AMITY SCHOOL OF ENGINEERING  
AMITY UNIVERSITY

BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND ENGINERING



**COMPILER CONSTRUCTION**

**Submitted to: Submitted by:**

Ms. Preeti Sharma Jayant Bhalla  
A12405215021  
CSE-1X  
V Semester

**Table of Contents**

**Topic** **Page No**

1. Check String Acceptability by given grammar. 3
2. Conversion of Infix expression to Postfix expression. 5
3. Conversion of Infix expression to Prefix expression. 8
4. Counting Tokens in given grammar. 13
5. Conversion of Regular Expression to NFA. 15
6. Conversion of NFA to DFA. 19
7. Calculate Leading & Trailing. 24
8. Calculate First & Follow. 29
9. Design Shift Reduce Parser. 34
10. Design Top Down Parser. 38

**Exp-1: Check String Acceptability By A Given Grammar**   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program to check string Acceptability by a given Grammar. \*  
\* Author : Jayant Bhalla \*  
\* Date : 20 Sep 17. \*  
\* Language: C. \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// StringAcceptability.c : Defines the entry point for the console application.

//

#include <conio.h>

#include <string.h>

#include <stdio.h>

using namespace std;

void main()

{

printf("Checking String Acceptability by a grammar:\n\t\t\t\t~By Jayant Bhalla\n\n");

char string[20];

int state = 0, count = 0;

printf("the grammar is: S->aS | S->Sb | S->ab \n\n");

printf("enter the string to be checked \n\n");

gets\_s(string);

while (string[count] != '\0')

{

switch (state)

{

case 0: if (string[count] == 'a')

state = 1;  
 else  
 state = 3;

break;

case 1: if (string[count] == 'a')  
 state = 1;  
 else if (string[count] == 'b')  
 state = 2;  
 else  
 state = 3;

break;

case 2: if (string[count] == 'b')  
 state = 2;

else

state = 3;

break;

default: break;

}

count++;

if (state == 3)  
 break;

}

if (state == 2)

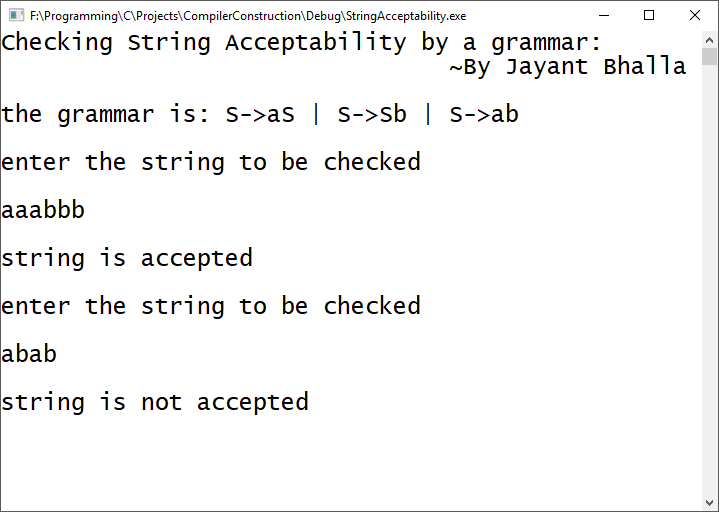
printf("\nstring is accepted\n\n");

else

printf("\nstring is not accepted\n\n");

getch();

}

**OUTPUT**

**Exp-2: Conversion of Infix expression to Postfix expression**   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program to convert Infix expression to Postfix expression. \*  
\* Author : Jayant Bhalla \*  
\* Date : 20 Sep 17. \*  
\* Language: C. \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// InfixToPostfix.c : Defines the entry point for the console application.

//

#include <iostream.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <ctype.h>

#define SIZE 50 /\* Size of Stack \*/

char s[SIZE];

int top = -1; /\* Global declarations \*/

void push(char elem)

{ /\* Function for PUSH operation \*/

s[++top] = elem;

}

char pop()

{ /\* Function for POP operation \*/

return(s[top--]);

}

int pr(char elem)

{ /\* Function for precedence \*/

switch (elem)

{

case '#': return 0;

case '(': return 1;

case '+':

case '-': return 2;

case '\*':

case '/': return 3;

}

}

int main()

{ char infx[50], pofx[50], ch, elem;

int i = 0, k = 0;

printf("Converting Infix to Postfix:\n\t\t\t~By Jayant Bhalla\n\n");

printf("\n\nEnter the Infix Expression: ");

scanf("%s", infx);

push('#');

while ((ch = infx[i++]) != '\0')

{

if (ch == '(') push(ch);

else

if (isalnum(ch)) pofx[k++] = ch;

else

if (ch == ')')

{

while (s[top] != '(')

pofx[k++] = pop();

elem = pop(); /\* Remove ( \*/

}

else

{ /\* Operator \*/

while (pr(s[top]) >= pr(ch))

pofx[k++] = pop();

push(ch);

