**PUNE INSTITUTE OF COMPUTER TECHNOLOGY DHANKAWADI, PUNE**

**Data Structures And Algorithms(DSA)**

**Assignment No. 02**

**Title : Stack**

**SE-IT-10**  **ACADEMIC YEAR :- 2020-2021**

**Name :- Diptesh Ravindra Varule Roll No :- 23277**

**Source Code :**

//============================================================================

// Name : dsa\_Assignment2.cpp

// Author : Diptesh Varule

// Version : Updating…..

// Copyright : The Cartel

// Description : Hello World in C++, Ansi-style

//============================================================================

**Source Code:**

#include<bits/stdc++.h>

using namespace std;

struct values

{

char c;

int data;

};

template < class t >

class Node

{

t data;

Node <t>\* next;

template < class T >

friend class Stack;

};

template < class t >

class Stack

{

Node <t>\* top;

public:

Stack()

{

top=NULL;

}

void push(t data)

{

Node <t>\* new1 = new Node <t>();

new1->data=data;

if(top==NULL){

top=new1;

}

else{

new1->next=top;

top=new1;

}

}

void pop()

{

Node <t>\* temp = top;

top=temp->next;

delete temp;

temp=NULL;

}

t getTop()

{

return top->data;

}

bool isEmpty()

{

if(top==NULL)

{

return true;

}

else{

return false;

}

}

void display(){

if(isEmpty()){

cout<<"Stack Empty "<<endl;

}

else{

cout<<"Stack is :- ";

Node <t>\* temp=top;

while(temp!=NULL){

cout<<temp->data<<" ";

temp=temp->next;

}

cout<<endl;

}

}

};

class expression

{

string s , Postfix , Prefix;

int cnt\_alpha = 0;

values\* mp;

public:

void input()

{

cout<<"\nEnter Expression :- ";

cin>>s;

}

int precedence(char c)

{

if(c=='^') return 3;

else if(c=='/' || c=='\*') return 2;

else if (c=='+' || c=='-') return 1;

else return 0;

}

void convert\_Postfix()

{

Stack <char> stack;

for(int i=0 ;i<s.size();++i)

{

if(isalpha(s[i]))

{

Postfix+=s[i];

cnt\_alpha+=1;

}

else if(s[i]=='(')

{

stack.push(s[i]);

}

else if(s[i]==')')

{

while(stack.getTop()!='(' && !stack.isEmpty())

{

Postfix+=stack.getTop();

stack.pop();

}

if(stack.getTop()=='(')

{

stack.pop();

}

}

else

{

if(stack.isEmpty()){

stack.push(s[i]);

}

else{

while(!stack.isEmpty() && (precedence(s[i])<= precedence(stack.getTop()))){

Postfix+=stack.getTop();

stack.pop();

}

stack.push(s[i]);

}

}

}

while(!stack.isEmpty())

{

Postfix+=stack.getTop();

stack.pop();

}

cout<<"\nPostfix :- "<<Postfix<<endl;

}

void convert\_Prefix()

{

string temp = "";

for(int i = s.size()-1; i>=0;i--)

{

temp+=s[i];

}

for(int i = 0 ;i<temp.size();++i)

{

if(temp[i]==')') temp[i]='(';

else if(temp[i]=='(') temp[i]=')';

}

string temp2 = "";

Stack <char> stack;

for(int i=0 ;i<temp.size();++i)

{

if(isalpha(temp[i]))

{

temp2+=temp[i];

}

else if(temp[i]=='(')

{

stack.push(temp[i]);

}

else if(temp[i]==')')

{

while(stack.getTop()!='(' && !stack.isEmpty())

{

temp2+=stack.getTop();

stack.pop();

}

if(stack.getTop()=='(')

{

stack.pop();

}

}

else

{

if(stack.isEmpty()){

stack.push(temp[i]);

}

else{

while(!stack.isEmpty() && (precedence(temp[i])< precedence(stack.getTop())))

{

temp2+=stack.getTop();

stack.pop();

}

stack.push(temp[i]);

}

}

}

while(!stack.isEmpty()){

temp2+=stack.getTop();

stack.pop();

}

for(int i = temp2.size()-1;i>=0;--i)

{

Prefix+=temp2[i];

}

cout<<"Prefix :- "<<Prefix<<endl;

}

void input\_values()

{

mp = new values[cnt\_alpha];

int cnt = 0;

for(int i = 0 ;i<s.size();++i)

{

if(isalpha(s[i]))

{

cout<<"Enter Value of "<<s[i]<<" :- ";

int temp;

cin>>temp;

mp[cnt].c = s[i];

mp[cnt].data = temp;

cnt+=1;

}

}

}

int get\_Values(char cc)

{

int info = 0;

for(int i = 0 ; i<cnt\_alpha;++i)

{

if(mp[i].c == cc)

{

info = mp[i].data;

break;

}

}

return info;

}

void evaluate\_Postfix()

{

Stack <int> stack;

for(int i = 0 ;i<Postfix.size(); ++i)

{

if(isalpha(Postfix[i]))

{

stack.push(get\_Values(Postfix[i]));

}

else

{

int a = stack.getTop();

stack.pop();

int b = stack.getTop();

stack.pop();

if(Postfix[i]=='+') stack.push(b+a);

else if(Postfix[i]=='-') stack.push(b-a);

else if(Postfix[i]=='\*') stack.push(b\*a);

else if(Postfix[i]=='/') stack.push(b/a);

else if(Postfix[i]=='^') stack.push(pow(b,a));

}

}

cout<<"Postfix Evaluation :- "<<stack.getTop()<<endl;

}

void evaluate\_Prefix()

{

Stack <int> stack;

for(int i = Prefix.size()-1 ;i>=0; i--)

{

if(isalpha(Prefix[i]))

{

stack.push(get\_Values(Prefix[i]));

}

else

{

int b = stack.getTop();

stack.pop();

int a = stack.getTop();

stack.pop();

if(Prefix[i]=='+') stack.push(b+a);

else if(Prefix[i]=='-') stack.push(b-a);

else if(Prefix[i]=='\*') stack.push(b\*a);

else if(Prefix[i]=='/') stack.push(b/a);

else if(Prefix[i]=='^') stack.push(pow(b,a));

}

}

cout<<"Prefix Evaluation :- "<<stack.getTop()<<endl;

}

};

int main()

{

int choice;

do

{

expression e ;

e.input();

e.convert\_Postfix();

e.convert\_Prefix();

e.input\_values();

e.evaluate\_Postfix();

e.evaluate\_Prefix();

cout<<"Enter 1 to continue:";

cin>>choice;

} while (choice==1);

cout<<"\nYou Quit"<<endl;

return 0;

}

**Output :**

Enter Expression :- (a+b)\*c/d-e

Postfix :- ab+c\*d/e-

Prefix :- -/\*+abcde

Enter Value of a :- 10

Enter Value of b :- 15

Enter Value of c :- 12

Enter Value of d :- 8

Enter Value of e :- 4

Postfix Evaluation :- 33

Prefix Evaluation :- 33

Enter 1 to continue: