****

Compiler Design Lab Manual

**PROGRAM -1**

**Write a LEX Program to scan reserved word & Identifiers of C Language**

%{

#include<stdio.h>

%}

%%

bool|int|float printf("Keyword");

[+%\*/]+ printf("operator");

[0-9]+ printf("Constants");

[,.'"]+ printf("Punctuation Chars");

[!@#$^&()]+ printf("Special Chars");

[a-zA-Z]+ printf("Identifiers");

%%

int yywrap( )

{

return 1;

}

int main()

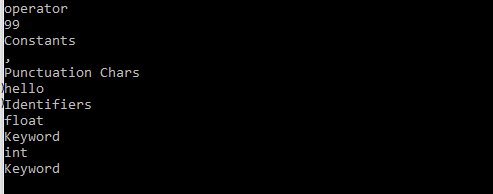
{

yylex();

return 0;

}

o/p:



**PROGRAM -2:**

**Implement Predictive Parsing algorithm**

#include<stdio.h>

#include<string.h>

#include<ctype.h>

char input[10];

int i,error;

void E();

void T();

void Eprime();

void Tprime();

void F();

void main()

{

i=0;

error=0;

printf("Enter an arithmetic expression : "); // Eg: a+a\*a

gets(input);

E();

if(strlen(input)==i&&error==0)

printf("\nAccepted..!!!\n");

else printf("\nRejected..!!!\n");

}

void E()

{

T();

Eprime();

}

void Eprime()

{

if(input[i]=='+')

{

i++;

T();

Eprime();

}

}

void T()

{

F();

Tprime();

}

void Tprime()

{

if(input[i]=='\*')

{

i++;

F();

Tprime();

}

}

void F()

{

if(isalnum(input[i]))i++;

else if(input[i]=='(')

{

i++;

E();

if(input[i]==')')

i++;

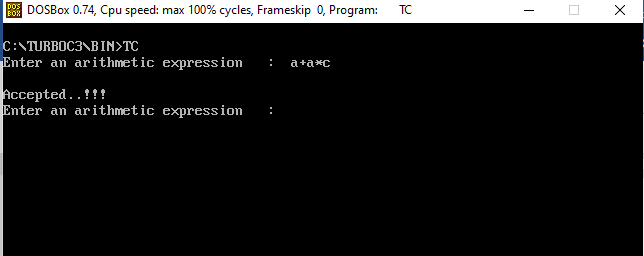
else error=1;

}

else error=1;

}

Output :



**PROGRAM -3:**

**Write a C program to generate three address code.**

#include<stdio.h>

#include<string.h>

void pm();

void plus();

void div();

int i,ch,j,l,addr=100;

char ex[10], exp[10] ,exp1[10],exp2[10],id1[5],op[5],id2[5];

void main()

{

clrscr();

while(1)

{

printf("\n1.assignment\n2.arithmetic\n3.relational\n4.Exit\nEnter the choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\nEnter the expression with assignment operator:");

scanf("%s",exp);

l=strlen(exp);

exp2[0]='\0';

i=0;

while(exp[i]!='=')

{

i++;

}

strncat(exp2,exp,i);

strrev(exp);

exp1[0]='\0';

strncat(exp1,exp,l-(i+1));

strrev(exp1);

printf("Three address code:\ntemp=%s\n%s=temp\n",exp1,exp2);

break;

case 2:

printf("\nEnter the expression with arithmetic operator:");

scanf("%s",ex);

strcpy(exp,ex);

l=strlen(exp);

exp1[0]='\0';

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

{

if(exp[i]=='+'||exp[i]=='-')

{

if(exp[i+2]=='/'||exp[i+2]=='\*')

{

pm();

break;

}

else

{

plus();

break;

}

}

else if(exp[i]=='/'||exp[i]=='\*')

{

div();

break;

}

}

break;

case 3:

printf("Enter the expression with relational operator");

scanf("%s%s%s",&id1,&op,&id2);

if(((strcmp(op,"<")==0)||(strcmp(op,">")==0)||(strcmp(op,"<=")==0)||(strcmp(op,">=")==0)||(strcmp(op,"==")==0)||(strcmp(op,"!=")==0))==0)

printf("Expression is error");

else

{

printf("\n%d\tif %s%s%s goto %d",addr,id1,op,id2,addr+3);

addr++;

printf("\n%d\t T:=0",addr);

addr++;

printf("\n%d\t goto %d",addr,addr+2);

addr++;

printf("\n%d\t T:=1",addr);

