Name : Kaushik Nitin Naik

Subject : Data Structures and Algorithms

Roll Number :23325

Batch : F11

Assignment No.2 – (Stack)

Code has three files: **1) Stack.h** **2) stack.cpp** **3) stack_implementation.cpp**

STACK.H

```
/*
 * stack.h

 *
 *  Created on: 08-Oct-2021
 *      Author: kaush
 */

#ifndef STACK_H_
#define STACK_H_
#include <iostream>
using namespace std;
template <class T>
class stack;

template <class T>
class node{
    public:
    T info;
    node <T>*next;
    node(){
       info=0;
       next =NULL;

    }
    friend class stack<T>;

};
template <class T>
class stack{
public:
       node <T>*top;
       bool isempty();
       void push(T ele);
       void pop();
       void display();
       T isTop();
       stack();
};
#endif /* STACK_H_ */
```

STACK.CPP

```
/*
 * stack.cpp
 *
 *  Created on: 08-Oct-2021
 *      Author: kaush
 */
#include <iostream>
#include "stack.h"
using namespace std;
template <class T>
stack<T> :: stack(){
	top=NULL;
}
template <class T>
void stack<T> ::push(T ele){
	node <T>*new1 = new node<T>;
	new1->info=ele;
	if(top==NULL){
		top=new1;
	}
	else{
		new1->next=top;
		top=new1;
	}
	cout<<ele<<" successfully added to the top of the stack"<<endl;

}
template <class T>
void stack<T>::pop(){
	 if(top==NULL){
		 cout<<"Empty Stack!"<<endl;
	   }
	 node<T> * temp = top->next;
	 cout<<top->info<<" successfully popped out of the stack !"<<endl;
	 top->next=NULL;
	 delete top;
	 top=temp;
}

template <class T>
T stack<T>:: isTop(){
	return top->info;
}

template <class T>
bool stack<T>::isempty(){
	if(top==NULL){
		return true;
	}
	return false;
}

template <class T>
void stack<T>::display(){
	 cout<<"Displaying the stack(TOP to BOTTOM)"<<endl;
	 node<T> * current = top;
	 while(current!=NULL)
	 {
		 cout<<current->info<<endl;
	    current = current->next;
	 }
}
```

