Github: https://github.com/ddbl/FLCD/tree/main/Lab%2012

Lang.lxi:

```
%{

#include <stdio.h>

#include <string.h>

int lines = 0;

%}


%option noyywrap

%option caseless


const               (0|-?[1-9][0-9]*)

id                  [a-zA-Z][0-9a-zA-Z_]*


%%


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

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

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

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

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

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

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

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

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

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

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

```
input {printf( "Reserved word: %s\n", yytext);}
for {printf( "Reserved word: %s\n", yytext);}
break {printf( "Reserved word: %s\n", yytext);}
then {printf( "Reserved word: %s\n", yytext);}
var {printf( "Reserved word: %s\n", yytext);}
string {printf( "Reserved word: %s\n", yytext);}
boolean {printf( "Reserved word: %s\n", yytext);}
true {printf( "Reserved word: %s\n", yytext);}
false {printf( "Reserved word: %s\n", yytext);}
while {printf( "Reserved word: %s\n", yytext);}
typedef {printf( "Reserved word: %s\n", yytext);}


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

{id} {printf( "Identifier: %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( "Operator: %s\n", yytext );}
"!"	{printf( "Operator: %s\n", yytext );}



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