}

break;

case 4:

exit(0);

}

}

}

void pm()

{

strrev(exp);

j=l-i-1;

strncat(exp1,exp,j);

strrev(exp1);

printf("Three addres code:\ntemp=%s\ntemp1=%c%ctemp\n",exp1,exp[j+1],exp[j]);

}

void div()

{

strncat(exp1,exp,i+2);

printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n",exp1,exp[i+2],exp[i+3]);

}

void plus()

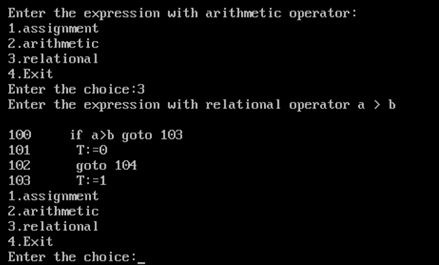
{

strncat(exp1,exp,i+2);

printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n",exp1,exp[i+2],exp[i+3]);

}

**Output :**



**Program – 4**

Implementation of SLR Parsing Technique.

# include <conio.h>

void main()

{

char p[20][10];

char item[20];

int i,k,j,m,n,l,length=0,states=-1;

char prev,current;

clrscr();

printf("\n Implementation of LR Predictive Parsing Technique");

printf("\n Enter number of productions of CFG:\n");

scanf("%d",&n);

printf("\n Enter %d productions:\n",n);

for(m=1;m<=n;m++)

{

scanf("%s",p[m]);

l=strlen(p[m]);

if(l>length)

length=l;

}

for(k=3;k<=length;k++)

{

prev= ' ';

for(m=1;m<=n;m++)

{

current=p[m][k-1];

l=strlen(p[m]);

if(k<=l)

{

for(j=0,i=0;i<l;i++)

{

if(i==k)

item[j++]='.';

item[j++]=p[m][i];

}

if(k==l)

item[j++]='.';

item[j]='\0';

if(current != prev )

printf("\n S%d: ",++states);

printf("\n %s",item);

prev=current;

}

}

}

getch();

}

Output :



**Program – 6**

**Implementation of DAG representation of Infix Expression.**

# include <stdio.h>

# include <conio.h>

char s[20];

char symbols[5]={'K','A','R','I','M'};

int k=0;

int SubExpression(int index)

{

int i,j;

if(islower(s[index-1]))

printf("\n %c = MakeLeaf(%c,%c.val)",s[index-1],s[index-1],s[index-1]);

if(islower(s[index+1]))

printf("\n %c = MakeLeaf(%c,%c.val)",s[index+1],s[index+1],s[index+1]);

printf("\n %c = MakeNode(%c,%c,%c)",symbols[k],s[index],s[index-1],s[index+1]);

i=index-1;

s[i++]=symbols[k];

for(j=index+2; j<strlen(s);j++)

s[i++]=s[j];

s[i]='\0';

if(k==4)

k=0;

else

k++;

return index-1;

}

void main()

{

int i;

clrscr();

printf("\n Implementation of DAG Representation of Infix Expression");

printf("\n Enter Infix Arithmetic Expression:\n");

scanf("%s",s);

printf("\n DAG Representation:\n");

for(i=0;i<strlen(s);i++)

if(s[i]=='\*' || s[i]=='/')

i=SubExpression(i);

for(i=0;i<strlen(s);i++)

