**PRACTICAL-4**

**AIM : WRITE A PROGRAM TO IMPLEMENT RECURSIVE DECENT PARSING IN C.**

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

char ip\_sym[15],ip\_ptr=0,op[50],tmp[50];

void e\_prime();

void e();

void t\_prime();

void t();

void f();

void advance();

int n=0;

void e()

{

strcpy(op,"TE'");

printf("E=%-25s",op);

printf("E->TE'\n");

t();

e\_prime();

}

void e\_prime()

{

int i,n=0,l;

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

if(op[i]!='e')

tmp[n++]=op[i];

strcpy(op,tmp);

l=strlen(op);

for(n=0;n<l&&op[n]!='E';n++);

if(ip\_sym[ip\_ptr]=='+')

{

i=n+2;

do

{

op[i+2]=op[i];

i++;

}while(i<=l);

op[n++]='+';

op[n++]='T';

op[n++]='E';

op[n++]=39;

printf("E=%-25s",op);

printf("E'->+TE'\n");

advance();

t();

e\_prime();

}

else

{

op[n]='e';

for(i=n+1;i<=strlen(op);i++)

op[i]=op[i+1];

printf("E=%-25s",op);

printf("E'->e");

}

}

void t()

{

int i,n=0,l;

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

if(op[i]!='e')

tmp[n++]=op[i];

strcpy(op,tmp);

l=strlen(op);

for(n=0;n<l&&op[n]!='T';n++);

i=n+1;

do

{

op[i+2]=op[i];

i++;

}while(i<l);

op[n++]='F';

op[n++]='T';

op[n++]=39;

printf("E=%-25s",op);

printf("T->FT'\n");

f();

t\_prime();

}

void t\_prime()

{

int i,n=0,l;

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

if(op[i]!='e')

tmp[n++]=op[i];

strcpy(op,tmp);

l=strlen(op);

for(n=0;n<l&&op[n]!='T';n++);

if(ip\_sym[ip\_ptr]=='\*')

{

i=n+2;

do

{

op[i+2]=op[i];

i++;

}while(i<l);

op[n++]='\*';

op[n++]='F';

op[n++]='T';

op[n++]=39;

printf("E=%-25s",op);

printf("T'->\*FT'\n");

advance();

f();

t\_prime();

}

else

{

op[n]='e';

for(i=n+1;i<=strlen(op);i++)

op[i]=op[i+1];

printf("E=%-25s",op);

printf("T'->e\n");

}

}

void f()

{

int i,n=0,l;

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

if(op[i]!='e')

tmp[n++]=op[i];

strcpy(op,tmp);

l=strlen(op);

for(n=0;n<l&&op[n]!='F';n++);

if((ip\_sym[ip\_ptr]=='i')||(ip\_sym[ip\_ptr]=='I'))

{

op[n]='i';

printf("E=%-25s",op);

printf("F->i\n");

advance();

}

else

{

if(ip\_sym[ip\_ptr]=='(')

{

advance();

e();

if(ip\_sym[ip\_ptr]==')')

{

advance();

i=n+2;

do

{

op[i+2]=op[i];

i++;

}while(i<=l);

op[n++]='(';

op[n++]='E';

op[n++]=')';

printf("E=%-25s",op);

printf("F->(E)\n");

}

}

else

{

printf("\n\t syntax error");

getch();

exit(1);

}

}

}

void advance()

{

ip\_ptr++;

}

void main()

{

int i;

clrscr();

printf("\nGrammar without left recursion");

printf("\n\t\t E->TE' \n\t\t E'->+TE|e' \n\t\t T->FT' ");

printf("\n\t\t T'->\*FT|e' \n\t\t F->(E)|i");

printf("\n Enter the input expression:");

gets(ip\_sym);

printf("Expressions");

printf("\t Sequence of production rules\n");

e();

for(i=0;i<strlen(ip\_sym);i++)

{

if(ip\_sym[i]!='+'&&ip\_sym[i]!='\*'&&ip\_sym[i]!='('&&

ip\_sym[i]!=')'&&ip\_sym[i]!='i'&&ip\_sym[i]!='I')

{

printf("\nSyntax error");

break;

}

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

if(op[i]!='e')

tmp[n++]=op[i];

