**Exp 3**

//INPUT PROGRAM

#inlude<stdio.h>

void main()

{

a=5;

b=3;

c=a+b;

}

**Code:**

/\*TITLE:PROGRAM TO IMPLEMENT LEXICAL ANALYSER\*/

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

#include<ctype.h>

#include<math.h>

struct lex

{

char name[30],type[30];

}l[20];

struct identifier

{

char f[8];

}id[10];

void main()

{

FILE \*fp;

char token[20]={0}, opr[20]={0};

char c,ch;

int i,j,m=0,p=0,s=0,flag=0,flag1=0,flag2=0;

int k=0,z=0,v=0,chk=0,op=0, avl=0;

float lit1[10]={0};

char lit[20]={0};

float u;

fp=fopen("/home/cl03-02/pallavi/plus.c","r");

strcpy(l[0].name,"void");

strcpy(l[1].name,"char");

strcpy(l[2].name,"int");

strcpy(l[3].name,"float");

strcpy(l[4].name,"if");

strcpy(l[5].name,"else");

strcpy(l[6].name,"for");

strcpy(l[7].name,"do");

strcpy(l[8].name,"break");

strcpy(l[9].name,"+");

strcpy(l[10].name,"-");

strcpy(l[11].name,"\*");

strcpy(l[12].name,"/");

strcpy(l[13].name,"%");

strcpy(l[14].name,"=");

strcpy(l[15].name,";");

strcpy(l[16].name,"(");

strcpy(l[17].name,")");

strcpy(l[18].name,"{");

strcpy(l[19].name,"}");

strcpy(l[20].name,",");

i=0;

printf("---------------------------------------");

printf("\n\tUniform Symbol Table");

printf(" \n\t--------------------\n");

printf("\nSymbol\t\tType\t\tIndex\n");

while(!feof(fp))

{

c=fgetc(fp);

//FOR LITERALS

while (isdigit(c))

{

lit[m]=c;

m++;

c=fgetc(fp);

if(c=='.')

{

lit[m]=c;

m++;

c=fgetc(fp);

}

if(c==';')

{

lit[m]='\0';

m++;

u=atof(lit);

lit1[v]=u;

printf("\n%0.1f\tliteral\t\t%d",lit1[v],v+1);

v++;

m=0;

}

}

i=0;

while(isalpha(c))

{

if(c==' '||c==','||c==';')

{

token[i]='\0';

}

token[i]=c;

i++;

c=fgetc(fp);

}

token[i]='\0';

//FOR DISPLAYING KEYWORDS

for(i=0;i<=20;i++)

{

j=strcmp(token,l[i].name);

if(j==0)

{

flag=1;

printf("\n%s\tkeyword\t",token);

printf("\t%d ",i);

break;

}

}

s=0;

ch=token[s];

if(!isdigit(ch)&&flag==0 && c!='('&&c!=')'&&c!='{'&&c!='}'&&c!='\t'&&c!=' '&&c!=','&&c!=';'&&c!='\n'&&c!='#'&&c!='<'&&c!='.'&&c!='>')

{

for(j=0;j<9;j++)

{

z=strcmp(id[j].f,token);

if(z==0)

{

flag1=1;

}

}

if(flag1==0)

{

strcpy(id[k].f,token);

printf("\n%s\t Identifier\t%d",id[k].f,k+1);

k++;

}

}

if(c=='('||c==')'||c=='{'||c=='}'||c==';'||c==','||c=='+'||c=='-'||c=='%'||c=='/'||c=='\*'||c=='=')

{

i=0;

token[i]=c;

i++;

token[i]='\0';

for(i=9;i<=20;i++)

{

chk=strcmp(token,l[i].name);

if(chk==0)

{

avl=1;

break;

}

}

if(avl==1)

{

for(j=0;j<=12;j++)

{

if(c==opr[j])

{

flag2=1;

break;

}

}

if (flag2==0)

{

printf("\n%s\tOperator\t%d",token,i);

opr[op]=c;

op++;

}

}

}

i=0;

flag=0;

flag1=0;

flag2=0;

avl=0;

}

//DISPLAYING LITERAL TABLE

printf("\n\n-------------------------------------------------\n\n");

printf("\n\n\tLiteral Table");

printf("\n\t-------------------\n");

printf("\nLiteral No.\tLiteral Name\n");

for(i=0;i<v;i++)

{

printf("\n%d\t\t%0.1f",i+1,lit1[i]);

}

//DISPLAYING IDENTIFIER TABLE

printf("\n\n--------------------------------------------------\n\n");

printf("\n\n\tIdentifier Table");

printf("\n\t-------------------\n");

printf("\nIdentifier No.\tIdentifier Name\n");

for(i=0;i<k;i++)

{

printf("\n%d\t\t%s",i+1,id[i].f);

}

printf("\n----------------------------------------------------");

fclose(fp);

}

**Output:**

cl03-02@cl0302-HP-Pro-3090-MT:~/pallavi$ gcc table.c

cl03-02@cl0302-HP-Pro-3090-MT:~/pallavi$ ./a.out

---------------------------------------

Uniform Symbol Table

--------------------

Symbol Type Index

void keyword 0

( Operator 16

) Operator 17

{ Operator 18

a Identifier 1

= Operator 14

5.0 literal 1

; Operator 15

b Identifier 2

3.0 literal 2

c Identifier 3

+ Operator 9

} Operator 19

-------------------------------------------------

Literal Table

-------------------

Literal No. Literal Name

1 5.0

2 3.0

--------------------------------------------------

Identifier Table

-------------------

Identifier No. Identifier Name

1 a

2 b

3 c

----------------------------------------------------