**NAME:-VIVEK BULANI**

**ROLL NO:-SECOA115**

**ASSIGNMENT NO:-09**

**AIM** **:-** Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions i. Operands and operator, both must be single character. ii. Input Postfix expression must be in a desired format. iii. Only '+', '-', '\*' and '/ ' operators are expected.

**PROGRAM:-**

#include<iostream>

using namespace std;

class stack

{

char stk[30];

int top;

public:

stack()

{

top=-1;

}

bool is\_stack\_empty();

bool is\_stack\_full();

void push(char);

//void push\_int(int);

char pop();

//int pop\_int();

int priority(char);

void convert(char[],char[]);

int evaluate(int,char,int);

void result\_post(char[]);

};

bool stack::is\_stack\_empty()

{

if(top==-1)

return 1;

else

return 0;

}

bool stack::is\_stack\_full()

{

if(top==29)

return 1;

else

return 0;

}

void stack::push(char ch)

{

if(is\_stack\_full())

cout<<"\n Sorry..stack is full";

else

{

top++;

stk[top]=ch;

}

}

char stack::pop()

{

char temp;

if(is\_stack\_empty())

cout<<"Sorry..stack is empty";

else

{

temp=stk[top];

top--;

}

return temp;

}

int stack::priority(char ch)

{

if(ch=='(')

return 0;

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

return 1;

else if(ch=='\*' || ch=='/' || ch=='%')

return 2;

else if(ch=='^')

return 3;

}

void stack::convert(char inf[],char post[])

{

int i;

int j=0;

char token,ch;

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

{

token=inf[i];

if(token=='(')

push(token);

else if(token==')')

{

for(int i=0;stk[top]!='(';i++)

{

//ch=stk[top];

ch=pop();

post[j]=ch;

j++;

}

//ch=stk[top];

ch=pop();

}

else if(token=='+'||token=='-'||token=='\*'||token=='/'||token=='%'||token=='^')

{

if(priority(stk[top])<priority(token))

{

push(token);

}

else

{

while(priority(stk[top])>=priority(token))

{

ch=pop();

post[j]=ch;

j++;

//push(token);

}

push(token);

}

}

else

{

post[j]=token;

j++;

}

}

while(!is\_stack\_empty())

{

ch=pop();

post[j]=ch;

j++;

}

post[j]='\0';

}

int stack::evaluate(int op1,char opr,int op2)

{

if(opr=='+')

return(op1+op2);

else if(opr=='-')

return(op1-op2);

else if(opr=='\*')

return(op1\*op2);

else if(opr=='/')

return(op1/op2);

else if(opr=='%')

return(op1%op2);

}

void stack::result\_post(char post[])

{

char x;

int op1,op2,b,val;

int i=0;

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

{

x=post[i];

if(x=='+'||x=='-'||x=='\*'||x=='/'||x=='%')

{

op2=pop();

op1=pop();

val=evaluate(op1,x,op2);

push(val);

}

else

{

b=(int)x;

push(b-48);

}

i++;

}

val=pop();

cout<<"\n result is="<<val;

}

int main()

{

stack s;

char inf[30];

char post[30];

cout<<"\n Enter infix expression";

cin>>inf;

cout<<"\n Given infix expression is:";

cout<<"\n"<<inf;

s.convert(inf,post);

cout<<"\n Conversion of infix to postfix";

cout<<"\n Postfix expression is:";

cout<<post;

cout<<"\n Evaluation of postfix expression is";

s.result\_post(post);

return 0;

}

**OUTPUT**

Enter infix expression3+2

Given infix expression is:

3+2

Conversion of infix to postfix

Postfix expression is:32+

Evaluation of postfix expression is

result is=5