LAB 8

Link github: https://github.com/teodoraarsene/Formal-Languages-and-Compiler-Design/tree/main/labs/lab8

Lex Specification File

```
%{
#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 ][\']

%%

"read"|"write"|"if"|"else"|"for"|"while"|"int"|"string"|"char"|"return"|"start"|"array"
        {printf("Reserved word: %s\n", yytext);}
"+"|"-"|"*"|"/"|"%"|"<="|">="|"=="|"!="|"<"|">"|"="            {printf("Operator: %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);}
[+|-]0                  {printf("Illegal numeric constant at line %d\n", currentLine);}
[+|-]?[0][0-9]*([.][0-9]*)?                     {printf("Illegal numeric constant at line %d\n", currentLine);}
[\'][a-zA-Z0-9 ]{2,}[\']|[\'][a-zA-Z0-9 ][a-zA-Z0-9 ][\']                       {printf("Illegal character constant at line %d\n", currentLine);}
[\"][a-zA-Z0-9_]+|[a-zA-Z0-9_]+[\"]                     {printf("Illegal string constant at line %d\n", currentLine);}

%%

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();

}

Demo

We first run the command:

```
>flex lang.lxi
```

The we run:

```
>gcc lex.yy.c
```

An executable was created after the second command, so now we can run the program.

We have 4 examples for which we can run the program (p1.txt, p2.txt, p3.txt and p1err.txt)

In this demo, I am going to run the program for p2.txt, using the following command:

```
>a.exe p2.txt
```

Where a.exe being the generated executable.

The output is the following

```
Reserved word: start
Separator: {
Reserved word: int
Identifier: a
Separator: ;
Reserved word: int
Identifier: b
Separator: ;
Reserved word: int
Identifier: c
Separator: ;
Reserved word: read
Separator: (
Identifier: a
Separator: )
Separator: ;
Reserved word: read
Separator: (
Identifier: b
Separator: )
Separator: ;
Reserved word: while
Separator: (
Identifier: b
Operator: !=
Number: 0
Separator: )
Separator: {
Identifier: c
Operator: =
Identifier: a
Operator: %
Identifier: b
Separator: ;
Identifier: a
Operator: =
Identifier: b
Separator: ;
Identifier: b
Operator: =
Identifier: c
Separator: ;
Separator: }
```