**Practical 1**

**Compiler Construction**

2CS701

**Mistry Unnat**

20BCE515



Department of Computer Science and Engineering

Institute of Technology

Nirma University

Ahmedabad

**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(;,{}())) .**

**lex.l :**

%{

#include <stdio.h>

%}

DATATYPES       (int|float|double|char)

ID              [A-Za-z][a-zA-Z0-9\_]\*

CONSTANT        ([0-9]+|[0-9]+.[0-9]+|'[a-zA-Z]+')

SIN\_LINE\_CMT    "//".\*

MUL\_LINE\_CMT    "/\*".\*"\*/"

OPERATOR        ("+"|"-"|"/"|"\*"|">>"|"<<"|"++"|"--"|"&&"|"||"|"="|"==")

SPECIAL\_SYM     [{}();.?]

%%

\n      {}

" "     {}

{DATATYPES}     { printf("\tDatatype : %s\n", yytext); }

{ID}            { printf("\tID : %s\n", yytext);}

{CONSTANT}      { printf("\tConstant : %s\n", yytext); }

{OPERATOR}      { printf("\tOperator : %s\n", yytext); }

{SIN\_LINE\_CMT}  { printf("\tSingle Line Commnent : %s\n", yytext); }

{MUL\_LINE\_CMT}  { printf("\tMulti Line Comment : %s\n", yytext); }

{SPECIAL\_SYM}   { printf("\tSpecial Symbol : %s\n", yytext); }

%%

int main(){

    yylex();

    return 0;

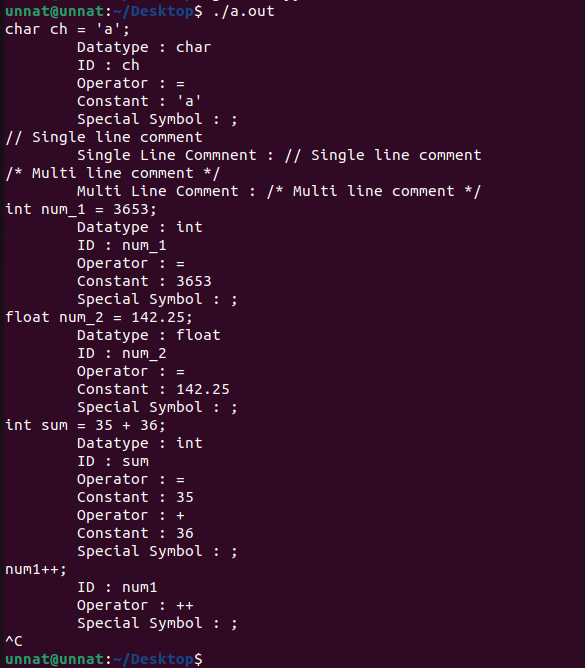
}

int yywrap(){

    return 0;

}

**Output :**

****

**Conclusion :**

From this practical, we learnt what is lexical analysis and wrote a lex code to identify various token classes.