Compiler Construction

Practical 1

**Aim:**To implement lexical analyse to recognize all distinct token classes: use flex/lex tool to recognize all distinct token classes (Data type, Identifier, constant (Integer, Float, Char, String), Operator (Arithmetic, Relational, Assign, Unary +/-, Increment), Single line/Multi-line comments, Special symbol(;,{}())) .

Generate Lexical error reports for invalid lexeme.

**Input:**

%{

#include <stdio.h>

int word\_count = 0;

int line\_count = 0;

int char\_count = 0;

%}

%%

int {printf("int"); word\_count++; char\_count+=yyleng;}

float {printf("floatType"); word\_count++; char\_count+=yyleng;}

char {printf("charType");}

string {printf("StringType");}

main {printf("main"); word\_count++; char\_count+=yyleng;}

return {printf("return"); word\_count++; char\_count+=yyleng;}

include<stdio.h> {printf("include<stdio.h>"); word\_count++; char\_count+=yyleng;}

include<string> {printf("include<string>"); word\_count++; char\_count+=yyleng;}

[0-9]+ {printf("NUM"); char\_count+=yyleng;}

[a-zA-Z]+[a-zA-Z0-9]\* {printf("identifier"); word\_count++; char\_count+=yyleng;}

[0-9]\*[.][0-9]+ {printf("float");}

['][a-zA-Z0-9 !@#$%^&\*()\_+-=/|\\?\`]['] {printf("charConst");}

["][a-zA-Z0-9 !@#$%^&\*()\_+-=/|\\?\`\[\]\{\}<>?,.~]\*["] {printf("String");}

[/]{2}.\* {printf("Comment");}

"/\*"([^\*]|\\*+[^\*/])\*\\*+"/" {printf("MultiLineComment");}

[\+\-\\*\/\^\%] {printf("operator"); char\_count+=yyleng;}

\n {printf("\n");line\_count++;}

. {printf("Lexical Error: Not Found");char\_count++;}

%%

int yywrap(){

return 1;

}

int main(){

yylex();

printf("line count: %d\n", line\_count);

printf("word count: %d\n", word\_count);

printf("char count: %d\n", char\_count);

return 0;

}

**Output:**





