Lab 8 - Lex

https://github.com/DanBesu/formal-languages-and-computer-design/tree/main/lab8-lex

Configuration:

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int currentLine = 1;
%}

%option noyywrap

IDENTIFIER          [a-zA-Z][a-zA-Z0-9_]*
NUMBER_CONST        0|[+|-]?[1-9][0-9]*([.][0-9]*)?|[+|-]?0[.][0-9]*
STRING_CONST        [\"][a-zA-Z0-9_]*[\"]
CHAR_CONST          [\'][a-zA-Z0-9_][\']

%%

"declarNumar"           {printf("Reserved word: %s\n", yytext);}
"declarCuvant"  {printf("Reserved word: %s\n", yytext);}
"catTimp"               {printf("Reserved word: %s\n", yytext);}
"saZicemCa"             {printf("Reserved word: %s\n", yytext);}
"dacaNu"                {printf("Reserved word: %s\n", yytext);}
"introdu"               {printf("Reserved word: %s\n", yytext);}
"afiseaza"              {printf("Reserved word: %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);}

"("                     {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);}

{IDENTIFIER}            {printf("Identifier: %s\n", yytext);}
{NUMBER_CONST}              {printf("Number: %s\n", yytext);}
{STRING_CONST}              {printf("String: %s\n", yytext);}
{CHAR_CONST}                {printf("Character: %s\n", yytext);}

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

[0-9_][a-zA-Z0-9_]*             {printf("Illegal identifier at line %d\n", currentLine); return -1;}
[+|-]0              {printf("Illegal numeric constant at line %d\n", currentLine); return -1;}
[+|-]?[0][0-9]*([.][0-9]*)?             {printf("Illegal numeric constant at line %d\n", currentLine); return -1;}
[\'][a-zA-Z0-9_]{2,}[\']|[\'][a-zA-Z0-9_]|[a-zA-Z0-9_][\']                  {printf("Illegal character constant at line
%d\n", currentLine); return -1;}
[\"][a-zA-Z0-9_]+|[a-zA-Z0-9_]+[\"]                 {printf("Illegal string constant at line %d\n", currentLine);
return -1;}

%%

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();
}
```

Commands:

1. Setup the scanner
   ```
   $ lex scanner.lxi
   ```

2. Compile the generated code:

```
$ gcc lex.yy.c -o scanner
```

3. Run the scanner:

```
$ ./scanner input1.txt
$ ./scanner.input2.txt
$ ./scanner.input-err.txt
```