STACK_IMPLEMENTATION_01.CPP

```
//============================================================================
// Name        : stack_implementation.cpp
// Author      : Kaushik Naik (23325)
// Version     :
// Copyright   : Your copyright notice
// Description : Implementation of Stack
//============================================================================

#include <iostream>
#include <string>
#include <algorithm>
#include <ctype.h>
#include "stack.h"
#include "stack.cpp"
using namespace std;
struct operandAndValues{
	char operand;
	float operandVal;
};
class expression{
public:
    string coversion_Infix_to_prefix(string I);
    string coversion_Infix_to_postfix(string I);
    string revString(string I);
    void eval_prefix(string I);
    void eval_postfix(string I);
    void display();
    int precedence(char c);
    int assoc(char c);
};
int expression :: precedence(char c){
	int p;
	if(c=='+' || c=='-'){
		p= 1; //+- has precedence 1
	}
	else if(c=='*' || c=='/'){
		p= 2; //*/ has precedence 2
	}
	else if(c=='^'){
		p= 3; // ^ has precedence 3
	}
	return p;
}
int expression :: assoc(char c){
	int a;
	if(c=='+' || c=='-' || c=='*' || c=='/'){
		a=0; // 0 indicates L --> R Associativity
	}
	else if(c=='^'){
		a=1; // 1 indicates R--> L Associativity
	}
	return a;
}
```

```
string expression :: revString(string I){
	string rev = "";
	int a = I.length();
	for(int k=a-1;k>=0;k--){
		if(I[k]=='('){
			rev = rev + ')';
		}
		else if(I[k]==')'){
			rev = rev + '(';
		}
		else{
			rev = rev + I[k];
		}
	}
	return rev;
}
string expression :: coversion_Infix_to_postfix(string I){
	stack<char> sc;
	string postfix="";
	for(char ch : I){
		if(ch=='('){
			sc.push(ch);
		}
		else if(ch==')'){
			while(sc.isTop()!='('){
				postfix = postfix+sc.isTop();
				sc.pop();
			}
			sc.pop(); // removing the '('
		}
		else if(ch=='+' || ch=='-' || ch=='*' || ch=='/' || ch =='^'){
			while(sc.isempty()==false && sc.isTop()!='(' &&
(precedence(ch)<= precedence(sc.isTop()))){
				postfix = postfix + sc.isTop();
				sc.pop();
			}
			sc.push(ch);

		}
		else{
			postfix = postfix + ch;
		}
	}
	while(sc.top!=NULL){
		postfix = postfix + sc.isTop();
		sc.top=sc.top->next;
	}
    return postfix;
}
string expression::coversion_Infix_to_prefix(string I){
	string revInput = revString(I);
	string Ans = revString(coversion_Infix_to_postfix(revInput));
	return Ans;
}
```

```
void expression :: eval_postfix(string I){
	//string Res = coversion_Infix_to_postfix(I);
	struct operandAndValues listofoperandvalues[I.length()];
	stack <float> evalAns;
	for(int i=0;i<I.length();i++){
		char ch = I[i];
		listofoperandvalues[i].operand = ch;
		if(ch=='+' || ch=='-' || ch=='*' || ch=='/'){
			float a = evalAns.isTop();
			evalAns.pop();
			float b = evalAns.isTop();
			evalAns.pop();
			float c;
			switch(ch){
			case '+':
			{
				c = a+b;
				break;
			}
			case '-':
			{
				c = b-a;
				break;
			}
			case '*':
			{
				c = b*a;
				break;
			}
			case '/':
				c = b/a;
				break;
			}
			evalAns.push(c);
		}
		else{
			cout<<"Enter Value of ' "<<ch<<" ' =>";
			cin>>listofoperandvalues[i].operandVal;
			evalAns.push(listofoperandvalues[i].operandVal);
		}
	}
	cout<<"ANS : "<<evalAns.isTop()<<endl;

}
```

```
void expression :: eval_prefix(string I){
	string convertedPrefix = revString(I);
	struct operandAndValues listofoperandvalues[I.length()];
	stack <float> evalAns;
		for(int j=0;j<convertedPrefix.length();j++){
			char ch = convertedPrefix[j];
			if(ch=='+' || ch=='-' || ch=='*' || ch=='/'){
				float a = evalAns.isTop();
				evalAns.pop();
				float b = evalAns.isTop();
				evalAns.pop();
				float c;
				switch(ch){
				case '+':
				{
					c = a+b;
					break;
				}
				case '-':
				{
					c = (a-b);
					break;
				}
				case '*':
				{
					c = b*a;
					break;
				}
				case '/':
					c = a/b;
					break;
				}
				evalAns.push(c);
			}
			else{
				cout<<"Enter Value of ' "<<ch<<" ' =>";
				cin>>listofoperandvalues[j].operandVal;
				evalAns.push(listofoperandvalues[j].operandVal);
			}
		}
		cout<<"ANS : "<<evalAns.isTop()<<endl;

}
```

```
int main() {
	stack<int>si;
	expression ex;
	string obtainedPostfix;
	string obtainedPrefix;
	string inputString;
	cout<<"Enter Infix Expression : ";
	cin>>inputString;
	while(true){
		cout<<endl;
		cout<<"--------MENU--------"<<endl;
		cout<<"[1] Infix to Postfix Conversion\n[2] Infix to Prefix Conversion\n[3]
Eval Postfix\n[4] Eval Prefix\n[0] Exit the Program"<<endl;
		int ch;
		cout<<"Enter Command -->";cin>>ch;
		switch(ch){
		case 0:
			cout<<"Program Exited Successfully!"<<endl;
			exit(0);
			break;
		case 1:
			obtainedPostfix = ex.coversion_Infix_to_postfix(inputString);
			cout<<"Required Postfix equation is :
"<<ex.coversion_Infix_to_postfix(inputString)<<endl;
			break;
		case 2:
		{
			obtainedPrefix = ex.coversion_Infix_to_prefix(inputString);
			cout<<"Required Prefix equation is :
"<<ex.coversion_Infix_to_prefix(inputString)<<endl;
			break;

		}
		case 3:
		{
			cout<<"PostFix Equation : "<<obtainedPostfix<<endl;
			ex.eval_postfix(obtainedPostfix);
			break;
		}
		case 4:
		{
			cout<<"Prefix Equation : "<<obtainedPrefix<<endl;
			ex.eval_prefix(obtainedPrefix);
			break;
		}

		}

	}
	return 0;
}
```

