Practical - 5

Aim : Implement Lexical Analyzer using Lex.

Program:

```
%{

%}
invalidID [0-9]+\.?([0-9]*)?([eE][-+]?[0-9]+)?[A-Za-z_]+
invalidSTRING1 \"[^\n\"]*[\n]?[^\n\"]*\"?
invalidSTRINGTEST \"[^\n\"]*[\n]?
invalidSTRING2 \'[^\n\']*[\n]?
literalString1 \"([^\\\"\n]|\\.)*(\\\n)?([^\\\"\n]|\\.)*\"
literalString2 \'([^\\\'\n]|\\.)*\'
literalNumber ([0-9]*)?\.?([0-9]*)?([eE][-+]?[0-9]+)?
separator [\[{};,()\]]
identifier [a-zA-Z][a-zA-Z0-9]*
MultilineComm "/*"([^*]|\*+[^*/])*\*+"/"
SinglellineComm \/\/(.*)
operator [+-/=*%]|->|&&|\|\||\!|\<=|\>=|\<|\==|\>|\+\+|\-\-|\!\=|\<\<|\>\>|&|\||\~|\^|\?\:|\+\=|\-\=|\*\=|\/\=|\%\=
keyword string|auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|int|long|
register|return|short|signed|sizeof|static|struct|switch|typedef|union|unsigned|void|volatile|while
%%
#.* {printf("\n\t%s is a preprocessor directive",yytext);}
{invalidID} {printf("\n\t%s\t\t ------> ERROR in indentifier...", yytext);}
{keyword} {printf("\n\t%s ------> KEYWORD",yytext);}
{SinglellineComm} {printf("\n");}
{MultilineComm} {}
{separator}   {printf("\n\t%s ------> SEPARATOR",yytext);}
{operator}   {printf("\n\t%s ------> OPERATOR",yytext);}
{identifier}([0-9]*)? {printf("\n\t%s ------> IDENTIFIER",yytext);}
{literalString1} {printf("\n\t%s ------> LITERAL STRING",yytext);}
{literalString2} {printf("\n\t%s ------> LITERAL STRING",yytext);}
{invalidSTRINGTEST} {printf("\n\t\t%s\t\t ------> ERROR in string...", yytext);}
{invalidSTRING2} {printf("\n\t\t%s\t\t ------> ERROR in string...", yytext);}
{literalNumber} {printf("\n\t%s ------> LITERAL NUM",yytext);}
%%
int main(int argc, char **argv)
{
        FILE *file;
        if (argv[1] != NULL) {
        printf("\n%s is the file name.\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n",argv[1]);
                file=fopen(argv[1],"r");
        }else{
                printf("please give the file name...");
                return 0;
        }

        if(!file)
        {
                printf("could not open the file");
                exit(0);
```

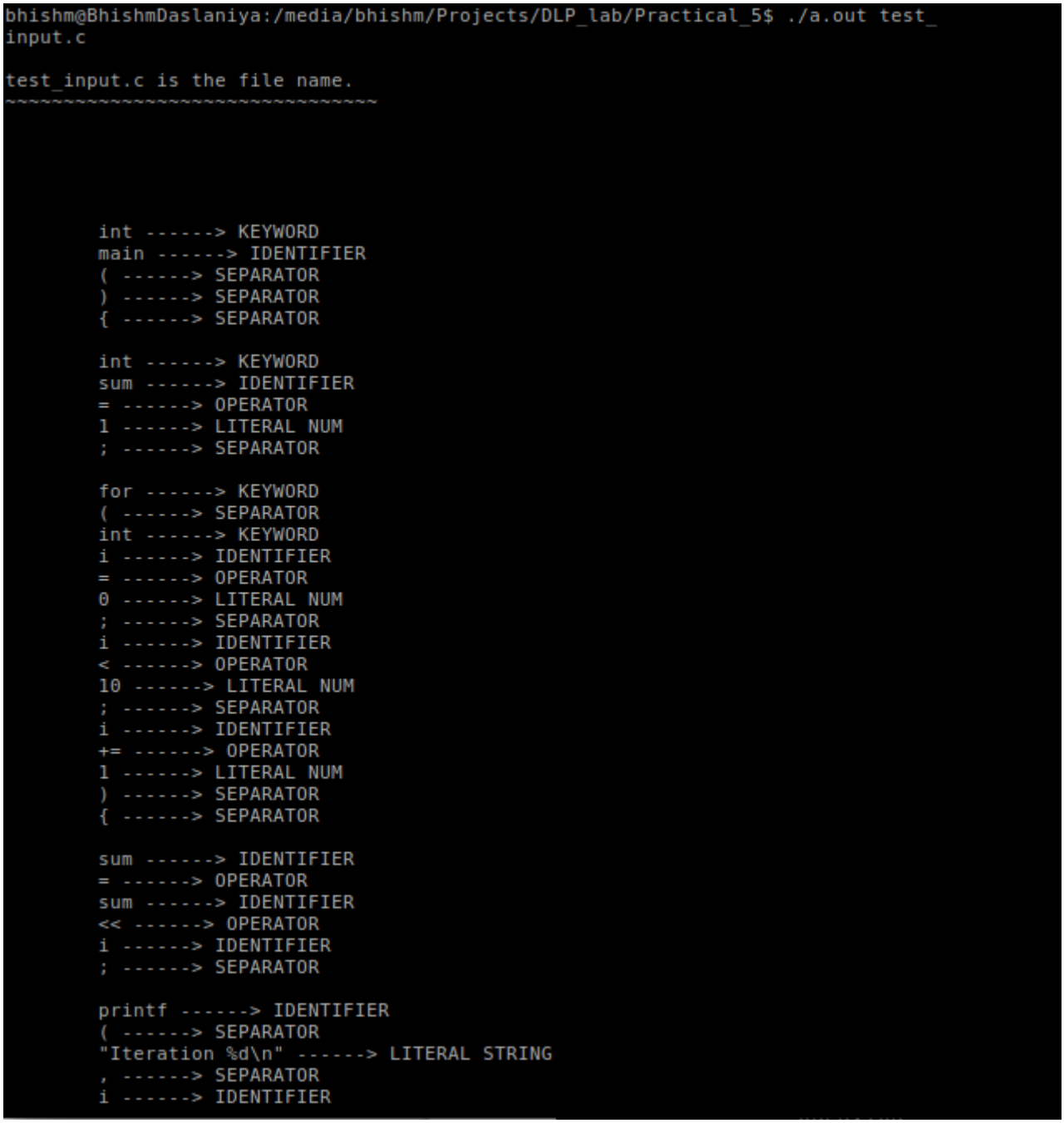
```
        }
        yyin=file;
        yylex();
        printf("\n");
        return(0);
}
int yywrap(){
        return(1);
}
```