}

}

while (s[top] != '#') /\* Pop from stack till empty \*/

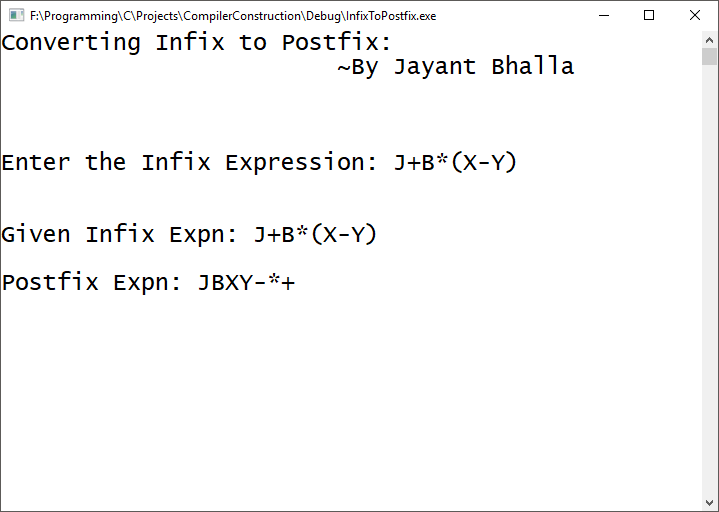
pofx[k++] = pop();

pofx[k] = '\0'; /\* Make pofx as valid string \*/

printf("\n\nGiven Infix Expn: %s\n\nPostfix Expn: %s", infx, pofx);

getch();

}

**OUTPUT** 

**Exp-3: Conversion of Infix expression to Prefix expression**   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program to convert Infix expression to Prefix expression. \*  
\* Author : Jayant Bhalla \*  
\* Date : 20 Sep 17. \*  
\* Language: C. \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// InfixToPrefix.c : Defines the entry point for the console application.

//

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <ctype.h>

#define MAX 50  
struct infix

{

char target[MAX];

char stack[MAX];

char \*s, \*t;

int top, l;

};

void initinfix(struct infix \*);

void setexpr(struct infix \*, char \*);

void push(struct infix \*, char);

char pop(struct infix \*);

void convert(struct infix \*);

int priority(char c);

void show(struct infix);

void main()

{

struct infix q;

char expr[MAX];

initinfix(&q);

printf("Converting Infix to Prefix:\n\t\t\t~By Jayant Bhalla\n\n");  
 printf("\nEnter an expression in infix form: ");  
 gets\_s(expr);  
 setexpr(&q, expr);  
 convert(&q);

printf("\nThe Prefix expression is: ");

show(q);  
 getch();

}

/\* initializes elements of structure variable \*/

void initinfix(struct infix \*pq)

{

pq->top = -1;

strcpy\_s(pq->target, "");

strcpy\_s(pq->stack, "");

pq->l = 0;

}

/\* reverses the given expression \*/

void setexpr(struct infix \*pq, char \*str)

{

pq->s = str;

\_strrev(pq->s);

pq->l = strlen(pq->s);

\*(pq->target + pq->l) = '\0';

pq->t = pq->target + (pq->l - 1);

}

/\* adds operator to the stack \*/

void push(struct infix \*pq, char c)

{

if (pq->top == MAX - 1)

printf("\nStack is full.\n");

else

{

pq->top++;

pq->stack[pq->top] = c;

}

}

/\* pops an operator from the stack \*/

char pop(struct infix \*pq)

{

if (pq->top == -1)

{

printf("Stack is empty\n");

return -1;

}

else

{

char item = pq->stack[pq->top];

pq->top--;

return item;

}

}

/\* converts the infix expr. to prefix form \*/

void convert(struct infix \*pq)

{

char opr;

while (\*(pq->s))

{

if (\*(pq->s) == ' ' || \*(pq->s) == '\t')

{

pq->s++;

continue;

}

if (isdigit(\*(pq->s)) || isalpha(\*(pq->s)))

{

while (isdigit(\*(pq->s)) || isalpha(\*(pq->s)))

{

\*(pq->t) = \*(pq->s);

pq->s++;

pq->t--;

}

}

if (\*(pq->s) == ')')

{

push(pq, \*(pq->s));

pq->s++;

}

if (\*(pq->s) == '\*' || \*(pq->s) == '+' || \*(pq->s) == '/' ||

\*(pq->s) == '%' || \*(pq->s) == '-' || \*(pq->s) == '$')

{

if (pq->top != -1)

{

opr = pop(pq);

while (priority(opr) > priority(\*(pq->s)))

{

\*(pq->t) = opr;

pq->t--;

opr = pop(pq);

}

push(pq, opr);

push(pq, \*(pq->s));

}

else

push(pq, \*(pq->s));

pq->s++;

}

if (\*(pq->s) == '(')

{

opr = pop(pq);

while (opr != ')')

{

\*(pq->t) = opr;

pq->t--;

opr = pop(pq);

