**SSN COLLEGE OF ENGINEERING, KALAVAKKAM**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**UCS1602 - Compiler Design Programming**

**Assignment-2**

**Implementation of Lexical Analyzer using LEX tool**

**Name:** Rahul Ram M

**Class:** CSE - B

**Reg No:** 185001121

**Date:** 16/02/2021

**CODE:**

**lextokens.l**

%{

#include<stdio.h>

%}

digit [0-9]

letter [A-Za-z]

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

keywords auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|int|long|register|return|short|signed|sizeof|static|struct|switch|typedef|union|unsigned|void|volatile|while

fcalls {letter}+"(".\*")"

single\_line\_comment "//".\*

integer {digit}+

float {digit}+\.{digit}+

string \"({letter}|{digit})\*\"

multiple\_line\_comment "/\*"([^\*])\*"\*/"

aop "+"|"-"|"\*"|"/"|"%"

aaop "+="|"-="|"\*="|"/="|"%="

lop "&&"|"||"|"!"

relop "<"|"<="|">"|">="|"=="|"!="

bitop "^"|"&"|"|"|"<<"|">>"

unop "-"|"++"|"--"

assop "="

spec ";"|","|"."|"["|"]"|"("|")"|"{"|"}"|"["|"]"

prepro #.\*

%%

{integer} { printf("\n%30s - Integer Value",yytext); }

{keywords} { printf("\n%30s - Keyword",yytext);}

(if|while|for|sizeof|switch)/"(".\*")" { printf("\n%30s - Keyword",yytext);}

{identifier} { printf("\n%30s - Identifier",yytext);}

{single\_line\_comment} { printf("\n%30s - Single line Comment",yytext);}

{float} { printf("\n%30s - Float Value",yytext); }

{string} { printf("\n%30s - String Value",yytext); }

{multiple\_line\_comment} { printf("\n%30s - Multiple line comment",yytext);}

{aop} { printf("\n%30s - Arithmetic Operators%s",yytext);}

{aaop} { printf("\n%30s - Arithmetic assignment operators",yytext);}

{lop} { printf("\n%30s - Logical operators",yytext);}

{relop} { printf("\n%30s - Relational operators",yytext);}

{bitop} { printf("\n%30s - Bitwise operators",yytext);}

{unop} { printf("\n%30s - Unary operators",yytext);}

{assop} { printf("\n%30s - Assignment operator",yytext);}

{spec} { printf("\n%30s - Special character",yytext);}

{fcalls} { printf("\n%30s - Function calls",yytext);}

{prepro} { printf("\n%30s - Preprocessor Directive",yytext);}

.|\n { }

%%

void main(){

FILE \*f;

f=fopen("test.c","r");

yyin=f;

yylex();

}

int yywrap(){return(1);}

**Sample Input:**

**test.c**

int main()

{

int a=10,b=20;

if(a>b)

printf("a is greater");

else

printf("b is greater");

hello(int w);

}

**Sample Output:**

int - Keyword

main() - Function calls

{ - Special character

int - Keyword

a - Identifier

= - Assignment operator

10 - Integer Value

, - Special character

b - Identifier

= - Assignment operator

20 - Integer Value

; - Special character

if - Keyword

( - Special character

a - Identifier

> - Relational operators

b - Identifier

) - Special character

printf("a is greater") - Function calls

; - Special character

else - Keyword

printf("b is greater") - Function calls

; - Special character

hello(int w) - Function calls

; - Special character

} - Special character

**Learning Outcomes:**

This assignment helped me

1. To understand the lexical analysis in the coding level.

2. To implement lexical analysis using flex tool.

3. To extract tokens from a c file using flex.

4. To understand the difference in implementation between function calls and condition statements.