# Exp 2 : LEXICAL ANALYZER USING LEX TOOL

1. Start
2. Declare global file pointer output
3. Define start conditions : COMMENT BCOMMENT

4. "#include" Preprocessor Directive
5. void | int | return etc ( 32 ) – keyword
6. \"[^\"\n]*\" Strings
7. [0-9]+ Numbers
8. [a-zA-Z][a-zA-Z0-9]* Identifiers
9. [][{}();,] Special Symbols
10. <COMMENT> handle single comments
11. <BCOMMENT> handle block comments
12. . ( rules other than mentioned above ) Operators

13. yywrap() indicate the end of input
14. In the main function - Open input file (input.c) for reading and output file (output.txt) for writing
15. Set the input file for lexical analysis using yyin
16. Call yylex to initiate lexical analysis
17. Stop

<u>Lexical Analyzer code ( Lex )</u>

```
%{
   #include <stdio.h>
   FILE *output;  // Declare the file pointer globally
%}

%x COMMENT BCOMMENT

%%
"#include" { fprintf(output, "%s\tPreprocessor Directive\n", yytext); }
void|int|main|auto|double|struct|break|else|long|switch|case|enum|register|typedef|char|extern|return|
union|const|float|short|unsigned|continue|for|signed|volatile|do|if|static|while { fprintf(output, "%s\
tkeyword\n", yytext); }

\"[^\"]*\" { fprintf(output, "%s\tstring\n", yytext); }
[0-9]+ { fprintf(output, "%s\tdigit\n", yytext); }
[a-zA-Z][a-zA-Z0-9]* { fprintf(output, "%s\tidentifier\n", yytext); }
[][{}();,] { fprintf(output, "%c\tspecial symbol\n", yytext[0]); }
[ \t\n]

"//" { BEGIN(COMMENT); }
<COMMENT>[^\n]*\n { BEGIN(INITIAL); }

"/*" { BEGIN(BCOMMENT); }
<BCOMMENT>"*/" { BEGIN(INITIAL); }

. { fprintf(output, "%c\toperator\n", yytext[0]); }
%%
```

```
int yywrap(){
    return 1;
}

int main() {
    FILE *input = fopen("input.c", "r");
    output = fopen("output.txt", "w");
    if (!input || !output) {
        printf("Could not open the file\n");
        return 1;
    }

    fprintf(output, "token\tlexeme\n--------------\n");
    yyin = input;
    yylex();
            printf("output file created successfuly\n");
    fclose(input);
    fclose(output);
    return 0;
}
```

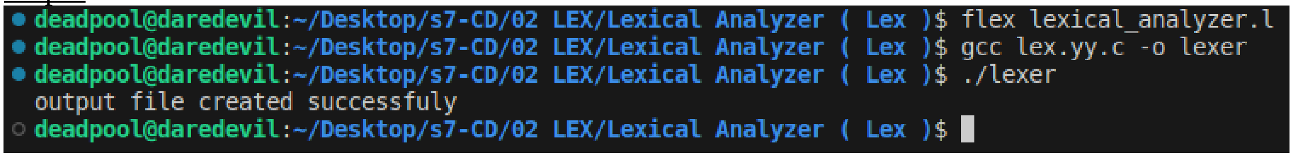
<u>input.c</u>

```
//C-program to calculate sum
#include<stdio.h>
/*This is a comment
block  comment*/
int main(){
    //This is a single line comment
    int num1=5,num2=10;
    int sum=num1+num2;
    printf("sum = %d\n",sum);
    return 0;
}
```

<u>output.txt</u>

```
token   lexeme
--------------
#include        Preprocessor Directive
<       operator
stdio   identifier
.       operator
h       identifier
>       operator
int     keyword
main    keyword
(       special symbol
)       special symbol
{       special symbol
int     keyword
num1    identifier
=       operator
5       digit
,       special symbol
num2    identifier
=       operator
10      digit
;       special symbol
int     keyword
sum     identifier
=       operator
num1    identifier
+       operator
num2    identifier
;       special symbol
printf  identifier
(       special symbol
"sum = %d\n" string
,       special symbol
sum     identifier
)       special symbol
;       special symbol
return  keyword
0       digit
;       special symbol
}       special symbol
```

output



```
● deadpool@daredevil:~/Desktop/s7-CD/02 LEX/Lexical Analyzer ( Lex )$ flex lexical_analyzer.l
● deadpool@daredevil:~/Desktop/s7-CD/02 LEX/Lexical Analyzer ( Lex )$ gcc lex.yy.c -o lexer
● deadpool@daredevil:~/Desktop/s7-CD/02 LEX/Lexical Analyzer ( Lex )$ ./lexer
  output file created successfuly
○ deadpool@daredevil:~/Desktop/s7-CD/02 LEX/Lexical Analyzer ( Lex )$ 
```