}

pq->s++;

}

}

while (pq->top != -1)

{

opr = pop(pq);

\*(pq->t) = opr;

pq->t--;

}

pq->t++;

}

/\* returns the priotity of the operator \*/

int priority(char c)

{

if (c == '$')

return 3;

if (c == '\*' || c == '/' || c == '%')

return 2;

else

{

if (c == '+' || c == '-')

return 1;

else

return 0;

}

}

/\* displays the prefix form of given expr. \*/

void show(struct infix pq)

{

while (\*(pq.t))

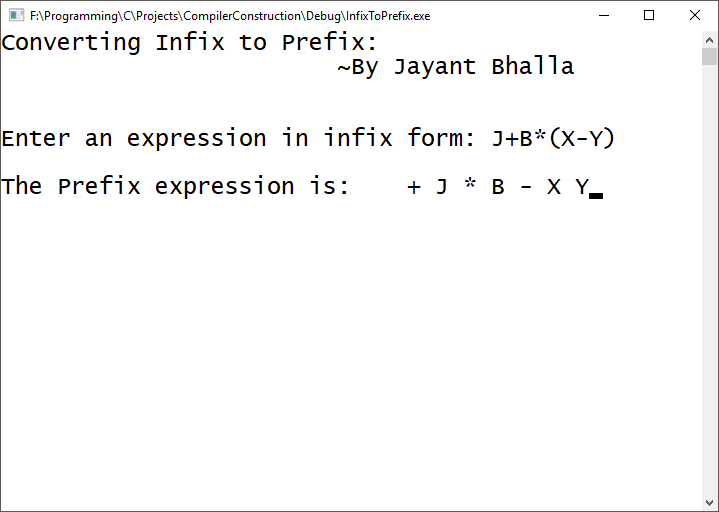
{

printf(" %c", \*(pq.t));

pq.t++;

}

}

**OUTPUT**

**Exp-4: Count Tokens in a given expression**   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program to count Tokens in a given expression. \*  
\* Author : Jayant Bhalla \*  
\* Date : 20 Sep 17. \*  
\* Language: C. \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// ExpressionTokenCount.c : Defines the entry point for the console application.

//

#include <stdio.h>

#include <ctype.h>

#include <conio.h>

#include <string.h>

int main()

{

char str[50];

int len;

int i, a = 0, b = 0, d = 0, f = 0, var = 0, tokens = 0, constant = 0, oper = 0;

printf("Counting tokens in an expression:\n\t\t\t~By Jayant Bhalla\n\n");

printf("enter string :");

scanf("%s", str);

len = strlen(str);

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

{

if (isalpha(str[i]))

a++;

if (isdigit(str[i]))

{

while (isdigit(str[i]))

{

i++;

}

d++;

}

if (str[i] == '%' || str[i] == '\*' || str[i] == '/' || str[i] == '+' || str[i] == '-' || str[i] == '=')

f++;

else

b++;

}

var = a;

constant = d;

oper = f;

tokens = var + constant + oper;

printf("\ntotalvar:%d ", var);

printf("\ntotal constants:%d", constant);

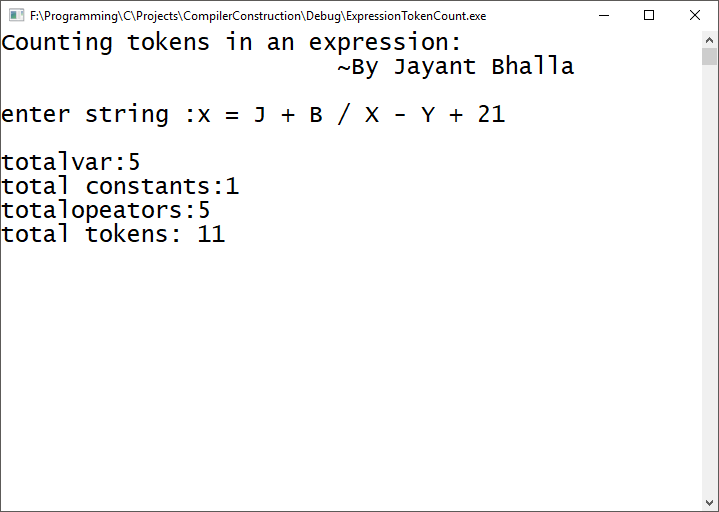
printf("\ntotalopeators:%d", oper);

printf("\ntotal tokens: %d", tokens);

getch();

return 0;

}

**OUTPUT**

**Exp-5: Conversion of Regular Expression to NFA**   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program to convert Regular expression to NFA. \*  
\* Author : Jayant Bhalla \*  
\* Date : 20 Sep 17. \*  
\* Language: C. \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// ExpressionToNFA.c : Defines the entry point for the console application.