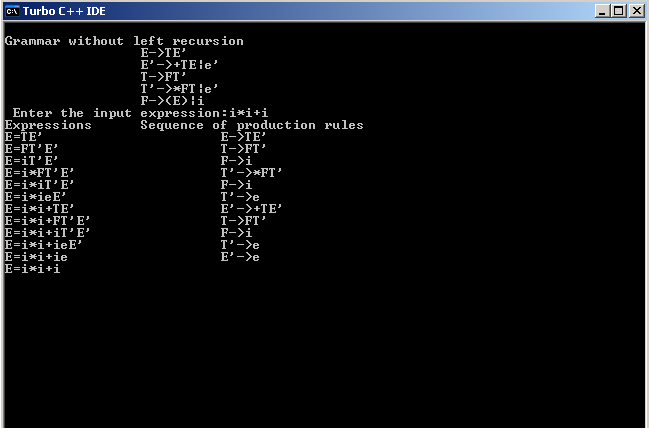
strcpy(op,tmp);

printf("\nE=%-25s",op);

}

getch();

}

**OUTPUT:**

**PRACTICAL-5**

**AIM : WRITE A PROGRAM TO IMPLEMENT PREDICITVE PARSING IN C.**

**PROGRAM:**

#include<string.h>

#include<conio.h>

char a[10];

int top=-1,i;

void error(){

printf("Syntax Error");

}

void push(char k[]) //Pushes The Set Of Characters on to the Stack

{

for(i=0;k[i]!='\0';i++)

{

if(top<9)

a[++top]=k[i];

}

}

char TOS() //Returns TOP of the Stack

{

return a[top];

}

void pop() //Pops 1 element from the Stack

{

if(top>=0)

a[top--]='\0';

}

void display() //Displays Elements Of Stack

{

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

printf("%c",a[i]);

}

void display1(char p[],int m) //Displays The Present Input String

{

int l;

printf("\t");

for(l=m;p[l]!='\0';l++)

printf("%c",p[l]);

}

char\* stack(){

return a;

}

void main()

{

char ip[20],r[20],st,an;

int ir,ic,j=0,k;

char t[5][6][10]={"$","$","TH","$","TH","$",

"+TH","$","e","e","$","e",

"$","$","FU","$","FU","$",

"e","\*FU","e","e","$","e",

"$","$","(E)","$","i","$"};

clrscr();

printf("\nEnter any String(Append with $)");

gets(ip);

printf("Stack\tInput\tOutput\n\n");

push("$E");

display();

printf("\t%s\n",ip);

for(j=0;ip[j]!='\0';)

{

if(TOS()==an)

{

pop();

display();

display1(ip,j+1);

printf("\tPOP\n");

j++;

}

an=ip[j];

st=TOS();

if(st=='E')ir=0;

else if(st=='H')ir=1;

else if(st=='T')ir=2;

else if(st=='U')ir=3;

else if(st=='F')ir=4;

else {

error();

break;

}

if(an=='+')ic=0;

else if(an=='\*')ic=1;

else if(an=='(')ic=2;

else if(an==')')ic=3;

else if((an>='a'&&an<='z')||(an>='A'&&an<='Z')){ic=4;an='i';}

else if(an=='$')ic=5;

strcpy(r,strrev(t[ir][ic]));

strrev(t[ir][ic]);

pop();

push(r);

if(TOS()=='e')

{

pop();

display();

display1(ip,j);

printf("\t%c->%c\n",st,238);

}

else{

display();

display1(ip,j);

printf("\t%c->%s\n",st,t[ir][ic]);

}

if(TOS()=='$'&&an=='$')

break;

if(TOS()=='$'){

error();

break;

}

}

k=strcmp(stack(),"$");

if(k==0 && i==strlen(ip))

printf("\n Given String is not accepted");

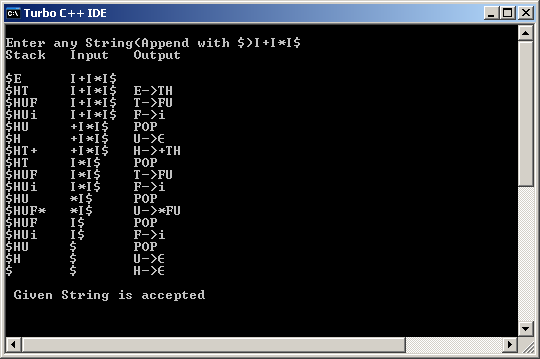
else

printf("\n Given String is accepted");

getch();

}

**OUTPUT:**



**Practical - 6**

**AIM : PROGRAM TO IMPLEMENT OPERATOR PRECEDENCE PARSING**

**PROGRAM:**

#include <iostream.h>

#include <conio.h>

#include <stdlib.h>

struct node

{

char symbol;

struct node \*left;

struct node \*right;

};

typedef struct node node;

struct stack

{

char op;

node \*op\_pointer;

};

typedef struct stack stack;

int TOS=-1;

stack s[10];

