**UNIVERSITY INSTITUTE OF ENGINEERING**

**AND TECHNOLOGY**

**PANJAB UNIVERSITY, CHANDIGARH**



**COMPILER DESIGN**

**CS-654**

**SUBMITTED TO: SUBMITTED BY:**

AKASHDEEP SHARMA BALKISHA

CSE SEC-1

UE 143023

GROUP 2

**INDEX**

1. Program for Lexical Analyzer……………..……………3-21

**AIM**: Program for Lexical Analyzer

**PROGRAM**:

#include<iostream>

#include<conio.h>

#include<cstring>

#include<fstream>

#include<cstring>

#include<stdio.h>

using namespace std;

int isdelimiter(char ch)

{

if(ch==';'||ch=='('||ch==')'||ch=='\r'||ch=='{'||ch=='}'||ch=='['||ch==']'||ch==' '||ch=='\n'||ch=='\t'||ch==';'||ch==':'||ch==','||ch=='#')

return 1;

else

return 0;

}

int isoperator(char ch)

{

if(ch=='+'||ch=='-'||ch=='\*'||ch=='/'||ch=='='|| ch=='&'|| ch=='|' || ch=='^' || ch=='<' || ch=='>'

|| ch=='>'|| ch=='!'|| ch=='%' || ch=='~')

return 1;

else

return 0;

}

int main()

{

ifstream file;

char ch;

file.open("input.c",ios::in);

int fid=0,finvalid=0,fhex=0,foct=0,fbin=0,ffloat=0,fint=0,fcom=0,fpre=0,i=0,j=0,he=0,be=0,line=0,err[100];

char numbers[100][100],invalid[100][100],ident[100][100],keyword[100][100],

operators[100][10],strings[1000][100],preprocessor[1000][100],filename[1000][100];

char arr[1000],integer[100][100],floatnum[100][100],hexnum[100][100],binnum[100][100],octnum[100][100],punct[100][2];

int carr=0,cint=0,cfloat=0,dot=0,chex=0,cbin=0,coct=0,cop=0,cstr=0,cpunct=0,cinvalid=0,cid=0,cpre=0,cfile=0;

char sKeywords[64][20] = {

"asm","auto","bool","break","case","catch","char","class","const","const\_cast",

"continue","default","delete","do","double","dynamic\_cast","else","enum","explicit",

"export","extern","false","float","for","friend","goto","if","inline","int","long",

"main","mutable","namespace","new","operator","private","protected","public",

"register","reinterpret\_cast","return","short","signed","sizeof","static",

"static\_cast","struct","switch","template","this","throw","true","try","typedef",

"typeid","typename","union","unsigned","using","virtual","void","volatile","printf","while",

};

while(!file.eof())

{

carr=0;

start:

finvalid=0; fid=0; fint=0,ffloat=0,fbin=0,fcom=0,fpre=0;

file.get(ch);

if(!isoperator(ch)&& ! isdelimiter(ch) && !((ch>=97 && ch<=122)|| (ch>=65 && ch<=90) ||(ch>=48 && ch<=57) || (ch==95) || (ch=='"')) )

{

finvalid=1;

goto invalid;

}

if((ch>=97 && ch<=122) || (ch>=65 && ch<=90) || ch=='\_')

{

fid=1;

goto id;

}

if((ch>=49 && ch<=57))

{

fint=1;

goto intlabel;

}

if(ch==48)

{

arr[carr++]=ch;

file.get(ch);

if(isoperator(ch)|| isdelimiter(ch))

{

strcpy(integer[cint++],"0");

cout<<arr<<" - integer"<<endl;

if(!file.eof())

file.unget();

}

else{

finvalid=1;

goto invalid;

}

}

if(ch=='/') //Single line comments

{

file.get(ch);

if(ch=='/')

{

while(ch!='\n')

{

file.get(ch);

if(file.eof())

break;

}

}

if(ch=='\*')

{

fcom=1;

goto multiline; // multi line comments

}

}

multiline:

if(fcom==1)

{

int flag=0;

while(ch!='/')

{

file.get(ch);

if(file.eof())

{

flag=1;

break;

}

}

if(flag!=1) {

file.unget();

file.unget();

file.get(ch);

if(ch!='\*')

{

fcom=0;

goto invalid;

}

file.get(ch);

file.get(ch);}

}

intlabel:

if(fint==1)

{

while(!isoperator(ch) && !isdelimiter(ch))

{

if((ch>=48 && ch<=57))

{

arr[carr++]=ch;

file.get(ch);

if(file.eof())

break;

else

{

finvalid=1; fint=0;

goto invalid;

}

}

if(!file.eof())

file.unget();

strcpy(integer[cint++],arr);

cout<<arr<<" - integer"<<endl;

carr=0;

memset(arr,'\0',100);

}}

if(ch=='#')//--------------header files---------

{

fpre=1;

goto Pre;

}

Pre: if(fpre==1)

{

while(ch!='<')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(preprocessor[cpre++],arr);

cout<<arr<<" - preprocessor directive"<<endl;

carr=0;

memset(arr,'\0',100);

file.get(ch);

while(ch!='>')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(filename[cfile++],arr);

cout<<arr<<" - file name"<<endl;

carr=0;

memset(arr,'\0',100);

file.get(ch);

}

if(isoperator(ch))

{

if(ch=='+')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='+' || ch=='=')

{

arr[carr++]=ch;

file.get(ch);

}

carr=0;

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

memset(arr,'\0',100);

if(ch=='"')

goto stringlabel;

if(!file.eof())

file.unget();

}

else if(ch=='=')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='=')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

carr=0;

memset(arr,'\0',100);

if(ch=='"')

{

goto stringlabel;

}

if(!file.eof())

file.unget();

}

else if(ch=='-')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='-' || ch=='=')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

carr=0;

memset(arr,'\0',100);

if(ch=='"')

goto stringlabel;

if(!file.eof())

file.unget();

}

else if(ch=='\*')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='\*' || ch=='=')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

carr=0;

memset(arr,'\0',100);

if(ch=='"')

goto stringlabel;

if(!file.eof())

file.unget();

}

else if(ch=='&')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='&')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

carr=0;

memset(arr,'\0',100);

if(ch=='"')

goto stringlabel;

if(!file.eof())

file.unget();

}

else if(ch=='|')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='|')

{

arr[carr++]=ch;

file.get(ch);

}

if(!file.eof())

file.unget();

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

carr=0;

memset(arr,'\0',100);

if(ch=='"')

{

file.unget();

goto start;

}

}

else if(ch=='/')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='=')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

carr=0;

memset(arr,'\0',100);

if(ch=='"')

goto stringlabel;

if(!file.eof())

file.unget();

}

else if(ch=='%')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='=')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

carr=0;

memset(arr,'\0',100);

if(ch=='"')

goto stringlabel;

if(!file.eof())

file.unget();

}

else if(ch=='<')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='=' || ch=='<')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

carr=0;

memset(arr,'\0',100);

if(ch=='"')

goto stringlabel;

if(!file.eof())

file.unget();

}

else if(ch=='>')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='=' || ch=='>')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

carr=0;

memset(arr,'\0',100);

if(ch=='"')

goto stringlabel;

if(!file.eof())

file.unget();

}

else if(ch=='!')