//

#include <stdio.h>

#include <string.h>

#include <conio.h>

int main()

{

char reg[20];

int q[20][3], i, j, len, a, b;

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

{

for (b = 0; b < 3; b++)

{

q[a][b] = 0;

}

}

printf("Convert Regular Expression to NFA:\n\t\t\t~By Jayant Bhalla\n\n");

printf("Enter a regular expression: \n");

scanf("%s", reg);

len = strlen(reg);

i = 0;

j = 1;

while (i < len)

{

if (reg[i] == 'a'&&reg[i + 1] != '/'&&reg[i + 1] != '\*')

{

q[j][0] = j + 1;

j++;

}

if (reg[i] == 'b'&&reg[i + 1] != '/'&&reg[i + 1] != '\*')

{

q[j][1] = j + 1;

j++;

}

if (reg[i] == 'e'&&reg[i + 1] != '/'&&reg[i + 1] != '\*')

{

q[j][2] = j + 1;

j++;

}

if (reg[i] == 'a'&&reg[i + 1] == '/'&&reg[i + 2] == 'b')

{

q[j][2] = ((j + 1) \* 10) + (j + 3);

j++;

q[j][0] = j + 1;

j++;

q[j][2] = j + 3;

j++;

q[j][1] = j + 1;

j++;

q[j][2] = j + 1;

j++;

i = i + 2;

}

if (reg[i] == 'b'&&reg[i + 1] == '/'&&reg[i + 2] == 'a')

{

q[j][2] = ((j + 1) \* 10) + (j + 3);

j++;

q[j][1] = j + 1;

j++;

q[j][2] = j + 3;

j++;

q[j][0] = j + 1;

j++;

q[j][2] = j + 1;

j++;

i = i + 2;

}

if (reg[i] == 'a'&&reg[i + 1] == '\*')

{

q[j][2] = ((j + 1) \* 10) + (j + 3);

j++;

q[j][0] = j + 1;

j++;

q[j][2] = ((j + 1) \* 10) + (j - 1);

j++;

}

if (reg[i] == 'b'&&reg[i + 1] == '\*')

{

q[j][2] = ((j + 1) \* 10) + (j + 3);

j++;

q[j][1] = j + 1;

j++;

q[j][2] = ((j + 1) \* 10) + (j - 1);

j++;

}

if (reg[i] == ')'&&reg[i + 1] == '\*')

{

q[0][2] = ((j + 1) \* 10) + 1;

q[j][2] = ((j + 1) \* 10) + 1;

j++;

}

i++;

}

printf("Transition function \n");

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

{

if (q[i][0] != 0)

printf("\n q[%d,a]-->%d", i, q[i][0]);

if (q[i][1] != 0)

printf("\n q[%d,b]-->%d", i, q[i][1]);

if (q[i][2] != 0)

{

if (q[i][2] < 10)

printf("\n q[%d,e]-->%d", i, q[i][2]);

else

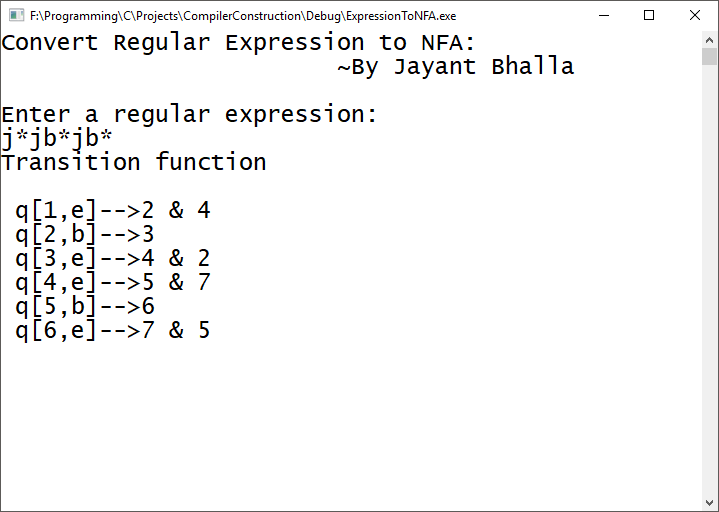
printf("\n q[%d,e]-->%d & %d", i, q[i][2] / 10, q[i][2] % 10);

}

getch();

}

}

**OUTPUT**

**Exp-6: Conversion of NFA to DFA**   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program to convert NFA to DFA. \*  
\* Author : Jayant Bhalla \*  
\* Date : 20 Sep 17. \*  
\* Language: C. \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// NFAtoDFA.c : Defines the entry point for the console application.