/\* This is the operator precedence table stored as it is in the

form of matrix and operators are assigned values as follows:

NULL = 0, Equal (=) = 1, less than (<) = 2, greater than (>) = 3. \*/

// +,\*,(,),<>,-,/,^

int m[8][8]={ {3,2,2,3,3,3,2,2}, //+

{3,3,2,3,3,3,2,2}, //\*

{2,2,2,1,0,2,2,2}, //(

{3,3,0,3,3,3,3,3}, //)

{2,2,2,0,1,2,2,2}, //><

{3,2,2,3,3,3,2,2}, //-

{3,3,3,3,3,3,3,2}, ///

{3,3,3,3,3,3,3,2}}; //^

void main()

{

clrscr();

int comp(char );

void push(char);

void pop();

void display(node \*);

node \*temp;

char str[20];

int i=0;

cout << "Enter the string : ";

cin >> str;

push('<');

i++;

while(str[i]!='\0')

{

if((str[i]>='a'&&str[i]<='z'))

{

temp = new node;

temp->symbol=str[i];

temp->left=NULL;

temp->right=NULL;

s[TOS].op\_pointer = temp;

}

else

{

int p\_index,q\_index;

p\_index = comp(s[TOS].op);

q\_index = comp(str[i]);

while(m[p\_index][q\_index]==3)

{

temp = new node;

temp->symbol=s[TOS].op;

temp->left=s[TOS-1].op\_pointer;

temp->right=s[TOS].op\_pointer;

pop();

s[TOS].op\_pointer=temp;

p\_index = comp(s[TOS].op);

q\_index = comp(str[i]);

}

if(m[p\_index][q\_index]==2)

{

push(str[i]);

}

if(m[p\_index][q\_index]==1)

{

if(comp(s[TOS].op)==4)

break;

if(comp(s[TOS].op)==2)

{

temp=s[TOS].op\_pointer;

pop();

s[TOS].op\_pointer=temp;

}

}

if(m[p\_index][q\_index]==0)

{

cout << "Invalid string.";

getch();

exit(1);

}

}

i++;

}

display(s[TOS].op\_pointer);

getch();

}

// function to get the index of the operator that comes in the TOS

// and also for the current operator that come in the string.

int comp(char x)

{

int y;

switch(x)

{

case '+':

y=0;

break;

case '\*':

y=1;

break;

case '(':

y=2;

break;

case ')':

y=3;

break;

case '>':

case '<':

y=4;

break;

case '-':

y=5;

break;

case '/':

y=6;

break;

case '^':

y=7;

break;

}

return y;

}

void push(char x)

{

TOS++;

s[TOS].op=x;

}

void pop()

{

TOS--;

}

void display(node \*s)

{

if(s==NULL)

return;

if(s->left!=NULL)

display(s->left);

if(s->right!=NULL)

display(s->right);

cout << s->symbol;

}

**OUTPUT:**



**PRACTICAL-7**

**AIM : WRITE A PROGRAM TO IMPLEMENT SHIFT REDUCE PARSING.**

**PROGRAM:**

#include<stdio.h>

#include<iostream.h>

#include<ctype.h>

#include<string.h>

#include<conio.h>

struct stru1

{

char non\_ter[1],pro[25];

}cfg[25];

int n,st=-1,j,i,t=-1,m;

int v,c,p=1;

char str[20],stack[20],ch,tmp[10];

void match(int k);

void matchl(int k);

void main()

{

clrscr();

cprintf("Enter the number of productions:\n\r");

cscanf("%d",&n);

cprintf("\n\r");

cprintf("Enter the productions on LEFT and RIGHT sides:\n\r");

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

{

cscanf("%s",cfg[i].non\_ter);

cprintf("\n\r");

cprintf("->\n\r");

cscanf("%s",cfg[i].pro);

cprintf("\n\r");

}

cprintf("Enter the input string:\n\r");

cscanf("%s",str);

cprintf("\n\r");

i=0;

do

{

ch=str[i];

stack[++st]=ch;

tmp[0]=ch;

match(1);

i++;

}while(str[i]!='\0');

c=st;

v=st;

cputs(stack);

cprintf("\n\r");

while(st!=0)

{

v=--st;

t=-1;

p=0;

while(v<=c)

{

tmp[++t]=stack[v++];

p++;

}

matchl(p);

}

cfg[0].non\_ter[1]='\0';

if(strcmp(stack,cfg[0].non\_ter)==0)

cprintf("String is present in Grammar G\n\r");

else

cprintf("String is not present in Grammar G\n\r");

}

void match(int k)

{

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

{

if(strlen(cfg[j].pro)==k)

