFIER: a
SEPARATORS: ;
RESERVED WORD: read
IDENTIFIER: b
SEPARATORS: ;
RESERVED WORD: read
IDENTIFIER: c
SEPARATORS: ;
RESERVED WORD: int
IDENTIFIER: min
SEPARATORS: ;
IDENTIFIER: min
OPERATOR: <-
IDENTIFIER: a
SEPARATORS: ;
RESERVED WORD: if
SEPARATORS: (
IDENTIFIER: b
OPERATOR: <
IDENTIFIER: min
SEPARATORS: )
SEPARATORS: {
IDENTIFIER: min
OPERATOR: <-
IDENTIFIER: b
SEPARATORS: ;
SEPARATORS: }
RESERVED WORD: if
SEPARATORS: (
IDENTIFIER: c
OPERATOR: <
IDENTIFIER: min
SEPARATORS: )
SEPARATORS: {
IDENTIFIER: min
OPERATOR: <-
IDENTIFIER: c
SEPARATORS: ;
SEPARATORS: }
RESERVED WORD: write
IDENTIFIER: min
SEPARATORS: ;
```