INPUT – OUTPUT Screenshots

1) Input: a+b+c

```
Enter Infix Expression : a+b+c

--------MENU--------
[1] Infix to Postfix Conversion
[2] Infix to Prefix Conversion
[3] Eval Postfix
[4] Eval Prefix
[0] Exit the Program
Enter Command -->1
+ successfully added to the top of the stack
+ successfully popped out of the stack !
+ successfully added to the top of the stack
+ successfully added to the top of the stack
+ successfully popped out of the stack !
+ successfully added to the top of the stack
Required Postfix equation is : ab+c+

--------MENU--------
[1] Infix to Postfix Conversion
[2] Infix to Prefix Conversion
[3] Eval Postfix
[4] Eval Prefix
[0] Exit the Program
Enter Command -->2
+ successfully added to the top of the stack
+ successfully popped out of the stack !
+ successfully added to the top of the stack
+ successfully added to the top of the stack
+ successfully popped out of the stack !
+ successfully added to the top of the stack
Required Prefix equation is : +a+bc
```

Figure 2 - Conversion

```
Enter Command -->3
PostFix Equation : ab+c+
Enter Value of ' a ' =>2
2 successfully added to the top of the stack
Enter Value of ' b ' =>3
3 successfully added to the top of the stack
3 successfully popped out of the stack !
2 successfully popped out of the stack !
5 successfully added to the top of the stack
Enter Value of ' c ' =>7
7 successfully added to the top of the stack
7 successfully popped out of the stack !
5 successfully popped out of the stack !
12 successfully added to the top of the stack
ANS : 12

--------MENU--------
[1] Infix to Postfix Conversion
[2] Infix to Prefix Conversion
[3] Eval Postfix
[4] Eval Prefix
[0] Exit the Program
Enter Command -->4
Prefix Equation : +a+bc
Enter Value of ' c ' =>7
7 successfully added to the top of the stack
Enter Value of ' b ' =>3
3 successfully added to the top of the stack
3 successfully popped out of the stack !
7 successfully popped out of the stack !
10 successfully added to the top of the stack
Enter Value of ' a ' =>2
2 successfully added to the top of the stack
2 successfully popped out of the stack !
10 successfully popped out of the stack !
12 successfully added to the top of the stack
ANS : 12
```

Figure 1 - Evaluation

2) Input: a +b*c^e

```
Enter Infix Expression : a+b*c^e

--------MENU--------
[1] Infix to Postfix Conversion
[2] Infix to Prefix Conversion
[3] Eval Postfix
[4] Eval Prefix
[0] Exit the Program
Enter Command -->1
+ successfully added to the top of the stack
* successfully added to the top of the stack
^ successfully added to the top of the stack
+ successfully added to the top of the stack
* successfully added to the top of the stack
^ successfully added to the top of the stack
Required Postfix equation is : abce^*+

--------MENU--------
[1] Infix to Postfix Conversion
[2] Infix to Prefix Conversion
[3] Eval Postfix
[4] Eval Prefix
[0] Exit the Program
Enter Command -->2
^ successfully added to the top of the stack
^ successfully popped out of the stack !
* successfully added to the top of the stack
* successfully popped out of the stack !
+ successfully added to the top of the stack
^ successfully added to the top of the stack
^ successfully popped out of the stack !
* successfully added to the top of the stack
* successfully popped out of the stack !
+ successfully added to the top of the stack
Required Prefix equation is : +a*b^ce
```

