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

```
%option noyywrap
letter [a-zA-Z]
digit [0-9]
id [_|a-zA-Z]
AO [+|-|/|%|*]
RO  [<|>|<=|>=|==]
pp [#]
%{
int n=0;
%}

%%
"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}({id}|{digit})*          printf("%s Identifier\n",yytext);
{digit}{digit}*                    printf("%d 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");
        yylex();
        printf("No of Lines %d\n",n);
}
```

OUTPUT

```
[student@localhost ~]$ vi cd2.l
[student@localhost ~]$ lex cd2.l
[student@localhost ~]$ cc lex.yy.c
[student@localhost ~]$ ./a/out
bash: ./a/out: No such file or directory
[student@localhost ~]$ ./a.out
#include<studio.h> processor directive
int keywords
 main() function
{ others
int keywords
 i identifier
= relational operator
10 Number
; others
printf keywords
("% Arithmetic operator
d identifier
", others
i identifier
); others
return identifier
 0 Number
; others
} others
No of Lines 7
[student@localhost ~]$
```

RESULT