//

#include <stdio.h>

#include <string.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

char nfa[50][50], s[20], st[10][20], eclos[20], input[20];

int x, e, top = 0, topd = 0, n = 0, ns, nos, in;

int checke(char a)

{

int i;

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

{

if (eclos[i] == a)

return i;

}

return -1;

}

int check(char a)

{

int i;

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

{

if (input[i] == a)

return i;

}

return -1;

}

void push(char a)

{

s[top] = a;

top++;

}

char pop()

{

top--;

return s[top];

}

void pushd(char \*a)

{

strcpy(st[topd], a);

topd++;

}

char \*popd()

{

topd--;

return st[topd];

}

int ctoi(char a)

{

int i = a - 48;

return i;

}

char itoc(int a)

{

char i = a + 48;

return i;

}

char \*eclosure(char \*a)

{

int i, j;

char c;

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

push(a[i]);

e = strlen(a);

strcpy(eclos, a);

while (top != 0)

{

c = pop();

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

{

if (nfa[ctoi(c)][j] == 'e')

{

if (check(itoc(j)) == -1)

{

eclos[e] = itoc(j);

push(eclos[e]);

e++;

}

}

}

}

eclos[e] = '\0';

return eclos;

}

void main()

{

int i, j, k, count;

char ec[20], a[20], b[20], c[20], dstates[10][10];

printf("Convert NFA to DFA:\n\t\t~By Jayant Bhalla\n\n");

printf("Enter the number of states\n");

cin >> ns;

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

{

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

{

cout << "Move[" << i << "][" << j << "]";

cin >> nfa[i][j];

if (nfa[i][j] != '-'&&nfa[i][j] != 'e')

{

if ((check(nfa[i][j])) == -1)

input[in++] = nfa[i][j];

}

}

}

topd = 0;

nos = 0;

c[0] = itoc(0);

c[1] = '\0';

pushd(eclosure(c));

strcpy(dstates[nos], eclosure(c));

for (x = 0; x<in; x++)

cout << "\t" << input[x];

cout << "\n";

while (topd>0)

{

strcpy(a, popd());

cout << a << "\t";

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

{

int len = 0;

for (j = 0; j<strlen(a); j++)

{

int x = ctoi(a[j]);

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

{

if (nfa[x][k] == input[i])

ec[len++] = itoc(k);

}

}

ec[len] = '\0';

strcpy(b, eclosure(ec));

count = 0;

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

{

if (strcmp(dstates[j], b) == 0)

count++;

}

if (count == 0)

{

if (b[0] != '\0')

{

nos++;

pushd(b);

strcpy(dstates[nos], b);

}

}

cout << b << "\t";

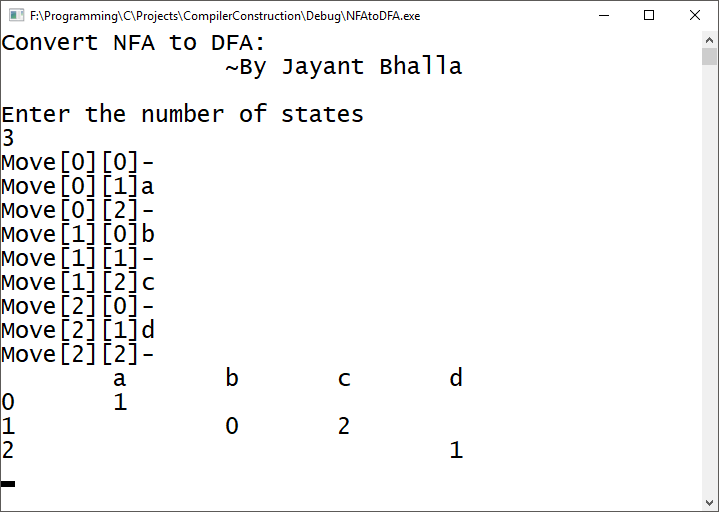
}

cout << endl;

}

getch();

}

**OUTPUT**

**Exp-7: Leading and Trailing**  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program to calculate Leading & Trailing. \*  
\* Author : Jayant Bhalla \*  
\* Date : 20 Sep 17. \*  
\* Language: C. \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// LeadingTrailing.c : Defines the entry point for the console application.

//

#include <iostream.h>

#include <conio.h>

int nt, t, top = 0;

char s[50], NT[10], T[10], st[50], l[10][10], tr[50][50];

int searchnt(char a)

{

int count = -1, i;

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

{

if (NT[i] == a)

return i;

}

return count;

}

int searchter(char a)

{

int count = -1, i;

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

{

if (T[i] == a)

return i;

}

return count;

}

void push(char a)

{

s[top] = a;

top++;

}

char pop()

{

top--;

return s[top];

}

void installl(int a, int b)

{

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

{

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

push(T[b]);

push(NT[a]);

}

}

void installt(int a, int b)

{

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

{

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

push(T[b]);

push(NT[a]);

}

}

void main()

{

int i, s, k, j, n;

char pr[30][30], b, c;

printf("Calculate Leading & Trailing:\n\t\t\t~By Jayant Bhalla\n\n");

cout << "Enter the no of production rules:";

cin >> n;

cout << "Enter the production rules one by one\n";

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

cin >> pr[i];

nt = 0;

t = 0;

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

{

if ((searchnt(pr[i][0])) == -1)

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

}

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

{

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

{

if (searchnt(pr[i][j]) == -1)

{

if (searchter(pr[i][j]) == -1)