{

arr[carr++]=ch;

file.get(ch);

if(ch=='=')

{

arr[carr++]=ch;

file.get(ch);

}

strcpy(operators[cop++],arr);

cout<<arr<<" - operator"<<endl;

carr=0;

memset(arr,'\0',100);

if(ch=='"')

goto stringlabel;

if(!file.eof())

file.unget();

}

}

//-----------------------check dilimiter------------------------//

if(isdelimiter(ch))

{

if(ch!='\n'&& ch!='\w'&&ch!='\t'&&ch!=' ' && ch!='\r')

{

cout<<ch<<" - punctuater"<<endl;

punct[cpunct++][0]=ch;

}

if(ch=='\n')

line++;

}

//----------------------All the labels used--------------------//

stringlabel: if(ch=='"')

{

file.get(ch);

int st=0;

while(ch!='"')

{

arr[carr++]=ch;

file.get(ch);

if(file.eof())

{

strcpy(invalid[cinvalid++],arr);

st=1;

break;

}

}

if(st==0)

{

strcpy(strings[cstr++],arr);

cout<<arr<<" - string literal"<<endl;

carr=0;

memset(arr,'\0',100);

}

}

id:

if(fid==1)

{

while(!isdelimiter(ch) && !isoperator(ch) && file.tellg()!=-1)

{

if((ch>=97 && ch<=122) || (ch>=65 && ch<=90) || (ch>=48 && ch<=57) || (ch==95))

{

arr[carr++]=ch;

file.get(ch);

}

else

{

finvalid=1;

goto invalid;

}

}

if(!file.eof())

file.unget();

strcpy(ident[cid++],arr);

int key=0;

for(int j=0;j<64;j++)

{

if(strcmp(arr,sKeywords[j])==0)

{

strcpy(keyword[key++],arr);

cout<<arr<<" - keyword"<<endl;

key=1;

}

}

if(key==0)

cout<<arr<<" - identifier"<<endl;

carr=0;

memset(arr,'\0',100);

}

invalid:

if(finvalid==1)

{

while(!isdelimiter(ch) && !isoperator(ch))

{

arr[carr++]=ch;

file.get(ch);

if(file.tellg()==-1)

break;

}

if(!file.eof())

file.unget();

err[cinvalid]=line+1;

strcpy(invalid[cinvalid++],arr);

cout<<"line no. "<<line+1<<" "<<arr<<" - invalid"<<endl;

carr=0;

memset(arr,'\0',100);

}

}

file.close();

//cout<<"\n Bye";

return 0;

}

**INPUT FILE:**

#include<stdio.h>

#include<conio.h>

using namespace std;

int main()

{

/\*using namespace\*/

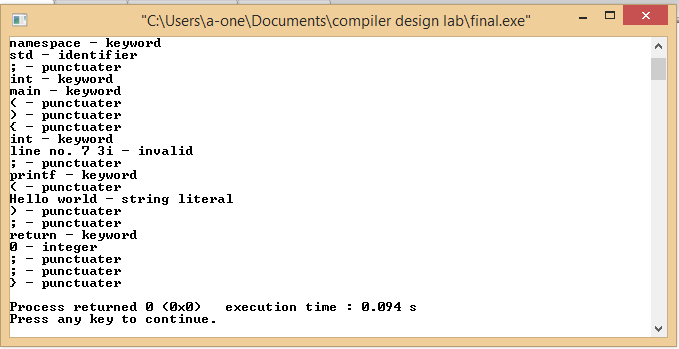
int 3i;

printf("Hello world");

return 0;

}

**OUTPUT:**



**AIM**: Program to implement regular expression to NFA

PROGRAM:

#include<iostream>

#include<stdio.h>

#include<conio.h>

#include<string.h>

using namespace std;

char reg[10];

char nfa[10][10];

int i,j,k,states,r=0;

void display(int n)

{

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

cout<<' '<<j;

cout<<"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

cout<<"\n";

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

{

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

{ cout<<nfa[i][j]<<' ';}

cout<<" | "<<i;

cout<<"\n";

}

cout<<"\nStart state: 0\nFinal state:"<<i-1;

}

int main()

{

for( i=0;i<10;i++)

{

for( j=0;j<10;j++)

{

nfa[i][j]=' ';

}

}

cout<<"enter the regular expression\n";

gets(reg);

//ex inputs 1. (a\*) 2.(a|b) 3.(a.b) 4. a 5.b

int len=strlen(reg);

cout<<len<<endl;

if(len==1)

{

if(reg[0]=='a')

{

nfa[r][r+1]=reg[0];

}

else if(reg[0]=='b')

{

nfa[r][r+1]=reg[0];

}

r=r+1;

}

else{

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

{switch(reg[i])

{case '|' :

{

nfa[r][r+1]='E';

nfa[r+1][r+2]=reg[i-1];

nfa[r+2][r+5]='E';

nfa[r][r+3]='E';

nfa[r+4][r+5]='E';

nfa[r+3][r+4]=reg[i+1];

r=r+5;//fr final state

break;

}

case '\*':

{nfa[r-1][r]='E';

nfa[r][r+1]='E';

nfa[r][r+3]='E';

nfa[r+1][r+2]=reg[i-1];

nfa[r+2][r+1]='E';

nfa[r+2][r+3]='E';

r=r+3;

break;

}

case '.':

{

nfa[r][r+1]=reg[i-1];

nfa[r+1][r+2]='E';

nfa[r+2][r+3]=reg[i+1];

r=r+3;

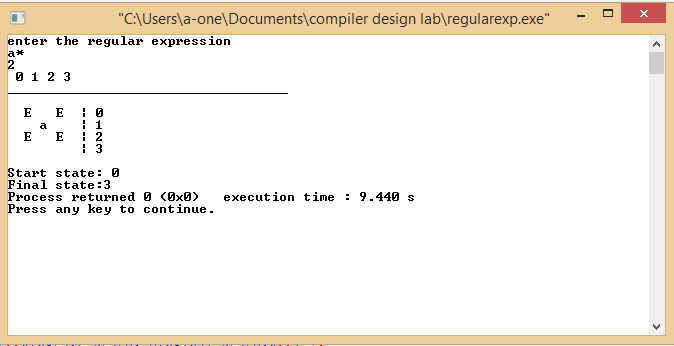
break;

}}}}

display(r);

return 0;}

**OUTPUT:**



**AIM:** Introduction to LEX Tool

Lex is a computer program that generates lexical analyzers’.

Lex reads an input stream specifying the lexical analyses and outputs [source code](https://en.wikipedia.org/wiki/Source_code) implementing the Lexar in the C programming language.

**STRUCTURE OF LEX FILE**

The structure of a Lex file is intentionally similar to that of a yacc file; files are divided into three sections, separated by lines that contain only two percent signs, as follows:

Definition section

%%

Rules section

%%

C code section

The definition section defines macros and imports header files written in C. It is also possible to write any C code here, which will be copied verbatim into the generated source file.

The rules section associates regular expression patterns with C statements. When the lexer sees text in the input matching a given pattern, it will execute the associated C code.

The C code section contains C statements and functions that are copied verbatim to the generated source file. These statements presumably contain code called by the rules in the rules section. In large programs it is more convenient to place this code in a separate file linked in at compile time.

Lex source is a table of regular expressions and corresponding program fragments. The table is translated to a program which reads an input stream, copying it to an output stream and partitioning the input into strings which match the given expressions. The program fragments written by the user are executed in the order in which the corresponding regular expressions occur in the input stream.

The regular expressions are specified by the user in the source specifications given to Lex. The Lex written code recognizes these expressions in an input stream and partitions the input stream into strings matching the expressions.

Lex turns the user's expressions and actions into the host general-purpose language; the generated program is named yylex.

