Lab 8 – Documentation

Github link: https://github.com/anamariadem/lftc-mini-language/tree/lab8/lex-yacc/lex

Commands used:

- `$ flex  specif.lxi`
- `$ gcc lex.yy.c -o exe –ll`
- `$ ./exe < p1.txt`

specif.lxi file content:

```
%{
#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}


%%

start {printf("Reserved word: %s\n", yytext);}
finish {printf("Reserved word: %s\n", yytext);}
def {printf("Reserved word: %s\n", yytext);}
else   {printf( "Reserved word: %s\n", yytext);}
execute    {printf( "Reserved word: %s\n", yytext);}
while  {printf( "Reserved word: %s\n", yytext);}
if {printf( "Reserved word: %s\n", yytext);}
then   {printf( "Reserved word: %s\n", yytext);}
int    {printf( "Reserved word: %s\n", yytext);}
```

```
char {printf( "Reserved word: %s\n", yytext); }
read   {printf( "Reserved word: %s\n", yytext); }
log    {printf( "Reserved word: %s\n", yytext); }
string {printf( "Reserved word: %s\n", yytext); }
exit   {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( "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 ); }


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

%%
```