T[t++] = pr[i][j];

}

}

}

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

{

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

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

}

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

{

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

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

}

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

{

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

{

if (NT[(searchnt(pr[j][0]))] == NT[i])

{

if (searchter(pr[j][3]) != -1)

installl(searchnt(pr[j][0]), searchter(pr[j][3]));

else

{

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

{

if (searchnt(pr[j][k]) == -1)

{

installl(searchnt(pr[j][0]), searchter(pr[j][k]));

break;

}

}

}

}

}

}

while (top != 0)

{

b = pop();

c = pop();

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

{

if (pr[s][3] == b)

installl(searchnt(pr[s][0]), searchter(c));

}

}

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

{

cout << "Leading[" << NT[i] << "]" << "\t{";

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

{

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

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

}

cout << "}\n";

}

top = 0;

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

{

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

{

if (NT[searchnt(pr[j][0])] == NT[i])

{

if (searchter(pr[j][strlen(pr[j]) - 1]) != -1)

installt(searchnt(pr[j][0]), searchter(pr[j][strlen(pr[j]) - 1]));

else

{

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

{

if (searchnt(pr[j][k]) == -1)

{

installt(searchnt(pr[j][0]), searchter(pr[j][k]));

break;

}

}

}

}

}

}

while (top != 0)

{

b = pop();

c = pop();

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

{

if (pr[s][3] == b)

installt(searchnt(pr[s][0]), searchter(c));

}

}

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

{

cout << "Trailing[" << NT[i] << "]" << "\t{";

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

{

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

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

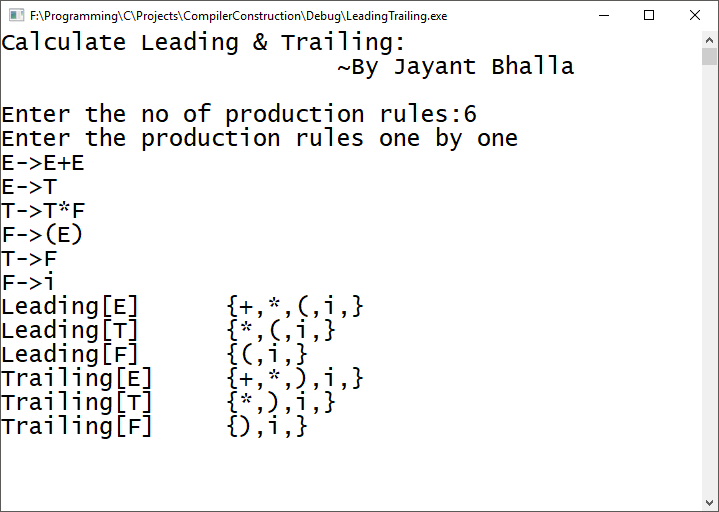
}

cout << "}\n";

}

getch();

}

**OUTPUT**

**Exp-8: FIRST and FOLLOW**  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program to calculate First & Follow. \*  
\* Author : Jayant Bhalla \*  
\* Date : 20 Sep 17. \*  
\* Language: C. \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// FirstFollow.c : Defines the entry point for the console application.

//

#include <stdio.h>

#include <conio.h>

#include <ctype.h>

#include <iostream.h>

int n, m = 0, p, i = 0, j = 0;

char a[10][10], followResult[10];

void follow(char c);

void first(char c);

void addToResult(char);

void addToResultSet(char[], char);

void FIRST(char[], char);

int main()

{

int i;

char c, choice;

char firstResult[20];

printf("\t\tCalculate FIRST & FOLLOW:\n\t\t\t\t~By Jayant Bhalla\n\n");

printf("Enter the no.of productions: ");

scanf("%d", &n);

printf("\n");

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

{

printf("Enter productions Number %d : ", i + 1);

scanf(" %s", a[i], \_countof(a));

}

do

{

m = 0;

printf("\nFind FIRST & FOLLOW of -->");

scanf(" %c", &c);

follow(c);

FIRST(firstResult, c);

printf("\nFIRST(%c)= { ", c);

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

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

printf("}\n");

printf("FOLLOW(%c) = { ", c);

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

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

printf(" }\n");

printf("\nDo you want to continue(y/n)?");

scanf(" %c", &choice);

} while (choice == 'Y' || choice == 'y');

}

void follow(char c)

{

if (a[0][0] == c)addToResult('$');

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

{

for (j = 2; j < strlen(a[i]); j++)

{

if (a[i][j] == c)

{

if (a[i][j + 1] != '\0')first(a[i][j + 1]);

if (a[i][j + 1] == '\0'&&c != a[i][0])

follow(a[i][0]);

}

}

}

}

void FIRST(char\* Result, char c)

{

int i, j, k;

char subResult[20];

int foundEpsilon;

subResult[0] = '\0';

Result[0] = '\0';

if (!(isupper(c)))

{

addToResultSet(Result, c);

return;

}

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

{

if (a[i][0] == c)

