**INPUT**

**Postfix Evaluation**

#include <iostream>

#include<math.h>

using namespace std;

int top=-1;

int stk[20];

void push(int x)

{

if(top<20)

{

stk[++top]=x;

}

else

{

cout<<"\nStack is full";

}

}

int pop(int y)

{

int x;

x=stk[y];

top--;

return x;

}

int main()

{

cout << "Postfix evaluation!!!\n";

char arr[20];

char \*p=arr;

cout<<"\nEnter Postfix Expression : ";

cin>>arr;

int temp;

while(\*p!='\0')

{

if(isalnum(\*p))

{

push(\*p-48);

}

else if(\*p=='+')

{

temp=pop(top)+pop(top);

push(temp);

}

else if(\*p=='-')

{

temp=pop(top-1)-pop(top+1);

push(temp);

}

else if(\*p=='\*')

{

temp=pop(top)\*pop(top);

push(temp);

}

else if(\*p=='/')

{

temp=pop(top-1)/pop(top+1);

push(temp);

}

else if(\*p=='^')

{

temp=pow(pop(top-1),pop(top));

push(temp);

}

p++;

}

while(top!=-1)

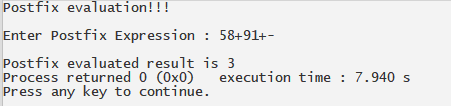
{

cout<<"\nPostfix evaluated result is "<<stk[top]; top--;

}

}

**OUTPUT**



**INPUT**

**INFIX TO POSTFIX CONVERTION**

#include <iostream>

#include<algorithm>

#include<stack>

#include<math.h>

using namespace std;

string infixToPostfix(string infix);

int isoperand(char symbol)

{

if(symbol>='a'&&symbol<='z')

return 1;

else

return 0; }

int isoperator(char symbol)

{

if(symbol=='+'||symbol=='-'||symbol=='\*'||symbol=='/'||symbol=='^')

return 1;

else

return 0; }

int isprecedence(char symbol)

{

int result=0;

switch(symbol)

{

case '{':

result=0;

break;

case '+':

case '-':

result=1;

break;

case '\*':

case '/':

result=2;

break;

case '^':

result=3;

break; }

return result; }

string infixToPostfix(string infix)

{

stack<char> s;

char smbl;

string postfix;

s.push('(');

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

{

smbl=infix[i];

if(isoperand(smbl))

{

postfix += smbl; }

if(smbl=='(')

{

s.push(smbl); }

if(isoperator(smbl))

{

while(isprecedence(s.top())>=isprecedence(smbl))

{

postfix += s.top();

s.pop(); }

s.push(smbl); }

if(smbl==')')

{

while(s.top()!='(')

{

postfix += s.top();

s.pop(); }

s.pop(); } }

while(s.top()!='(')

{

postfix += s.top();

s.pop(); }

return postfix; }

int main()

{

cout<<"Infix to postfix conversion !!!\n";

cout<<"\nEnter an infix expression : ";

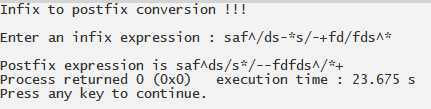
string infix;

cin>>infix;

cout<<"\nPostfix expression is "<<infixToPostfix(infix);

}

**OUTPUT**



**INPUT**

**PREFIX EVALUATION**

#include <iostream>

#include<math.h>

using namespace std;

int top1=-1;

class stack {

public:

int stk[20];

void push(int x) {

if(top1<20) {

stk[++top1]=x; }

else {

cout<<"\nStack is full"; } }

int pop(int y) {

int x;

x=stk[y];

top1--;

return x; }

int top()

{

return stk[top1]; } };

bool isOperand(char c) {

return isdigit(c); }

int evaluatePrefix(string exprsn) {

stack Stack;

for (int j = exprsn.size() - 1; j >= 0; j--) {

if (isOperand(exprsn[j]))

Stack.push(exprsn[j] - '0');

else {

int o1 = Stack.top();

Stack.pop(exprsn[j]);

int o2 = Stack.top();

Stack.pop(exprsn[j]);

switch (exprsn[j]) {

case '+':

Stack.push(o1 + o2);

break;

case '-':

Stack.push(o1 - o2);

break;

case '\*':

Stack.push(o1 \* o2);

break;

case '/':

Stack.push(o1 / o2);

break;

case '^':

Stack.push(pow(o1 , o2));

break; } } }

return Stack.top(); }

int main() {

string exprsn;

cout<<"Prefix evaluation\n";

cout<<"Enter prefix expression : ";

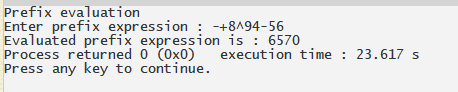
cin>>exprsn;

cout <<"Evaluated prefix expression is : "<<evaluatePrefix(exprsn);

return 0;

}

**OUTPUT**

****