Figure 5-Conversion

```
--------MENU--------
[1] Infix to Postfix Conversion
[2] Infix to Prefix Conversion
[3] Eval Postfix
[4] Eval Prefix
[0] Exit the Program
Enter Command -->3
PostFix Equation : abce^*+
Enter Value of ' a ' =>3
3 successfully added to the top of the stack
Enter Value of ' b ' =>6
6 successfully added to the top of the stack
Enter Value of ' c ' =>2
2 successfully added to the top of the stack
Enter Value of ' e ' =>3
3 successfully added to the top of the stack
3 successfully popped out of the stack !
2 successfully popped out of the stack !
8 successfully added to the top of the stack
8 successfully popped out of the stack !
6 successfully popped out of the stack !
48 successfully added to the top of the stack
48 successfully popped out of the stack !
3 successfully popped out of the stack !
51 successfully added to the top of the stack
ANS : 51
```

Figure 4- Postfix Evaluation

```
[1] Infix to Postfix Conversion
[2] Infix to Prefix Conversion
[3] Eval Postfix
[4] Eval Prefix
[0] Exit the Program
Enter Command -->4
Prefix Equation : +a*b^ce
Enter Value of ' e ' =>3
3 successfully added to the top of the stack
Enter Value of ' c ' =>2
2 successfully added to the top of the stack
2 successfully popped out of the stack !
3 successfully popped out of the stack !
8 successfully added to the top of the stack
Enter Value of ' b ' =>6
6 successfully added to the top of the stack
6 successfully popped out of the stack !
8 successfully popped out of the stack !
48 successfully added to the top of the stack
Enter Value of ' a ' =>3
3 successfully added to the top of the stack
3 successfully popped out of the stack !
48 successfully popped out of the stack !
51 successfully added to the top of the stack
ANS : 51
```

Figure 3- Prefix Evaluation

3) Input: a*b/c

```
Enter Infix Expression : a*b/c

--------MENU--------
[1] Infix to Postfix Conversion
[2] Infix to Prefix Conversion
[3] Eval Postfix
[4] Eval Prefix
[0] Exit the Program
Enter Command -->1
* successfully added to the top of the stack
* successfully popped out of the stack !
/ successfully added to the top of the stack
* successfully added to the top of the stack
* successfully popped out of the stack !
/ successfully added to the top of the stack
Required Postfix equation is : ab*c/

--------MENU--------
[1] Infix to Postfix Conversion
[2] Infix to Prefix Conversion
[3] Eval Postfix
[4] Eval Prefix
[0] Exit the Program
Enter Command -->2
/ successfully added to the top of the stack
/ successfully popped out of the stack !
* successfully added to the top of the stack
/ successfully added to the top of the stack
/ successfully popped out of the stack !
* successfully added to the top of the stack
Required Prefix equation is : *a/bc
```

Figure 7-Conversion

```
Enter Command -->3
PostFix Equation : ab*c/
Enter Value of ' a ' =>3
3 successfully added to the top of the stack
Enter Value of ' b ' =>8
8 successfully added to the top of the stack
8 successfully popped out of the stack !
3 successfully popped out of the stack !
24 successfully added to the top of the stack
Enter Value of ' c ' =>2
2 successfully added to the top of the stack
2 successfully popped out of the stack !
24 successfully popped out of the stack !
12 successfully added to the top of the stack
ANS : 12

--------MENU--------
[1] Infix to Postfix Conversion
[2] Infix to Prefix Conversion
[3] Eval Postfix
[4] Eval Prefix
[0] Exit the Program
Enter Command -->4
Prefix Equation : *a/bc
Enter Value of ' c ' =>2
2 successfully added to the top of the stack
Enter Value of ' b ' =>8
8 successfully added to the top of the stack
8 successfully popped out of the stack !
2 successfully popped out of the stack !
4 successfully added to the top of the stack
Enter Value of ' a ' =>3
3 successfully added to the top of the stack
3 successfully popped out of the stack !
4 successfully popped out of the stack !
12 successfully added to the top of the stack
ANS : 12
```

Figure 6-Evaluation