{

if (a[i][2] == '$') addToResultSet(Result, '$');

else

{

j = 2;

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

{

foundEpsilon = 0;

FIRST(subResult, a[i][j]);

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

addToResultSet(Result, subResult[k]);

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

if (subResult[k] == '$')

{

foundEpsilon = 1;

break;

}

if (!foundEpsilon)

break;

j++;

}

}

}

}

return;

}

void first(char c)

{

int k;

if (!(isupper(c)))

{

addToResult(c);

}

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

{

if (a[k][0] == c)

{

if (a[k][2] == '$') follow(a[i][0]);

else if (islower(a[k][2]))

{

addToResult(a[k][2]);

}

else first(a[k][2]);

}

}

}

void addToResult(char c)

{

int i;

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

if (followResult[i] == c)

return;

followResult[m++] = c;

}

void addToResultSet(char Result[], char val)

{

int k;

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

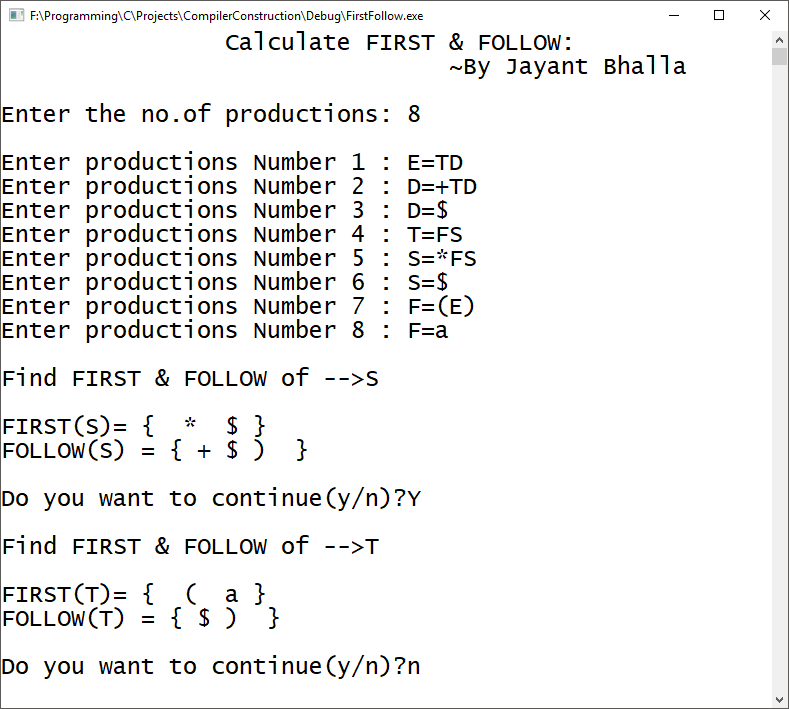
if (Result[k] == val)

return;

Result[k] = val;

Result[k + 1] = '\0';

}

**OUTPUT**

**Exp-9: Shift Reduce Parser**   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program to design a Shift Reduce Parser. \*  
\* Author : Jayant Bhalla \*  
\* Date : 20 Sep 17. \*  
\* Language: C. \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// ShiftReduceParser.c : Defines the entry point for the console application.

//

#include <stdio.h>

#include <conio.h>

#include <string.h>

char exp[30], stack[30], arr[30], temp[30];

int i, k = 0, j, l, r, s;

void push(char exp[])

{

arr[i] = exp[k];

i++;

}

void dispinp()

{

printf("\t\t\t");

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

printf("%c", exp[k]);

printf("$");

}

void dispstk()

{

printf("\n");

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

printf("%c", stack[k]);

}

void assign()

{

stack[++j] = arr[i];

exp[i] = ' ';

dispstk();

dispinp();

}

int main()

{

printf("\t\t\tSHIFT REDUCE PARSER\n\t\t\t\t\t~By Jayant Bhalla\n\n");

printf("\nThe Production is: E->E+E/E\*E/d/a\n");

printf("\nEnter the string to be parsed:\n");

scanf("%c", &exp);

printf("\nSTACK\t\t\tINPUT\t\t\tACTION\n");

printf("\n$");

dispinp();

printf("\t\t\tShift");

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

push(exp);

l = strlen(exp);

stack[0] = '$';

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

{

switch (arr[i])

{

case 'a':

assign();

printf("\t\t\tReduce by E->d");

stack[j] = 'E';

dispstk();

dispinp();

if (arr[i + 1] != '\0')

printf("\t\t\tShift");

break;

case 'd':

assign();

printf("\t\t\tReduce by E->d");

stack[j] = 'E';

dispstk();

dispinp();

if (arr[i + 1] != '\0')

printf("\t\t\tShift");

break;

case '+':

assign();

printf("\t\t\tShift");

break;

case '\*':

assign();

printf("\t\t\tShift");

break;

case '-':

assign();

printf("\t\t\tShift");

break;

default:

printf("\nError:String not accepted");

goto label;

