Github link: https://github.com/dckrck/flcd_lab9

Lang.lxi

```
---------------------------------------------------------------------------------------------------------------------------
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "y.tab.h"
int line = 1;
%}

IDENTIFIER          [_a-zA-Z][_a-zA-Z0-9]*
INT_CONST           0|[+|-]?[1-9][0-9]*
STRING_CONST                [\"][ a-zA-Z0-9]+[\"]

%%

"+"         {printf("OPERATOR: %s\n", yytext); return plus;}
"-"         {printf("OPERATOR: %s\n", yytext); return minus;}
"*"         {printf("OPERATOR: %s\n", yytext); return multiplication;}
"/"         {printf("OPERATOR: %s\n", yytext); return division;}
"%"         {printf("OPERATOR: %s\n", yytext); return modulo;}
"<"         {printf("OPERATOR: %s\n", yytext); return lessThan;}
"<="        {printf("OPERATOR: %s\n", yytext); return lessThanOrEqual;}
"="         {printf("OPERATOR: %s\n", yytext); return equal;}
">"         {printf("OPERATOR: %s\n", yytext); return moreThan;}
">="        {printf("OPERATOR: %s\n", yytext); return moreThanOrEqual;}
"=="        {printf("OPERATOR: %s\n", yytext); return doubleEqual;}
"!="        {printf("OPERATOR: %s\n", yytext); return notEqual;}
"++"        {printf("OPERATOR: %s\n", yytext); return increment;}
"--"        {printf("OPERATOR: %s\n", yytext); return decrement;}

"["         {printf("SEPARATOR %s\n", yytext); return leftBracket;}
"]"         {printf("SEPARATOR %s\n", yytext); return rightBracket;}
"{"         {printf("SEPARATOR %s\n", yytext); return leftCurlyBracket;}
"}"         {printf("SEPARATOR %s\n", yytext); return rightCurlyBracket;}
"("         {printf("SEPARATOR %s\n", yytext); return leftRoundBracket;}
")"         {printf("SEPARATOR %s\n", yytext); return rightRoundBracket;}
":"         {printf("SEPARATOR %s\n", yytext); return colon;}
";"         {printf("SEPARATOR %s\n", yytext); return semicolon;}
","         {printf("SEPARATOR %s\n", yytext); return comma;}
"'"         {printf("SEPARATOR %s\n", yytext); return apostrophe;}
"\""        {printf("SEPARATOR %s\n", yytext); return quote;}

"if"        {printf("KEYWORD: %s\n", yytext); return IF;}
"else"      {printf("KEYWORD: %s\n", yytext); return ELSE;}
"read"      {printf("KEYWORD: %s\n", yytext); return READ;}
"write"     {printf("KEYWORD: %s\n", yytext); return WRITE;}
"var"       {printf("KEYWORD: %s\n", yytext); return VAR;}
"while"             {printf("KEYWORD: %s\n", yytext); return WHILE;}
"for"       {printf("KEYWORD: %s\n", yytext); return FOR;}
```

```
"break"                 {printf("KEYWORD: %s\n", yytext); return BREAK;}
"return"        {printf("KEYWORD: %s\n", yytext); return RETURN;}
"not"           {printf("KEYWORD: %s\n", yytext); return NOT;}
"in"            {printf("KEYWORD: %s\n", yytext); return IN;}
"continue"      {printf("KEYWORD: %s\n", yytext); return CONTINUE;}
"and"           {printf("Reserved word: %s\n", yytext); return AND;}
"or"            {printf("Reserved word: %s\n", yytext); return OR;}

{IDENTIFIER}            {printf("IDENTIFIER: %s\n", yytext); return IDENTIFIER;}
{INT_CONST}             {printf("INT: %s\n", yytext); return INT_CONST;}
{STRING_CONST}          {printf("STRING: %s\n", yytext); return STRING_CONST;}

[ \t]+          {}
[\n]+    {line++;}

[0-9][a-zA-Z0-9_]*                                      {printf("Identifier cannot start
with a digit, line  %d\n", line);}
[_a-zA-Z]+[.][_a-zA-Z]+                                     {printf("Identifier cannot
contain decimal separator at line %d\n", line);}
[0][0-9]+                                               {printf("Int number cannot start
with 0 at line %d\n", line);}
[0-9]*[.][0-9]                                          {printf("Integer cannot contain
decimal separator (.) at line %d\n", line);}
[\"][a-zA-Z0-9_]+|[a-zA-Z0-9_]+[\"]                         {printf("String should be
closed between \" at line %d\n", line);}

%%
-----------------------------------------------------------------------------------------------------------------
```