https://github.com/915-Nichifor-Dragos/FLCD/tree/master/Lab9

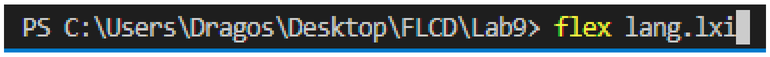
Lex Specification File

```
%{
#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"|"while"|"int"|"string"|"bool"|"return"|"list"|"function"|"and"|"or"|"true"|
"false"         {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();

}
```

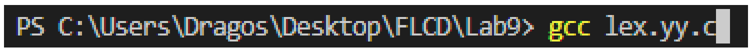
Demo

Run the command in the directory:



```
PS C:\Users\Dragos\Desktop\FLCD\Lab9> flex lang.lxi
```

After the first command, run:

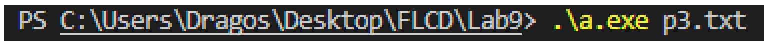


```
PS C:\Users\Dragos\Desktop\FLCD\Lab9> gcc lex.yy.c
```

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

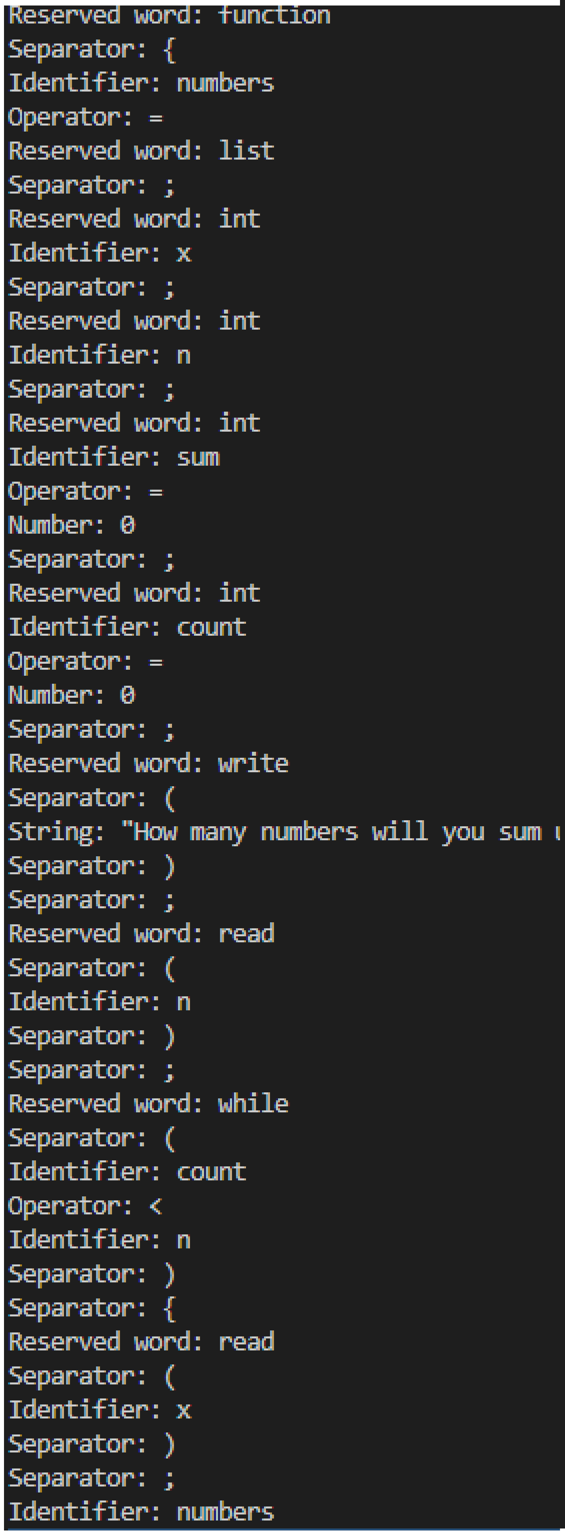
We have 4 examples for which we can run the program (p1.txt, p2.txt, p3.txt and p1err.txt)

In this demo, I am going to run the program for p3.txt, using the following command:

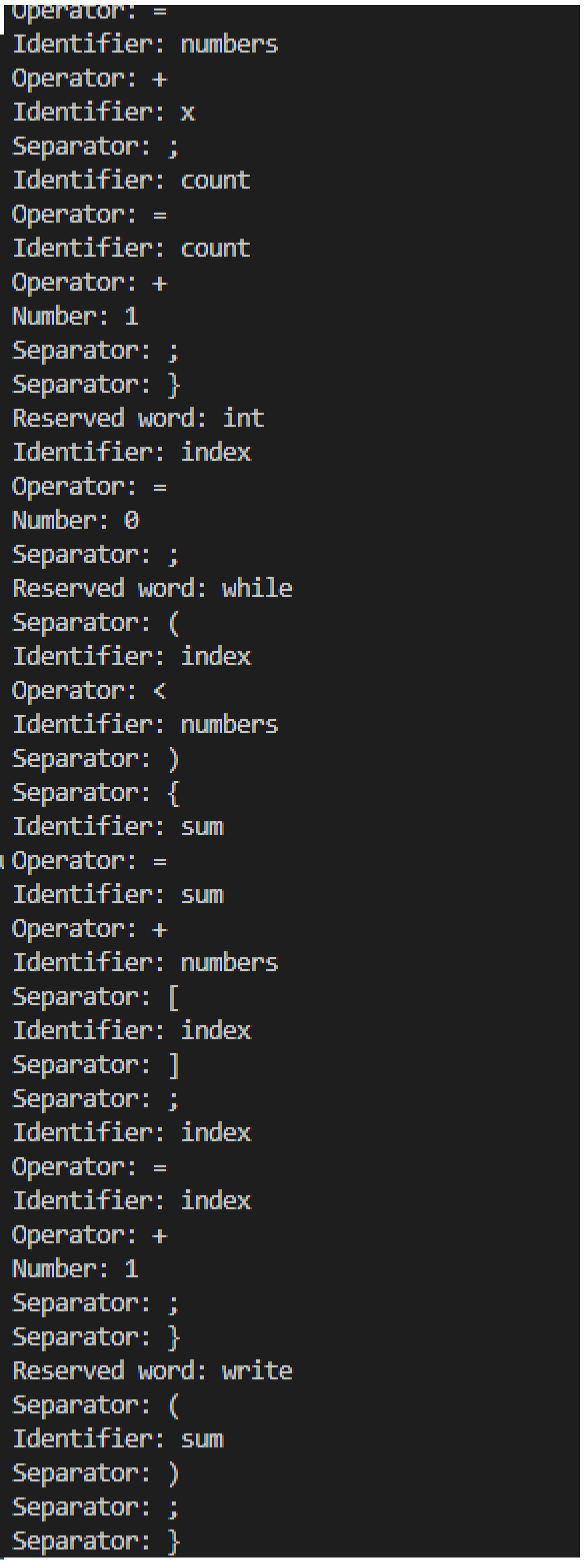


```
PS C:\Users\Dragos\Desktop\FLCD\Lab9> .\a.exe p3.txt
```

Output



```
Reserved word: function
Separator: {
Identifier: numbers
Operator: =
Reserved word: list
Separator: ;
Reserved word: int
Identifier: x
Separator: ;
Reserved word: int
Identifier: n
Separator: ;
Reserved word: int
Identifier: sum
Operator: =
Number: 0
Separator: ;
Reserved word: int
Identifier: count
Operator: =
Number: 0
Separator: ;
Reserved word: write
Separator: (
String: "How many numbers will you sum
Separator: )
Separator: ;
Reserved word: read
Separator: (
Identifier: n
Separator: )
Separator: ;
Reserved word: while
Separator: (
Identifier: count
Operator: <
Identifier: n
Separator: )
Separator: {
Reserved word: read
Separator: (
Identifier: x
Separator: )
Separator: ;
Identifier: numbers
```



```
Operator: =
Identifier: numbers
Operator: +
Identifier: x
Separator: ;
Identifier: count
Operator: =
Identifier: count
Operator: +
Number: 1
Separator: ;
Separator: }
Reserved word: int
Identifier: index
Operator: =
Number: 0
Separator: ;
Reserved word: while
Separator: (
Identifier: index
Operator: <
Identifier: numbers
Separator: )
Separator: {
Identifier: sum
Operator: =
Identifier: sum
Operator: +
Identifier: numbers
Separator: [
Identifier: index
Separator: ]
Separator: ;
Identifier: index
Operator: =
Identifier: index
Operator: +
Number: 1
Separator: ;
Separator: }
Reserved word: write
Separator: (
Identifier: sum
Separator: )
Separator: ;
Separator: }
```