

LEXICAL ANALYSIS

M RAJASOUMYA

ch.en.u4cse22049

Experiment No: 01

Aim: To implement Lexical Analyzer Using Lex Tool

CODE

```
UW PICO 5.09
%{
#include <stdio.h>
FILE *yyin;
}%

digit      [0-9]
letter     [a-zA-Z]
id         {letter}{letter}{digit}*
number     {digit}+
operator   [+\\-\\*/=]
rel_op     [<|>|<=|>=|!=|==]

%%

"int"      {printf("Keyword: %s\\n", yytext);}
"main"     {printf("Keyword: %s\\n", yytext);}
"if"       {printf("Keyword: %s\\n", yytext);}
"else"     {printf("Keyword: %s\\n", yytext);}
"while"    {printf("Keyword: %s\\n", yytext);}
"for"      {printf("Keyword: %s\\n", yytext);}

 "("       {printf("left paren: %s\\n", yytext);}
 ")"       {printf("right paren: %s\\n", yytext);}
 "{"       {printf("left brace: %s\\n", yytext);}
 "}"       {printf("right brace: %s\\n", yytext);}
 ";"       {printf("semicolon: %s\\n", yytext);}

{id}       {printf("identifier: %s\\n", yytext);}
{number}   {printf("number: %s\\n", yytext);}
{operator} {printf("operator: %s\\n", yytext);}
{rel_op}   {printf("relational operator: %s\\n", yytext);}
"#include".* {printf("Header File: %s\\n", yytext);}
[ \\t\\n]   ; // ignore spaces, tabs, newlines

%%

int main(int argc, char **argv)
{
    if(argc > 1)
    {
        yyin = fopen(argv[1], "r");
    }
    yylex();
    return 0;
}

int yywrap() {
    return 1;
}
```

OUTPUT

```
(base) rajasoumya@Rajasoumyas-MacBook-Air ~ % nano leex.l
(base) rajasoumya@Rajasoumyas-MacBook-Air ~ % lex leex.l
(base) rajasoumya@Rajasoumyas-MacBook-Air ~ % gcc lex.yy.c -o try
(base) rajasoumya@Rajasoumyas-MacBook-Air ~ % nano test.c
(base) rajasoumya@Rajasoumyas-MacBook-Air ~ % ./try test.c
Header File: #include <stdio.h>
Keyword: int
Keyword: main
left paren: (
right paren: )
left brace: {
Keyword: int
identifier: a
operator: =
number: 5
semicolon: ;
Keyword: int
identifier: b
operator: =
number: 10
semicolon: ;
Keyword: if
left paren: (
identifier: a
relational operator: <
identifier: b
right paren: )
left brace: {
identifier: printf
left paren: (
"identifier: a
identifier: is
identifier: smaller
\identifier: n
"right paren: )
semicolon: ;
right brace: }
Keyword: else
left brace: {
identifier: printf
left paren: (
"identifier: b
identifier: is
identifier: smaller
\identifier: n
"right paren: )
semicolon: ;
right brace: }
identifier: return
number: 0
semicolon: ;
right brace: }
```

RESULT

Lexical Analysis is successfully implemented for the given code snippet