**AIM**: Program to implement lexical analysis using Flex tool

**INPUT FILE:**

#include<stdio.h>

using namepsace std;

void main()

{

int i, j;

j=i+1;

cout<<j;///this is single line comment

cout<<"hello";

}

**PROGRAM:** To remove comments in a program

%option noyywrap

%{

#include<stdio.h>

%}

%%

\/\/.\*;

\/\\*(.\*\n)\*.\*\\*\/;

%%

int main()

{

yyin=fopen("sample.txt","r");

yylex();

return 0;

}

**OUTPUT**

#include<stdio.h>

using namepsace std;

void main()

{

int i, j;

j=i+1;

cout<<j;

cout<<"hello";

}

**PROGRAM:** To add line numbers in a program

%option noyywrap

%{

#include<stdio.h>

int lines=0;

%}

LINE \n

%%

{LINE} {ECHO; printf("%d", ++lines);}

%%

int main()

{

yylex();

return 0;

}

**OUTPUT:**

1#include<stdio.h>

2using namepsace std;

3void main()

4{

5int i, j;

6j=i+1;

7cout<<j;

8cout<<"hello";

9}

**PROGRAM:** To identify keywords and identifiers in program

%option noyywrap

%{

#include<stdio.h>

%}

letter [a-zA-Z]

S []

digit [0-9]+

id {letter}({letter}|{digit}|\_)\*

%%

("main")|("int")|("scanf")|("printf")|("if")|("else")|("float")|("else")|("while")|("for")|("char") {printf("%s is a keyword\n",yytext);}

"+"|"-"|"\*"|"/"|"="|"=="|"<"|"<="|">"|">="

"!="|"&&"|"||"| {printf("%s is a operator\n",yytext);}

"!"|";"|"["|"]"|"{"|"}"|"["|"]" {printf("%s is punctuator\n",yytext);}

" "{id} {printf("%s is an identifier\n",yytext);}

{digit} {printf("%s is a number\n",yytext);}

. ;

%%

int main()

{

yyin=fopen("output.txt","r");

yylex();

return 0;

}

**OUTPUT:**

< is a operator

> is a operator

namepsace is an identifier

std is an identifier

main is an identifier

int is a keyword

i is an identifier

j is an identifier

= is a operator

+ is a operator

1 is a number

< is a operator

< is a operator

< is a operator

< is a operator

**AIM:** Program to implement left recursion and left factoring on grammar

**PROGRAM:**

#include<vector>

#include<iostream>

#include<string>

using namespace std;

int main()

{

char lhs;

string str1;

string prod1,prod2;

string temp1;

string temppr1,temppr2;

vector <string> alpha;

vector <string> beta;

vector <string> parts;

vector <string> parts2;

cout<<"enter production \n";

cin>>str1;

lhs=str1[0];

str1=str1.substr(3);

str1+='\0';

for(int i=0;i<str1.length();i++)

{

if(str1[i]!='|' && str1[i]!='\0')

{

temp1+=str1[i];

}

else

{if(temp1[0]==lhs)

{alpha.push\_back(temp1.substr(1));

temp1.erase();

}else

{beta.push\_back(temp1);

temp1.erase();}}}

if(alpha.size()==0)

{cout<<"Grammar is not left-recursive";}

else{

cout<<"the grammar without left recursion"<<endl;

cout<<lhs<<"->";

prod1+=lhs;

prod1+='-';

prod1+='>';

int i;

for(i = 0; i <(beta.size()-1); i++)

{ cout<<beta[i]<<lhs<<"' | ";

prod1+=beta[i];

prod1+=lhs;

prod1+='\'';

prod1+='|';}

cout<<beta[i]<<lhs<<"'"<<endl;

prod1+=beta[i];

prod1+=lhs;

prod1+='\'';

prod1+='\0';

cout<<lhs<<"'"<<"-> ";

prod2+=lhs;

prod2+='\'';

prod2+='-';

prod2+='>';

for(i = 0; i<(alpha.size()); i++)

{ cout<<alpha[i]<<lhs<<"' |";

prod2+=alpha[i];

prod2+=lhs;

prod2+='\'';

prod2+='|';}

cout<<"E"<<endl;

prod2+='E';

prod2+='\0';}

temppr1=prod1.substr(3);

temppr2=prod2.substr(4);

for(int x=0;x<temppr1.length();x++)

{if(temppr1[x]!='|' && temppr1[x]!='\0')

{temp1+=temppr1[x];}

else

{parts.push\_back(temp1);

temp1.erase();}

}

string common,v,p;

v=parts[0];

int len=0;

for(int y=0;y<v.length();y++)

{

common+=v[y];

len=common.length();

for(int z=0;z<parts.size();z++)

{ p=parts[z];

p=p.substr(0,len);

if(p.compare(common) !=0)

{common=common.substr(0,len-1);

break;

} p.clear(); }}

len=common.length();

for(int n=0;n<parts.size();n++)

{ p=parts[n];

p=p.substr(len);

parts[n]=p;

}

cout<<endl;

cout <<prod1[0]<<prod1[1]<<prod1[2]<<common<<"X"<<endl;

cout<<"X ->";

for(int n=0;n<parts.size();n++)

cout<<parts[n]<<"|";

cout<<endl;

temp1.erase();

for(int x=0;x<temppr2.length();x++)

{if(temppr2[x]!='|' && temppr2[x]!='\0')

{temp1+=temppr2[x];

}

else

{parts2.push\_back(temp1);

temp1.erase();}}

string common1,v1,p1;

v1=parts2[0];

int len1=0;

for(int y=0;y<v1.length();y++)

{common1+=v1[y];

len1=common1.length();

for(int z=0;z<parts2.size();z++)

{p1=parts2[z];

p1=p1.substr(0,len1);

if(p1.compare(common1) !=0)

{common1=common1.substr(0,len-1);

break; }

p1.clear(); } }

len1=common1.length();

if(len1!=0)

{ for(int n=0;n<parts2.size();n++)

{ p1=parts2[n];

p1=p1.substr(len);

parts2[n]=p;

}

cout<<endl;

cout <<prod2[0]<<prod2[1]<<prod2[2]<<common1<<"Y"<<endl;

cout<<"Y ->";

for(int n=0;n<parts2.size();n++)

cout<<parts2[n]<<"|";}

else

{

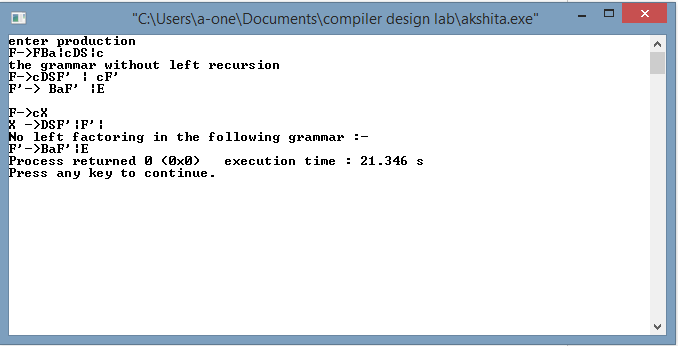
cout<<"No left factoring in the following grammar :-"<<endl<<prod2;

}

return 0;

}

**OUTPUT:**



**AIM:** Program to implement recursive descent parser

**PROGRAM:**

#include<iostream>

using namespace std;

#include<string.h>

#include<ctype.h>

#include<stdio.h>

#include<conio.h>

void E();

int A();

int B();

int C();

string \*point;

int i,len;

string input;

int main()

{

cout<<"Grammar is:"<<endl<<"E->+A|+B|-C"<<endl<<"A->a"<<endl<<"B->b|c"<<endl<<"C->C"<<endl;

i=0;

point=&input;

cout<<"enter any expression\n";

cin>>input;

E();

return 0;

}

void E()

{

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

{

i++;

int p=A();

if(p==0)

{

int m=B();

if(m==0)

cout<<"error occurs";

else

cout<<"match occurs";

}

else{

cout<<"match occurs";}

}

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

{

i++;

int k=C();

if(k==0)

cout<<"error occurs";

else

cout<<"match is there";

