# Ex No: 3

# Date:

**DEVELOP A LEXICAL ANALYZER TO RECOGNIZE TOKENS** **USING LEX TOOL**

**AIM:**

To implement the program to identify C keywords, identifiers, operators, end statements like [], {} using LEX tool.

**ALGORITHM:**

* Define patterns for C keywords, identifiers, operators, and end statements using regular expressions. Use %option noyywrap to disable the default behavior of yywrap.
* Utilize regular expressions to match patterns for C keywords, identifiers, operators, and end statements. Associate each pattern with an action to be executed when matched.
* Define actions to print corresponding token categories for matched patterns. Handle special cases like function declarations, numeric literals, and processor directives separately.
* Open the input file (sample.c in this case) for reading. Start lexical analysis using yylex() to scan the input and apply defined rules.
* Increment a counter (n) each time a newline character is encountered. Print the total number of lines at the end of the program execution.

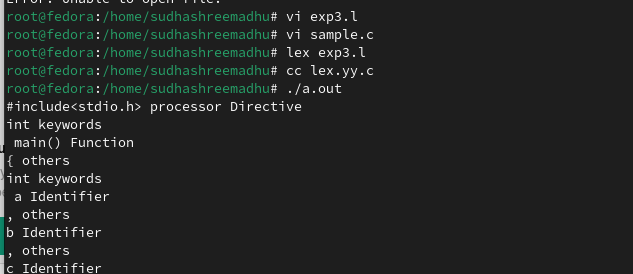
**PROGRAM:**

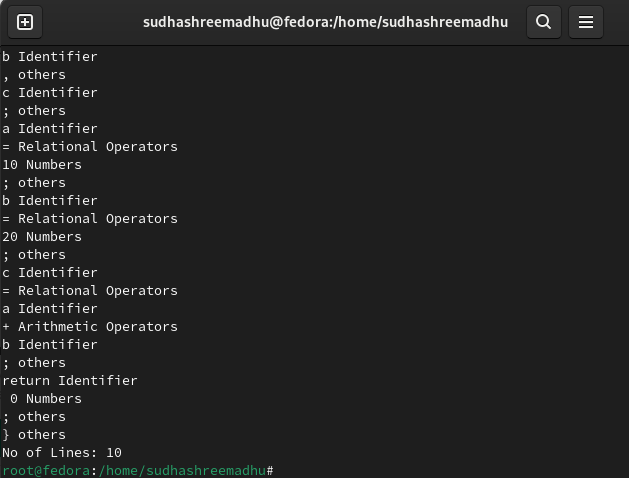
%{  
#include <stdio.h>  
int n = 0;  
%}  
  
%option noyywrap  
  
letter [a-zA-Z]  
digit [0-9]  
id [\_a-zA-Z][\_a-zA-Z0-9]\*  
AO [+|\-|/|%|\*]  
RO [<|>|<=|>=|==]  
pp [#]  
  
%%  
  
"void"      printf("%s return type\n", yytext);  
{letter}\*[(][)]  printf("%s Function\n", yytext);  
"int"|"float"|"if"|"else"  printf("%s keywords\n", yytext);  
"printf"    printf("%s keywords\n", yytext);  
{id}        printf("%s Identifier\n", yytext);  
{digit}+    printf("%s Numbers\n", yytext);  
{AO}        printf("%s Arithmetic Operators\n", yytext);  
{RO}        printf("%s Relational Operators\n", yytext);  
{pp}{letter}\*[<]{letter}\*[.]{letter}[>]  printf("%s processor Directive\n", yytext);  
[\n]        n++;  
[.,\}\{\;]  printf("%s others\n", yytext);  
  
%%  
  
int main() {  
    yyin = fopen("sample.c", "r");  
    if (!yyin) {  
        fprintf(stderr, "Error: Unable to open file.\n");  
        return 1;  
    }  
    yylex();  
    printf("No of Lines: %d\n", n);  
    fclose(yyin);  
    return 0;  
}

**OUTPUT:**

3

3

****



**RESULT:**