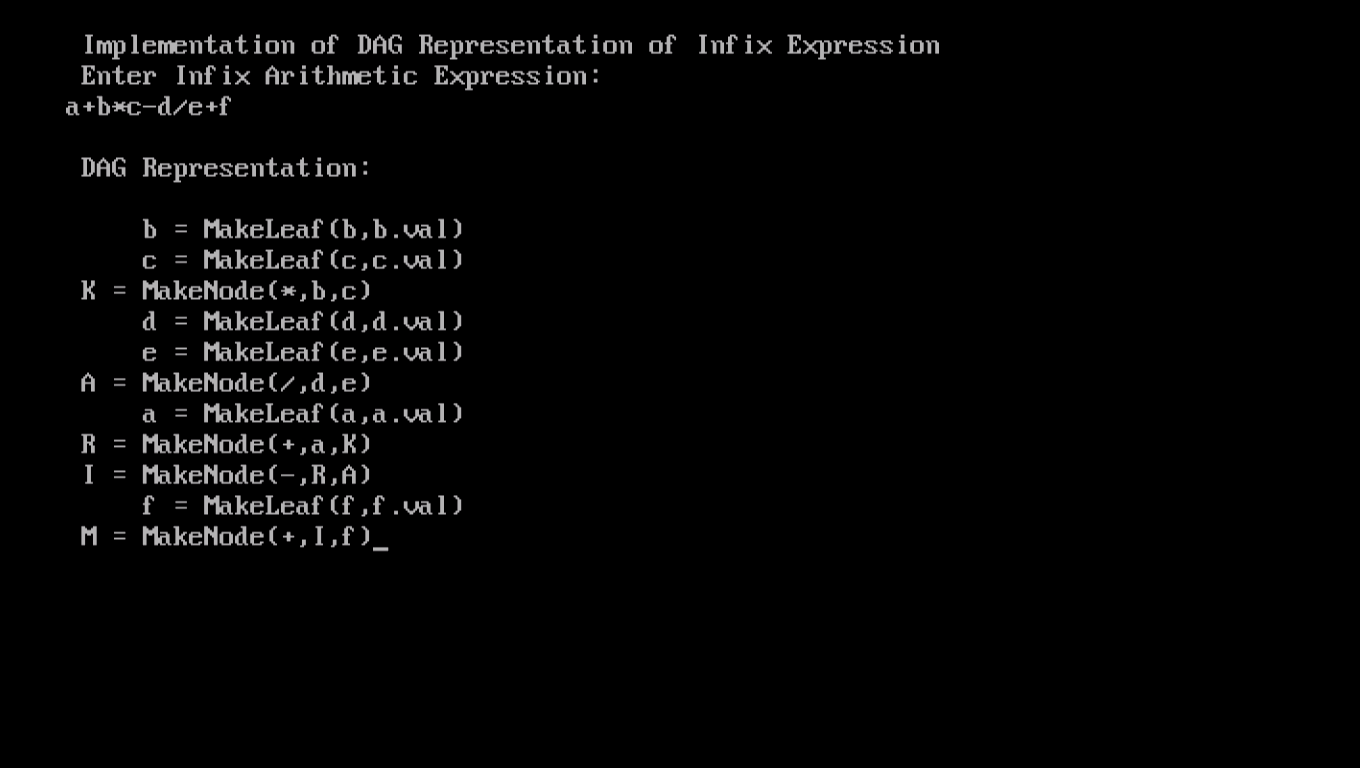
if(s[i]=='+' || s[i]=='-')

i=SubExpression(i);

getch();

}

Output :



Program - 7

**Implementation of Quadruple representation of Infix Expression.**

# include <stdio.h>

# include <conio.h>

struct Quadruple

{

char op;

char arg1;

char arg2;

char result;

}q[20];

int n=-1;

char s[20];

char symbols[5]={'K','A','R','I','M'};

int k=0;

int SubExpression(int index)

{

int i,j;

n++;

q[n].op=s[index];

q[n].arg1=s[index-1];

q[n].arg2=s[index+1];

q[n].result=symbols[k];

i=index-1;

s[i++]=symbols[k];

for(j=index+2; j<strlen(s);j++)

s[i++]=s[j];

s[i]='\0';

if(k==4)

k=0;

else

k++;

return index-1;

}

void main()

{

int i;

clrscr();

printf("\n Implementation of Quadruple Representation of Infix Expression");

printf("\n Enter Infix Arithmetic Expression:\n");

scanf("%s",s);

for(i=0;i<strlen(s);i++)

if(s[i]=='\*' || s[i]=='/')

i=SubExpression(i);

for(i=0;i<strlen(s);i++)

if(s[i]=='+' || s[i]=='-')

i=SubExpression(i);

printf("\n Quadruple Representation of Infix Expression");

printf("\n Operator Arg1 Arg2 Result");

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

printf("\n %c %c %c %c",q[i].op,q[i].arg1,q[i].arg2,q[i].result);

getch();

}