}

else

cout<<"error ocuurs";

}

int A()

{

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

{

return 1;

}

else

return 0;

}

int B()

{

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

{

return 1;

}

else

C();

}

int C()

{

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

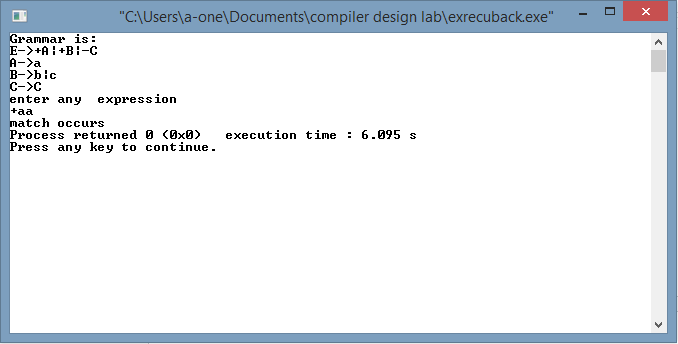
return 1;

else

return 0;

}

**OUTPUT:**



**AIM:** Program to find First and Follow in given set of productions of a grammar

**PROGRAM:**

#include<iostream>

#include<string.h>

#include<stdlib.h>

#include<ctype.h>

#include<algorithm>

using namespace std;

int isdelimiter(char ch)

{

if(ch==';'||ch=='('||ch==')'||ch=='\r'||ch=='{'||ch=='}'||ch=='['||ch==']'||ch==' '||ch=='\n'||ch=='\t'||ch==';'||ch==':'||ch==','||ch=='#')

return 1;

else

return 0;

}

int isoperator(char ch)

{

if(ch=='+'||ch=='-'||ch=='\*'||ch=='/'||ch=='='|| ch=='&'|| ch=='|' || ch=='^' || ch=='<' || ch=='>'

|| ch=='>'|| ch=='!'|| ch=='%' || ch=='~')

return 1;

else

return 0;

}

char productions[100][100];

char Nonterminal[100],terminal[100];

char G[100][100];

char first[100][100],firs[100][100],follow[100][100],foll0[100][100];

int first1(int);

int follow1(int);

void fg();

int flag[100];

int cg=0,ci=0;

int rules;

int check[100];

void addnext(int,int,int);

int Number[100];

int first1(int i)

{ int co=0,index=0;

ci=strlen(first[i]);

int flag=0,count\_call=0;

if(!isupper(G[i][3]))

{ first[i][ci++]=G[i][3];

index=i;

}else if(G[i][3]>=65 && G[i][3]<=90)

{

flag=-1;

int j;

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

{

if(G[j][0]==G[i][3])

{

index=j;

break;

} }

if(check[j]==0)

{ index=first1(j);

flag=1;

check[j]=1;

}

else if(check[j]==1)

flag=1;

if(flag==1)

{ int len=strlen(first[i]);

int len1=strlen(first[index]);

int s1=0;

co=4;

int var=strlen(first[i]);

ci=strlen(first[i]);

for(int s=len; s<len1+len;s++)

{

if(first[index][s1]=='e')

{

if(G[i][co]=='|'||G[i][co]=='\0')

{

first[i][var++]=first[index][s1];

ci++;

}}

else

{ first[i][var++]=first[index][s1];

ci++;}

s1++;}}

for(int k=0;k<strlen(first[index]);k++)

{

if(first[index][k]=='e' )

{

if(G[index][4]!='|')

{

addnext(4,i,ci);

}}}}}

co=4;

int flag1=0;

ci=strlen(first[i]);

while(G[i][co]!='\0')

{

if(G[i][co]=='|')

{

ci=strlen(first[i]);

if((G[i][co+1]>=97 && G[i][co+1]<=122) || isdelimiter(G[i][3])|| isoperator(G[i][3]))

{ first[i][ci++]=G[i][co+1];

index=i;

}

else if(G[i][co+1]>=65 && G[i][co+1]<=90)

{ flag1=-1;

int j;

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

{ if(G[j][0]==G[i][co+1])

{ index=j;

break;}}

if(check[j]==0)

{ index=first1(j);

flag1=1;

check[j]=1;

} else if(check[j]==1)

flag1=1;

if(flag1==1)

{ int len=strlen(first[i]);

int len1=strlen(first[index]);

int s1=0;

int var=strlen(first[i]);

ci=strlen(first[i]);

for(int s=len; s<len1+len;s++)

{ if(first[index][s1]=='e')

{ if(G[i][co+1]=='|'||G[i][co+1]=='\0')

{ first[i][var++]=first[index][s1];

ci++;}}

else

{ first[i][var++]=first[index][s1];

ci++;

}

s1++;}}

for(int k=0;k<strlen(first[index]);k++)

{

if(first[index][k]=='e')

{

co=co+1;

if(G[i][co+1]!='|' && G[i][co+1]!='\0')

{

addnext(co+1,i,ci);

break;

}}}}} co++;

}

return index;

}

void addnext(int c,int i,int ci)

{

if(G[i][c]=='\0' || G[i][c]=='|' || i==cg || c>strlen(G[i]))

{

return;

} int index=0,ind=0;

if((G[i][c]<=122 && G[i][c]>=97) || isdelimiter(G[i][3])|| isoperator(G[i][3]))

{

first[i][ci++]=G[i][c];

return;

}else

{

for(int j=0;j<cg;j++)

{

if(G[j][0]==G[i][c])

{

index=j;

break;

}

}

if(check[index]==0)

{

ind=first1(index);

check[index]=1;

}

if(check[index]==1)

{

int len=strlen(first[i]);

int len1=strlen(first[index]);

int s1=0;

for(int s=len; s<len1+len;s++)

{

first[i][s]=first[index][s1];

ci++; s1++;

}

}

for(int k=0;k<strlen(first[index]);k++)

{

if(first[index][k]=='e')

{

if(i<cg)

{

if((c+1)==strlen(G[i]))

{

cout<<"\n herere";

return;

}

if((c+1)<strlen(G[i]))

{

addnext(c+1,i,ci);

}

else{

return;

}

break;}}}}}}}

int follow1(int i)

{

int co=0,index=0;

ci=strlen(follow[i]);

if(i==0)

{

follow[i][ci++]='$';

}

char ch=G[i][0];

int ci=strlen(follow[i]);

for(int j=0;j<cg;j++)

{

for(int k=3;k<strlen(G[j]);k++)

{

if(ch==G[j][k])

{

if(G[j][k+1]>=65 && G[j][k+1]<=90)

{

label2: int here;

for(int y=0;y<cg;y++)

{

if(G[y][0]==G[j][k+1])

{

here=y;

break;

}

}

for(int l=0;l<strlen(firs[here]);l++)

{

if(firs[here][l]!='e')

follow[i][ci++]=firs[here][l];

}

k=k+1;

if(G[j][k+1]>=65 && G[j][k+1]<=90)

{

goto label2;

}

}

else if(!isupper(G[j][k+1]) && G[j][k+1]!='|' )

{

follow[i][ci++]=G[j][k+1];

}}}}}

void fg()

{

change:

int total=0;

for(int j=0;j<cg;j++)

{

total=total +strlen(follow[j]);

}

for(int j=0;j<cg;j++)

{

int p=strlen(G[j]);

label:

if(!isupper(G[j][p-1]))

{

while(G[j][p-1]!='|' && G[j][p-1]!='>')

{

p--;

if(G[j][p-1]=='>')

{

break;

}

else if(G[j][p-1]=='|')

{

p--;

goto label;

}

}

}

else if(G[j][p-1]!='>')

{

int here;

for(int y=0;y<cg;y++)

{

if(G[y][0]==G[j][p-1])

{

here=y;

break;