{

if(strcmp(tmp,cfg[j].pro)==0)

{

stack[st]=cfg[j].non\_ter[0];

break;

}

}

}

}

void matchl(int k)

{

int x=1,y;

y=k-1;

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

{

if(strlen(cfg[j].pro)==k)

{

if(strcmp(tmp,cfg[j].pro)==0)

{

k=c-k+1;

stack[k]=cfg[j].non\_ter[0];

do

{

stack[k+x]='\0';

tmp[t--]='\0';

c--;

x++;

}while(x<=y);

tmp[t]='\0';

cputs(stack);

cprintf("\n\r");

break;

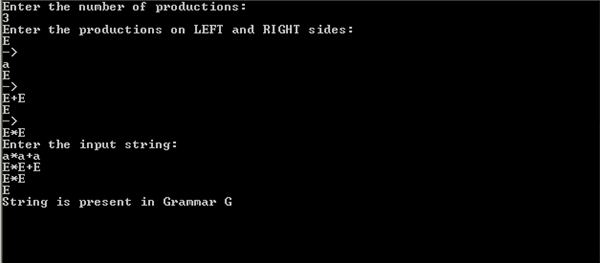
}

}

}

}

**OUTPUT:**



**PRACTICAL-8**

**AIM : WRITE A PROGRAM TO IMPLEMENT SLR PARSING.**

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

int axn[][6][2]={

{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

{{-1,-1},{100,6},{-1,-1},{-1,-1},{-1,-1},{102,102}},

{{-1,-1},{101,2},{100,7},{-1,-1},{101,2},{101,2}},

{{-1,-1},{101,4},{101,4},{-1,-1},{101,4},{101,4}},

{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

{{-1,-1},{101,6},{101,6},{-1,-1},{101,6},{101,6}},

{{100,5},{-1,-1},{-1,-1},{-1,-1},{-1,-1},{-1,-1}},

{{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

{{-1,-1},{100,6},{-1,-1},{-1,-1},{100,11},{-1,-1}},

{{-1,-1},{101,1},{100,7},{-1,-1},{101,1},{101,1}},

{{-1,-1},{101,3},{101,3},{-1,-1},{101,3},{101,3}},

{{-1,-1},{101,5},{101,5},{-1,-1},{101,5},{101,5}}

};//Axn Table

int gotot[12][3]={1,2,3,-1,-1,-1,-1,-1,-1,-1,-1,-1,8,2,3,-1,-1,-1,

-1,9,3,-1,-1,10,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1}; //GoTo table

int a[10];

char b[10];

int top=-1,btop=-1,i;

void push(int k)

{

if(top<9)

a[++top]=k;

}

void pushb(char k)

{

if(btop<9)

b[++btop]=k;

}

char TOS()

{

return a[top];

}

void pop()

{

if(top>=0)

top--;

}

void popb()

{

if(btop>=0)

b[btop--]='\0';

}

void display()

{

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

printf("%d%c",a[i],b[i]);

}

void display1(char p[],int m) //Displays The Present Input String

{

int l;

printf("\t\t");

for(l=m;p[l]!='\0';l++)

printf("%c",p[l]);

printf("\n");

}

void error()

{

printf("Syntax Error");

}

void reduce(int p)

{

int len,k,ad;

char src,\*dest;

switch(p)

{

case 1:dest="E+T";

src='E';

break;

case 2:dest="T";

src='E';

break;

case 3:dest="T\*F";

src='T';

break;

case 4:dest="F";

src='T';

break;

case 5:dest="(E)";

src='F';

break;

case 6:dest="i";

src='F';

break;

default:dest="\0";

src='\0';

break;

}

for(k=0;k<strlen(dest);k++)

{

pop();

popb();

}

pushb(src);

switch(src)

{

case'E':ad=0;

break;

case'T':ad=1;

break;

case'F':ad=2;

break;

default: ad=-1;

break;

}

push(gotot[TOS()][ad]);

}

int main()

{

int j,st,ic;

char ip[20]="\0",an;

clrscr();

printf("Enter any String");

gets(ip);

push(0);

display();

printf("\t%s\n",ip);

for(j=0;ip[j]!='\0';)

{

st=TOS();

an=ip[j];

if(an>='a'&&an<='z') ic=0;

elseif(an=='+') ic=1;

elseif(an=='\*') ic=2;

elseif(an=='(') ic=3;

elseif(an==')') ic=4;

elseif(an=='$') ic=5;

else {

error();

break;

}

if(axn[st][ic][0]==100)

{

pushb(an);

push(axn[st][ic][1]);

display();

j++;

display1(ip,j);

}

if(axn[st][ic][0]==101)

