Ex No: 3
Date:13/2/24

DEVELOP A LEXICAL ANALYZER TO RECOGNIZE TOKENSUSING 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:

```
thirueswaran@thirueswaran-Inspiron-3443:~$ vi sample.c
thirueswaran@thirueswaran-Inspiron-3443:~$ lex 290_ex3.l
thirueswaran@thirueswaran-Inspiron-3443:~$ cc lex.yy.c
thirueswaran@thirueswaran-Inspiron-3443:~$ ./a.out
#include Identifier
< Relational Operators
conio Identifier
. others
h Identifier
> Relational Operators
int keywords
 main()  Function
{ others
  int keywords
 b Identifier
, others
v Identifier
, others
c Identifier
; others
} others
No of Lines 5
```

RESULT:

Thus to develop a lexical analyzer to recognize tokens using lex tool has been executed successfully.