}

}

int her;

for(int y=0;y<cg;y++)

{

if(G[y][0]==G[j][0])

{

her=y;

break;

}

}

int cf=strlen(follow[here]);

for(int x=0;x<strlen(follow[her]);x++)

{

int flag=0;

for(int c=0;c<strlen(follow[here]);c++)

{

if( follow[here][c]==follow[her][x])

{

flag=1;

}

}

if(flag==0)

{

follow[here][cf++]=follow[her][x];

}

}

for(int k=p-2;k>=3;k--)

{

int curr;

for(int y=0; y<cg && (isupper(G[j][k])); y++)

{

if(G[y][0]==G[j][k])

{

curr=y;

];

break;

}

}

int next;

for(int y=0; y<cg && (isupper(G[j][k])) && (G[j][k+1]!='|') ;y++)

{

if(G[y][0]==G[j][k+1])

{

next=y;

break;

}

}

if(G[j][k]>=65 && G[j][k]<=90)

{

if(G[j][k+1]=='|')

{

for(int x=0;x<strlen(follow[her]);x++)

{

int flag=0;

for(int c=0;c<strlen(follow[curr]);c++)

{

if( follow[curr][c]==follow[her][x])

flag=1;

}

if(flag==0)

follow[curr][cf++]=follow[her][x];

}

}

else{

int cf=strlen(follow[curr]);

for(int y=0;y<strlen(first[next]);y++)

{

if(first[next][y]=='e' )

{

for(int x=0;x<strlen(follow[her]);x++)

{

int flag=0;

for(int c=0;c<strlen(follow[curr]);c++)

{

if( follow[curr][c]==follow[her][x])

flag=1;

}

if(flag==0)

follow[curr][cf++]=follow[her][x];

}

break;}}}}

else

{ while(G[j][k]!='|' && G[j][k]!='>' )

{

if(G[j][k]=='>')

{ break;}

else if(G[j][k]=='|')

{

p=k;

goto label;

}

k--;

}

if(G[j][k]=='>')

{

k=-1;}}}}

else if(G[j][p-1]=='>')

{

// cout<<"\n Continuing";

continue;

}

}

int total1=0;

for(int j=0;j<cg;j++)

{

total1=total1+strlen(follow[j]);

}

if(total!=total1)

{

goto change;

}

else

{

// cout<<"\n YO BRO ";

cout<<endl;

return;

}

// cout<<"\n add gya ";

}

int main()

{

char follo[100][100];

cout<<"\nEnter number of rules";

cin>>rules;

cout<<endl;

for (int i=0;i<rules;i++)

{

cout<<"\nEnter production rule "<<i<<endl;

cin>>productions[i];}

for(int i=0;i<rules;i++)

flag[i]=0;

for(int i=0;i<rules;i++)

{

if(flag[i]==0)

{

strcpy(G[cg],productions[i]);

char ch=productions[i][0];

for(int j=i+1;j<rules;j++)

{

if(ch==productions[j][0])

{

flag[j]=1;

int len=strlen(productions[j]);

int u=3,cstr=0; char x[100];

while(u<=len)

{

x[cstr++]=productions[j][u];

u++;

}

strcat(G[cg],"|");

strcat(G[cg],x);

}

}

cg++;

flag[i]=1;

}}

for(int i=0;i<cg;i++)

{

Nonterminal[i]=G[i][0];

Number[i]=i;

}

for(int i=0;i<cg;i++)

{

if(check[i]==0){

int j= first1(i);

check[i]=1;

}

}

for(int p=0;p<cg;p++)

{

int SIZE=strlen(first[p]);

sort(first[p], first[p] + SIZE);

int pi=0;

if(SIZE==1)

{

for(int i=0;i<SIZE;i++)

{

firs[p][pi++]=first[p][i];

}

}

else{

for(int i=0;i<SIZE;i++)

{

if(first[p][i]!=first[p][i+1])

firs[p][pi++]=first[p][i];

}

}

}

for(int i=0;i<cg;i++)

check[i]=0;

for(int i=0;i<cg;i++)

{

if(check[i]==0){

int j= follow1(i);

check[i]=1;

}

}

fg();

for(int p=0;p<cg;p++)

{

int SIZE=strlen(follow[p]);

sort(follow[p], follow[p] + SIZE);

int pi=0;

if(SIZE==1)

{

for(int i=0;i<SIZE;i++)

{

follo[p][pi++]=follow[p][i];

}

}

else{

for(int i=0;i<SIZE;i++)

{

if(follow[p][i]!=follow[p][i+1])

{

follo[p][pi++]=follow[p][i];

}

}

}

}

cout<<endl;

for(int i=0;i<cg;i++)

{

cout<<endl<<Nonterminal[i]<<"'s first is "<<endl;

for(int j=0;j<strlen(firs[i]);j++)

{

if(firs[i][j]=='e'&&i==0)

cout<<firs[i][j]<<",";

}

}

cout<<endl<<endl;

for(int i=0;i<cg;i++)

{

cout<<Nonterminal[i]<<"'s follow is "<<endl;

for(int j=0;j<strlen(follo[i]);j++)

{

cout<<follo[i][j]<<",";

}

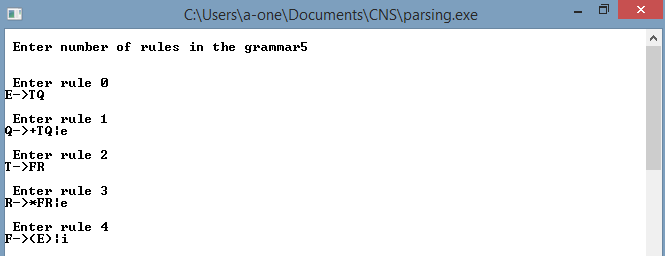
cout<<endl;

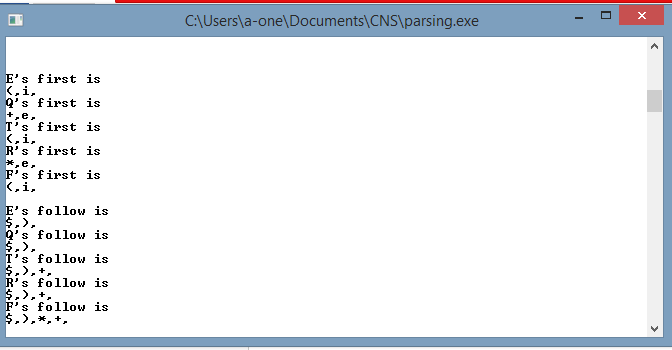
}

return 0;

}

**OUTPUT:**





**AIM:** Program for predictive parser for given grammar

**PROGRAM:**

#include<iostream>

#include<conio.h>

#include<cstring>

#include<fstream>

#include<cstring>

#include<stdio.h>

#include <algorithm>

#include<stack>

using namespace std;

int na1=0,ca1=0;

char first[1000][100],firs[1000][100],follow[1000][100],follo[1000][1000];

int check[1000];

void addnext(int,int,int);

int first1(int);

int follow1(int);

void replacet(int,int);

int nopipe(int);

void factoring(int index);

void fg();

char arr1[1000][1000];

char backup[1000][1000];

int conflict[100][100];

char prs[100];

char part[100][100][100],terminal[100],parts[100][100][100];

int nu[100],tt=0;

int trans[100][100];

string table[100][100];

ifstream file;

ofstream ofile;

ifstream fin;

ofstream fout;

void immediate();

void convert();

void parse();

int global=90;

int main()

{

cout<<"\n Enter number of rules in the grammar";

cin>>rules;

cout<<endl;

for(int i=0;i<rules;i++)

