Exp 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:

1. Initialize a variable n to count the number of lines.
2. Define patterns for letters, digits, identifiers, arithmetic operators (AO), relational operators (RO), preprocessor directives (pp), and other symbols.
3. Define actions to perform when a pattern is matched and display the corresponding pattern type.
4. Open the file "sample.c" for reading and invoke lexical analysis with yylex().
5. Count the number of newline characters encountered and store it in n.
6. Display the number of lines, n.

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);

“printf”("int"|"float"|"if"|"else"  printf("%s keywords\n",yytext);
  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);}
```

Roll Number: 210701075
Name: Harish R