{

reduce(axn[st][ic][1]);

display();

display1(ip,j);

}

if(axn[st][ic][1]==102)

{

printf("Given String is accepted");

break;

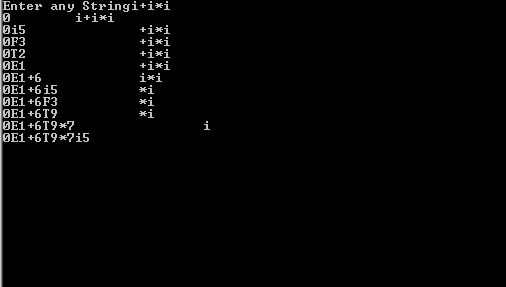
}

}

return 0;

}

**OUTPUT:**



**PRACTICAL: 9**

**AIM: WRITE A C PROGRAM TO IMPLEMENT CLR PARSING.**

**PROGRAM:**

#include<iostream.h>

#include <stdio.h>

#include<conio.h>

#include<string.h>

#include<ctype.h>

#include<process.h>

char \*production[6][5] = { {"","I","+","\*","#"},

{"E","TF","X","X","X"} ,

{"F","X","+TF","X","\0"},

{"T","VU","X","X","X"},

{"U","X","\0","\*VU","\0"},

{"V","I","X","X","X"}

};

char sstr[50],id[50],csf[50]={"E"},temp[50],t1[50],t2[50],str[50],s1[15];

int i=0,d=0,j=0,k=0,p=0,s=0,u=0,v=0,l;

void predict(char ch1,char ch2)

{

char \*tmp1,\*tmp2;

tmp1=new char(10);

tmp2=new char(10);

tmp1[0] = ch1;

tmp1[1] = '\0';

tmp2[0] = ch2;

tmp2[1] = '\0';

for(u=1;u<=6;u++)

{

if(strcmp(tmp1,production[u][0])==0)

{

for(v=1;v<=4;v++)

{

if(strcmp(tmp2,production[0][v])==0)

{

strcpy(temp,production[u][v]);

return;

}

}

}

}

}

void eval(char ch1, char ch2)

{

if(isalpha(ch2))

ch2 = 'I';

predict(ch1,ch2);

if(strcmp(temp,"X")==0)

{

cout<<"ERROR!!CANNOT PARSE FURTHER"<<endl;

getch();

exit(1);

return;

}

else if(strcmp(temp,"I")==0)

{

s = 0;

while(id[s]!='\0')

{

if(id[s]==sstr[i])

{

csf[i]=id[s];

i=0;

//k++;

break;

}

s++;

}

return;

}

else if(temp[0] == '\0')

{

p = i;

while(csf[p]!='\0')

{

csf[p]=csf[p+1];

p++;

}

i = 0;

return;

}

p = i;

while(csf[p]!='\0')

{

csf[p]=csf[p+1];

p++;

}

if(i>=1)

{

j = 0;

while(j<i)

t1[j] = csf[j++];

t1[j] = '\0';

j = 0;

k = i ;

while(csf[k]!='\0')

t2[j++]=csf[k++];

t2[j] = '\0';

j = 0;

while(t1[j]!='\0')

csf[j]=t1[j++];

k = 0;

while(temp[k]!='\0')

csf[j++]=temp[k++];

k = 0;

while(t2[k]!='\0')

csf[j++]=t2[k++];

}

else

{

strcat(temp,csf);

strcpy(csf,temp);

}

}

void main()

{

clrscr();

char ch;

cout<<"ENTER THE STRING WHICH U WANT TO PARSE(END WITH #) : ";

gets(sstr);

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

{

if(sstr[i] == ' ')

{

for(j=i;j<strlen(sstr)-1;j++)

sstr[j] = sstr[j+1];

sstr[j] = '\0';

}

}

i=0;

while(sstr[i]!='#')

{

if (isalpha(sstr[i]))

{

id[d++]=sstr[i];

}

i++;

id[d]='\0';

}

i = 0;

while(sstr[i]!='#')

str[i]=sstr[i++];

str[i]='\0';

i = 0;

cout<<"------------------------------------------------------"<<endl;

cout<<"CSF \t\t SYMBOL\t\tPRED "<<endl;

cout<<"------------------------------------------------------"<<endl;

cout<<"E";

while(strcmp(csf,str)!=0)

{

if(csf[i]==sstr[i])

{

i++;

continue;

}

ch = sstr[i];

eval(csf[i],sstr[i]);

cout<<"\t \t"<<ch<<"\t\t"<<temp<<endl;

cout<<csf;

}

getch();

}

**OUTPUT:**