{

cout<<"\n Enter rule "<<i<<endl;

cin>>grammar[i];

if(grammar[i][0]>=97 && grammar[i][0]<=122 )

{

cout<<"\n First character should be non terminal so enter again ";

i=i-1;

}

}

for(int i=0;i<rules;i++)

flag[i]=0;

for(int i=0;i<rules;i++)

{

if(flag[i]==0)

{

strcpy(G[cg],grammar[i]);

char ch= grammar[i][0];

for(int j=i+1;j<rules;j++)

{

if(ch==grammar[j][0])

{

flag[j]=1;

int len=strlen(grammar[j]);

int u=3;

char x[100];

int cstr=0;

while(u<=len)

{

x[cstr++]=grammar[j][u];

u++;

}

strcat(G[cg],"|");

strcat(G[cg],x);

}

}

cg++;

flag[i]=1;

}

}

ofstream of;

of.open("result1.txt",ios::trunc);

for(int i=0;i<cg;i++)

{

Nonterminal[i]=G[i][0];

of<<G[i]<<"\n";

strcpy(backup[i],G[i]);

Number[i]=i;

}

of.close();

convert();

of.open("result.txt",ios::trunc);

of.close();

if(LL1==0)

{

ch='y';

while(ch=='y')

{

memset(prs,'\0',100);

cout<<endl<<" Enter the string to be parsed";

cin>>prs;

prs[strlen(prs)]='$';

cout<<endl;

parse();

cout<<endl;

cout<<"\n More string to parse ";

cin>>ch;

else

{

cout<<"\n String can't be parsed because G is not LL1";

void parsetable()

{ for(int i=0;i<100;i++)

for(int j=0;j<100;j++)

conflict[i][j]=0;}

for(int i=0;i<cg;i++)

{strcpy(S[i],G[i]);

cout<<S[i]<<endl;}

for(int i=0;i<cg;i++)

{ int k=0;

for(int j=3;j<strlen(G[i]);j++)

{

int num=0;

while(G[i][j]!='\0')

{

for(int i=0;i<cg;i++)

{

for(int j=3;j<strlen(S[i]);j++)

{

if(!isupper(S[i][j]) && S[i][j]!='|'&& S[i][j]!='e')

{

int flag=0;

for(int k=0;k<strlen(terminal);k++)

{

if(terminal[k]==S[i][j] )

flag=1;

}

if(flag==0)

terminal[strlen(terminal)]=S[i][j];

for(int i=0;i<cg;i++)

{

int flag=0;

char ch=S[i][0];

for(int j=0;j<nu[i];j++)

{

char str[1000];

memset(G[i],'\0',1000);

memset(str,'\0',1000);

str[0]=ch;

strcat(str,"->");

strcat(str,part[i][j]);

strcpy(G[i],str);

for(int t=0;t<cg;t++)

memset(first[t],'\0',1000);

for(int t=0;t<cg;t++)

check[t]=0;

for(int t=0;t<cg;t++)

{if(check[t]==0){

int j= first1(t);

check[t]=1;

sort(first[p], first[p] + SIZE);

int pi=0;

if(SIZE==1)

{

for(int i=0;i<SIZE;i++)

firs[p][pi++]=first[p][i];

else

{

for(int i=0;i<SIZE;i++)

{

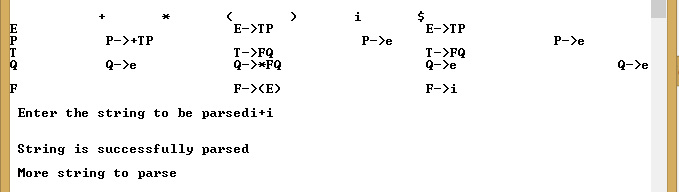
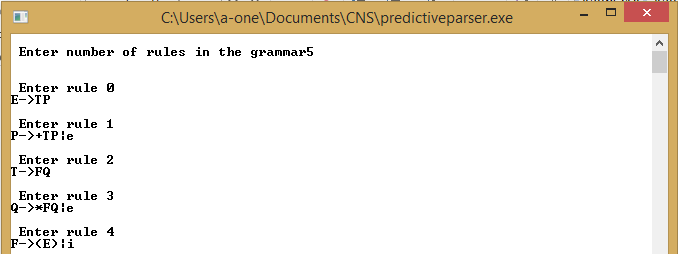
if(first[p][i]!=first[p][i+1])

firs[p][pi++]=first[p][i];

flag=0;

for(int t=0;t<strlen(firs[i]);t++)

**OUTPUT:**



**AIM:** Program for Shift-Reduce Parsing

**PROGRAM:**

#include<iostream>

#include<string.h>

using namespace std;

char rules[10][15],stack[10],ip[10],sc,ts[10];

int top,i=0,r,p,k,f,l,pos,m,np,cmp;

void shift()

{ top++;

sc=ip[i];

stack[top]=sc;}

int reduce()

{

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

{

f=0;

for(l=0;l<10;l++)

ts[l]='\0';

pos=0;

for(l=k;l<=top;l++)

{ ts[pos]=stack[l];

pos++;}

for(m=0;m<np;m++)

{ char prod[10];

int c=0,j;

for(j=3;rules[m][j]!='\0';j++){

prod[c]=rules[m][j];

c++;}

prod[c]='\0';

cmp=strcmp(ts,prod);

if(cmp==0)

{

for(l=k;l<=top;l++)

{stack[l]='\0';}

top=k;

stack[top]=rules[m][0];

r=2;

break;}}

int main()

{

strcpy(rules[0],"E->E+T");

strcpy(rules[1],"E->T");

strcpy(rules[2],"T->T\*F");

strcpy(rules[3],"T->F");

strcpy(rules[4],"F->(E)");

strcpy(rules[3],"F->i");

np=6;

cout<<"enter input:";

cin>>ip;

strcat(ip,"$");

int length=strlen(ip);

top=-1;

shift();

cout<<"\n\nStack\tInput\tAction\n";

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

do

{

i++;

r=1;

while(r!=0)

{ cout<<"\n";

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

cout<<stack[p];

cout<<"\t";

for(p=i;p<length;p++)

cout<<ip[p];

if(r==2) cout<<"\treduced";

else cout<<"\tshifted";

r=0;

r= reduce();}

shift();

}while(strlen(stack)!=1 && ip[i]!='$');

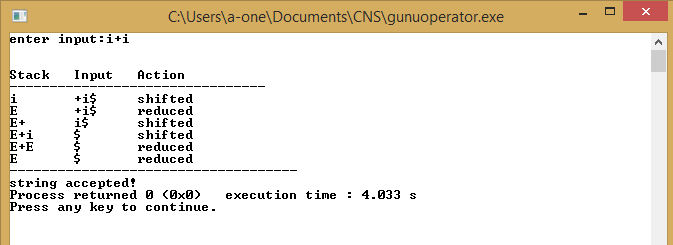
cout<<"\n------------------------------------";

if((ip[i]=='$')&&(strlen(stack)==2) && (stack[0]==rules[0][0]))

cout<<endl<<"string accepted!";

}

**OUTPUT:**



**AIM:** Program for Operator Precedence Parsing

**PROGRAM:**

#include<stdio.h>

#include<iostream>

#include<string.h>

#include<stdlib.h>

using namespace std;

char stack[10],s[10];

int top=0,i,j,k,ptr=0,p;

char prod[6][10]={

"><<<>>",

">><<>>",

">>ee>>",

"<<<<=e",

">>ee>>",

"<<<<ea"};

char c[6]={'+','\*','i','(',')','$'};

char symbol,next;

char G[6][10]={

"E->E+T",

" |T",

"T->T\*F",

" |F",

"F->(E)",

" |i" };

int f(char ch)

{

switch(ch)