}

}

l = strlen(stack);

while (l > 2)

{

r = 0;

for (i = l - 1; i >= l - 3; i--)

{

temp[r] = stack[i];

r++;

}

temp[r] = NULL;

if ((strcmp(temp, "E+E") == 0) || (strcmp(temp, "E\*E") == 0) || (strcmp(temp, "E\*E") == 0))

{

for (i = l; i > l - 3; i--)

stack[i] = ' ';

stack[l - 3] = 'E';

printf("\t\t\tReduce by E->");

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

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

dispstk();

dispinp();

l = l - 2;

}

else

{

printf("\nError:String not accepted");

goto label;

}

}

printf("\t\t\tAccept");

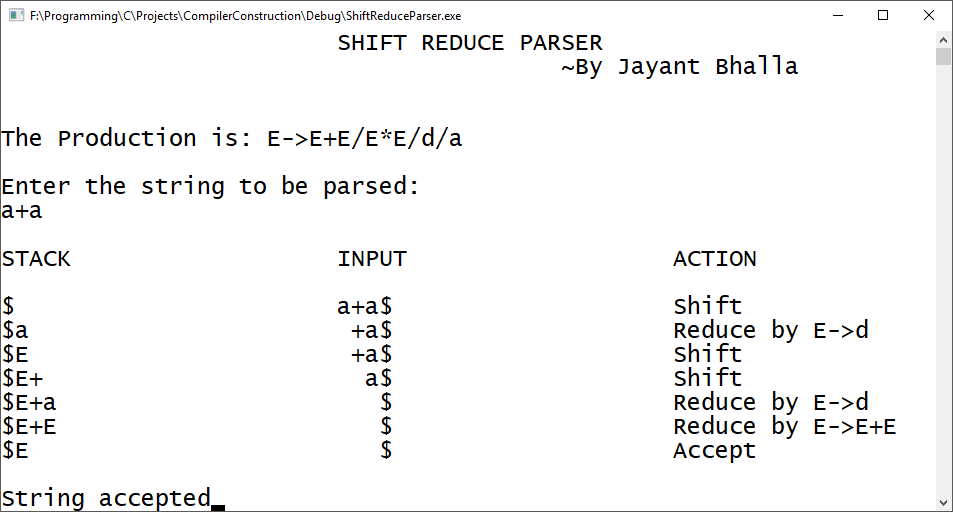
printf("\n\nString accepted");

label:

getch();

return 0;

}

**OUTPUT**

**Exp-10: Top Down Parser**   
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Program to design a Top Down Parser. \*  
\* Author : Jayant Bhalla \*  
\* Date : 20 Sep 17. \*  
\* Language: C. \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// TopDownParser.c : Defines the entry point for the console application.

//

#include <stdio.h>

#include <conio.h>

#include <iostream.h>

#include <string.h>

class parse

{

int nt, t, m[20][20], i, s, n, p1, q, k, j;

char p[30][30], n1[20], t1[20], ch, b, c, f[30][30], fl[30][30];

public:

int scant(char);

int scannt(char);

void process();

void input();

};

int parse::scannt(char a)

{

int c = -1, i;

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

{

if (n1[i] == a)

{

return i;

}

}

return c;

}

int parse::scant(char b)

{

int c1 = -1, j;

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

{

if (t1[j] == b)

{

return j;

}

}

return c1;

}

void parse::input()

{

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

cin >> n;

cout << "Enter the productions one by one" << endl;

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

cin >> p[i];

nt = 0;

t = 0;

}

void parse::process()

{

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

{

if (scannt(p[i][0]) == -1)

n1[nt++] = p[i][0];

}

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

{

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

{

if (p[i][j] != 'e')

{

if (scannt(p[i][j]) == -1)

{

if ((scant(p[i][j])) == -1)

t1[t++] = p[i][j];

}

}

}

}

t1[t++] = '$';

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

{

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

m[i][j] = -1;

}

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

{

cout << "Enter first[" << n1[i] << "]:";

cin >> f[i];

}

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

{

cout << "Enter follow[" << n1[i] << "]:";

cin >> fl[i];

}

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

{

p1 = scannt(p[i][0]);

if ((q = scant(p[i][3])) != -1)

m[p1][q] = i;

if ((q = scannt(p[i][3])) != -1)

{

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

m[p1][scant(f[q][j])] = i;

}

if (p[i][3] == 'e')

{

for (j = 0; j<strlen(fl[p1]); j++)

m[p1][scant(fl[p1][j])] = i;

}

}

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

cout << "\t" << t1[i];

cout << endl;

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

{

cout << n1[j];

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

{

cout << "\t" << " ";

if (m[j][i] != -1)

cout << p[m[j][i]];

}

cout << endl;

}

}

void main()

{

parse p;

printf("\t\t\tTOP DOWN PARSER\n\t\t\t\t\t~By Jayant Bhalla\n\n");

p.input();

p.process();

getch();

}

**OUTPUT**