Input Program:

```
// Input without preprocessor directives...
/*
        Author : Bhishm Daslaniya [17CE023]
*/

int main(){
        int sum = 1;
        for(int i = 0 ; i < 10; i+=1){
                sum = sum<<i;
                printf("Iteration %d\n",i);
        }
        scanf("sum : %d",&sum);
        float f = 1.2222;
        char c = 'c';
        char ch ='cdcd';
        char str[100] = "Copyright by Bhishm Daslaniya";
        int 1b = 55;
        ++sum;
        return 0;
}
```

Output:


bhishm@BhishmDaslaniya:/media/bhishm/Projects/DLP_lab/Practical_5$ ./a.out test_
input.c
test_input.c is the file name.
int ------> KEYWORD
main ------> IDENTIFIER
(------> SEPARATOR
) ------> SEPARATOR
{ ------> SEPARATOR
int ------> KEYWORD
sum ------> IDENTIFIER
= ------> OPERATOR
1 ------> LITERAL NUM
; ------> SEPARATOR
for ------> KEYWORD
(------> SEPARATOR
int ------> KEYWORD
i ------> IDENTIFIER
= ------> OPERATOR
0 ------> LITERAL NUM
; ------> SEPARATOR
i ------> IDENTIFIER
< ------> OPERATOR
10 ------> LITERAL NUM
; ------> SEPARATOR
i ------> IDENTIFIER
+= ------> OPERATOR
1 ------> LITERAL NUM
) ------> SEPARATOR
{ ------> SEPARATOR
sum ------> IDENTIFIER
= ------> OPERATOR
sum ------> IDENTIFIER
<< ------> OPERATOR
i ------> IDENTIFIER
; ------> SEPARATOR
printf ------> IDENTIFIER
(------> SEPARATOR
"Iteration %d\n" ------> LITERAL STRING
, ------> SEPARATOR
i ------> IDENTIFIER

```
scanf ------> IDENTIFIER
( ------> SEPARATOR
"sum : %d" ------> LITERAL STRING
, ------> SEPARATOR
& ------> OPERATOR
sum ------> IDENTIFIER
) ------> SEPARATOR
; ------> SEPARATOR

float ------> KEYWORD
f ------> IDENTIFIER
= ------> OPERATOR
1.2222 ------> LITERAL NUM
; ------> SEPARATOR

char ------> KEYWORD
c ------> IDENTIFIER
= ------> OPERATOR
'c' ------> LITERAL STRING
; ------> SEPARATOR

char ------> KEYWORD
ch ------> IDENTIFIER
= ------> OPERATOR
'cdcd' ------> LITERAL STRING
; ------> SEPARATOR

char ------> KEYWORD
str ------> IDENTIFIER
[ ------> SEPARATOR
100 ------> LITERAL NUM
] ------> SEPARATOR
= ------> OPERATOR
"Copyright by Bhishm Daslaniya" ------> LITERAL STRING
; ------> SEPARATOR

int ------> KEYWORD
1b              ------> ERROR in indentifier...
= ------> OPERATOR
55 ------> LITERAL NUM
; ------> SEPARATOR

++ ------> OPERATOR
sum ------> IDENTIFIER
; ------> SEPARATOR

return ------> KEYWORD
0 ------> LITERAL NUM
; ------> SEPARATOR

} ------> SEPARATOR
```

Conclusion: From this practical I have learnt about how to impplement lexical analyzer using lex tool.