{

case '+':return 0;

case '\*':return 1;

case 'i':return 2;

case '(':return 3;

case ')':return 4;

case '$':return 5;

default :

cout<<"\n ERROR entry";

exit(0);}}

int main()

{

printf("Grammar\n");

for(j=0;j<7;j++)

{

for(k=0;k<6;k++)

cout<<G[j][k];

cout<<endl;

}

cout<<"\n\n OPERATOR PRECEDENCE RELATIONS \n";

cout<<"\n ---------------------------------------------------------------------------\n";

cout<<"\t"<<'+'<<"\t"<<'\*'<<"\t"<<'i'<<"\t"<<'('<<"\t"<<')'<<"\t"<<'$'<<"\t";

cout<<"\n----------------------------------------------------------------------------\n";

for(j=0;j<6;j++)

{

cout<<c[j]<<"\t";

for(k=0;k<10;k++)

cout<<prod[j][k]<<"\t";

cout<<endl;

}

cout<<"Enter the string : ";

gets(s);

stack[top]='$';

s[strlen(s)]='$';

//cout<<s;

cout<<"Stack\tInput\tAction";

cout<<"\n------------------------------\n";

next=s[ptr];

symbol=prod[f(stack[top])][f(next)];

cout<<stack[top]<<"\t";

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

cout<<s[i];

while(1){

if(stack[top]=='$'&&s[ptr]=='$')

{cout<<"\n-----parsing successful------";

break;

}

else {

if (symbol=='<' || symbol=='=')

{

stack[++top]=symbol;

stack[++top]=next;

cout<<"\t shift\n";

ptr++;

}

else if(symbol=='>')

{

char handle[5]={'\0'};

stack[++top]=symbol;

int u=top;

while(stack[top]!='<')

{top--;}

int q=0;

for( k=top;k<=u;k++)

{

handle[q]=stack[k];

q++;

stack[k]='\0';}

handle[q]='\0';

top--;

cout<<"\t"<<handle<<" reduce"<<"\n";

}

next=s[ptr];

symbol=prod[f(stack[top])][f(next)];

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

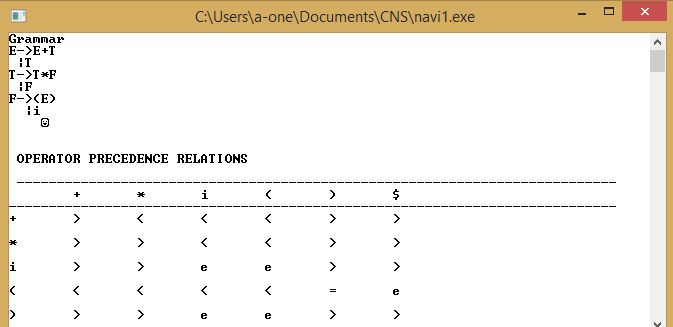
cout<<stack[i];

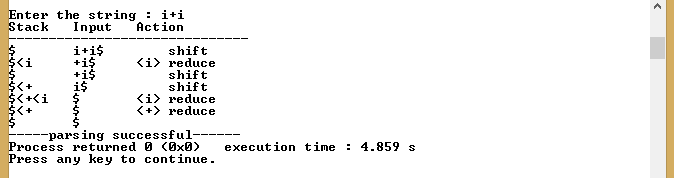
cout<<"\t";

for(int j=ptr;j<=strlen(s);j++)

cout<<s[j];}}

**OUTPUT:**





**AIM:** Program for creating Table for Operator Precedence parser

**PROGRAM:**

#include<stdio.h>

#include<iostream>

#include<string.h>

#include<stdlib.h>

#include<stack>

#include<stdlib.h>

using namespace std;

std::stack <char> s;

std::stack <char> s1;

int top=0,i,j,k1=0,k2=0,ptr=0,p;

int num,nt,t;

char G[10][10],NT[10],T[10],lead[10][10],trail[10][10],a,b,a1,b1,table[10][10],leading[10][10],trailing[10][10];

int flag1=0,flag2=0;

int searchT(char a)

{

int count=-1;i;

for(int i=0;i<t;i++)

{

if(T[i]==a)

return i;

}

return count;}

int searchNt(char a)

{

int count=-1,i;

for(int i=0;i<nt;i++)

{if(NT[i]==a)

return i;}

return count;

}

void addresult(int a,int b)

{

if(lead[a][b]=='f')

{

lead[a][b]='t';

push(NT[a]);

s.push(T[b]);

s.push(NT[a]);}}

void addresult1(int a,int b)

{

if(trail[a][b]=='f')

{trail[a][b]='t';

s1.push(T[b]);

s1.push(NT[a]);}}

int main()

{

cout<<"Enter number of productions::";

cin>>num;

cout<<"Enter productions one by one::\n";

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

{

cin>>G[i];

}

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

{

if(searchNt(G[i][0])==-1)

NT[nt++]=G[i][0];

}

NT[nt]='\0';

for(int p=0;p<nt;p++)

cout<<"nonterminals ::"<<NT[p]<<endl;

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

{

for(j=3;j<strlen(G[i]);j++){

if(searchNt(G[i][j])==-1)

{

if(searchT(G[i][j])==-1)

T[t++]=G[i][j];}}}

cout<<endl;

for(int p=0;p<t;p++)

cout<<"terminals ::"<<T[p]<<endl;

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

{

for (j=3;j<strlen(G[i]);j++)

{

if(searchNt(G[i][j])!=-1&& searchNt(G[i][j+1])!=-1)

{flag1=1;

break;}}}

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

{

for (j=3;j<strlen(G[i]);j++)

{

if(searchNt(G[i][j])=='$')

{flag2=1;

break;}}}

if(flag1==1||flag2==1)

{

cout<<"\n------grammar is not operator precedence--------\n";

exit(0);

}

cout<<flag2;

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

{

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

{

lead[i][j]='f';

}

}

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

{

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

{

trail[i][j]='f';

}

}

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

{

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

{

if(NT[searchNt(G[j][0])]==NT[i])

{

if(searchT(G[j][3])!=-1)

addresult(searchNt(G[j][0]),searchT(G[j][3]));

else

{

for(int k=3;k<strlen(G[j]);k++)

{

if(searchNt(G[j][k])==-1)

{

addresult(searchNt(G[j][0]),searchT(G[j][k]));

break;}}}}}}

while(!s.empty())

{

a=s.top();

s.pop();

b=s.top();

s.pop();

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

{

if(G[i][3]==a)

{

addresult(searchNt(G[i][0]),searchT(b));}}}

cout<<"\n----------------------------\n";

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

{

int o=0;

cout<<"Leading of"<<NT[i]<<"is::";

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

{

if(lead[i][j]=='t')

{

cout<<T[j]<<",";

leading[i][o]=T[j];

o++;}

}

cout<<endl;

}

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

{

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

{

if(NT[searchNt(G[j][0])]==NT[i])

{

if(searchT(G[j][strlen(G[j])-1])!=-1)

addresult1(searchNt(G[j][0]),searchT(G[j][strlen(G[j])-1]));

else

{

for(int k=(strlen(G[j])-1);k>=3;k--)

{

if(searchNt(G[j][k])==-1)

{

addresult1(searchNt(G[j][0]),searchT(G[j][k]));

break;}}}}}}

while(!s1.empty())

{

a1=s1.top();

s1.pop();

b1=s1.top();

s1.pop();

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

{

if(G[i][3]==a1)

{

addresult1(searchNt(G[i][0]),searchT(b1));}}}

cout<<"\n----------------------------\n";

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

{

int o=0;

cout<<"Trailing of"<<NT[i]<<"is::";

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

{

if(trail[i][j]=='t')

{

cout<<T[j]<<",";

trailing[i][o]=T[j];

o++;}

}

cout<<endl;

