**Implementation of Lexical Analyzer using Lex Tool**

**AIM:** Implementation of Lexical Analyzer using Lex Tool

**ALGORITHM**

1. Start
2. Write regular expression for identifier, number,whitespace and for the operators
3. Read the file
4. Compare each lexeme against the defined Regular expression and output the respective token
5. Stop

**PROGRAM**

// Program name as “lexicalfile.l”

%{

#include<stdio.h>

%}

delim [\t]

ws {delim}+

letter [A-Za-z]

digit [0-9]

id {letter}({letter}|{digit})\*

num {digit}+(\.{digit}+)?(E[+|-]?{digit}+)?

%%

ws {printf("no action");}

if|else|then {printf("%s is a keyword",yytext);} // TYPE 32 KEYWORDS

{id} {printf("%s is a identifier",yytext);}

{num} {printf(" it is a number");}

"<" {printf("it is a relational operator less than");}

"<=" {printf("it is a relational operator less than or equal");}

">" {printf("it is a relational operator greater than");}

">=" {printf("it is a relational operator greater than");}

"==" {printf("it is a relational operator equal");}

"<>" {printf("it is a relational operator not equal");}

%%

main()

{

yylex();

}

**OUTPUT**

Lex lexicalfile.l

gcc lex.yy.c -ll

if

if is a keyword

number is a identifier

254

It is a number

<>

it is a relational operator not equal