}

strcat(T,"$");

t=strlen(T);

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

{

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

table[i][j]='-';

}

table[0][0]=' ';

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

{ table[0][i+1]=T[i];

table[i+1][0]=T[i];}

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

{

for(j=0;j<strlen(leading[0]);j++)

{

if(leading[0][j]==T[i])

table[t][i+1]='<';

}

for(j=0;j<strlen(trailing[0]);j++)

{

if(trailing[0][j]==T[i])

table[i+1][t]='>';

}

}

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

{

for(j=3;j<strlen(G[i]);j++)

{ int pt=searchT(G[i][j]);

int pt2=searchT(G[i][j+2]);

if((pt !=-1)&&(pt2!=-1))

{

table[pt+1][pt2+1]='=';

}

}

}

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

{

for(j=3;j<strlen(G[i]);j++)

{ int pt=searchT(G[i][j]);

int pt2=searchNt(G[i][j+1]);

if((pt !=-1)&&(pt2!=-1))

{

for(int l=0;l<=t;l++){

for(int k=0;k<strlen(leading[pt2]);k++)

{

if(leading[pt2][k]==T[l])

table[pt+1][l+1]='<';}}}}

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

{

for(j=strlen(G[i]);j>3;j--)

{ int pt=searchT(G[i][j]);

int pt2=searchNt(G[i][j-1]);

if((pt !=-1)&&(pt2!=-1))

{

for(int l=0;l<=t;l++){

for(int k=0;k<strlen(trailing[pt2]);k++)

{

if(trailing[pt2][k]==T[l])

table[l+1][pt+1]='>';}}}}}}

cout<<endl;

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

{

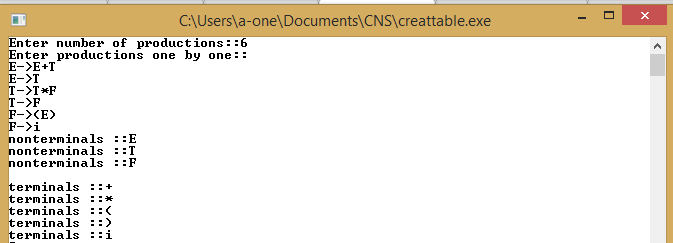
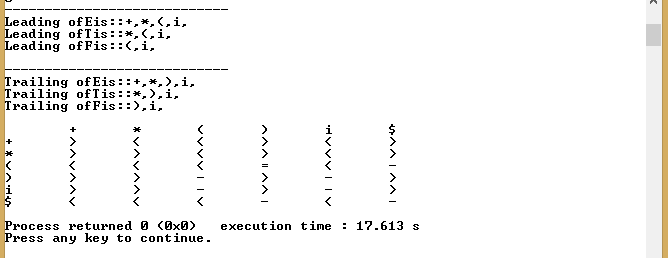
for(j=0;j<=t;j++)

cout<<table[i][j]<<"\t";

cout<<endl;

}

**OUTPUT:**

**AIM:** Program to implement LR parser

**PROGRAM:**

#include<iostream>

using namespace std;

#include<conio.h>

#include<string.h>

#include<stdio.h>

#include<stdlib.h>

#include<sstream>

#include<ctype.h>

char gram[10][10],s[20],stack[20], par[3]={},val[2];

int num,i,j,k,l,top=0,ptr=0,ps=0,acc=0,index,tlen=1;

char table[12][9][5]={{"s5"," "," ","s4"," "," ","1","2","3"},

{" ","s6"," "," "," ","a"," "," "," "},

{" ","r2","s7"," ","r2","r2"," "," "," "},

{" ","r4","r4"," ","r4","r4"," "," "," "},

{"s5"," "," ","s4"," "," ","8","2","3"},

{" ","r6","r6"," ","r6","r6"," "," "," "},

{"s5"," "," ","s4"," "," "," ","9","3"},

{"s5"," "," ","s4"," "," "," "," ","10"},

{" ","s6"," "," ","s11"," "," "," "," "},

{" ","r1","s7"," ","r1","r1"," "," "," "},

{" ","r3","r3"," ","r3","r3"," "," "," "},

{" ","r5","r5"," ","r5","r5"," "," "," "}};

char c[9]={'i','+','\*','(',')','$','E','T','F'};

int f(char ch)

{

switch(ch)

{

case 'i':return 0;

case '+':return 1;

case '\*':return 2;

case '(':return 3;

case ')':return 4;

case '$':return 5;

case 'E':return 6;

case 'T':return 7;

case 'F':return 8;

default :

cout<<"\n ERROR entry";

exit(0);}}

void shift()

{

top++; stack[top]= s[ptr]; ptr++;

for(l=1;l<strlen(par);l++)

{ top++;

stack[top]=par[l];}

if(strlen(par)>2)

{tlen=strlen(par)-1;}}

void reduce()

{ int o=0,gi;

char temp[10],git;

for(l=1;l<strlen(par);l++)

{ git=par[l];

gi=git-'0';}

gi=gi-1;

for(int z=3;z<strlen(gram[gi]);z++)

{temp[o]=gram[gi][z];

o++;}

temp[o]='\0';

int len=2\*strlen(temp);

int count=0;

if(tlen>1){

for(i=top;i>1;i--)

{if(isdigit(stack[i])&&isdigit(stack[i-1]))

count=count+1;}}

len+=count;

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

{stack[top]='\0';

top--;}

top++;

stack[top]=gram[gi][0];

if(stack[top]=='E'||stack[top]=='T'||stack[top]=='F')

{char a1=stack[top-1];

int a=a1-'0';

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

{if(stack[top]==c[i])

{tlen=strlen(table[a][i]);

for(int z=0;z<strlen(table[a][i]);z++)

{ top++;

stack[top]=table[a][i][z];}

break;}}}}

int main()

{

cout<<"enter the number of productions\n";

cin>>num;

cout<<"enter the productions one by one\n";

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

{

cin>>gram[i];}

cout<<"Given table is\n";

cout<<"------------------------------------------------------------------------------\n";

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

{

cout<<"\t"<<c[j]<<" ";}

cout<<"\n----------------------------------------------------------------------------\n";

for(i=0;i<=11;i++)

{cout<<i<<"\t";

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

{cout<<table[i][j]<<" ";}

cout<<endl;}

cout<<"\nEnter the string to parse :";

cin>>s;

s[strlen(s)]='$';

stack[top]='$';

stack[++top]='0';

cout<<"\nStack\t\tInput\t\tAction";

cout<<"\n------------------------------\n";

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

{cout<<stack[j];}

cout<<"\t\t";

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

cout<<s[i];

while(1)

{int va1[tlen],va;

if(tlen>1)

{for(int y=tlen-1;y>=0;y--)

{val[y]=stack[top];

top--;}

val[tlen]='\0';

for(int y=0;y<tlen;y++)

{va1[y]=val[y]-'0';

top++;}

va=va1[0]\*10+va1[1];}

else

{ val[0]=stack[top];

va=val[0]-'0';}

index=f(s[ptr]);

par[3]='\0';

int m=0;

for(k=0;k<strlen(table[va][index]);k++)

{ par[m]=table[va][index][k];

m++;}

par[m]='\0';

if(par[0]=='s')

{cout<<"\t\tshift\n";

shift();}

else if(par[0]=='r')

{cout<<"\t\treduce\n";

reduce();}

else if (par[0]=='a')

{ acc=1;

break;

cout<<"\t\taccept\n";}

else

{cout<<"\n\nerror entry\n\n";

exit(0);}

if(acc==1&&s[ptr]=='$')

cout<<"\n\n----------parsing successful--------\n\n";

return 0;}

**OUTPUT:**

