

Write a C++ program to demonstrate working of Stack using Array.

Code:

OperationsOnStack.cpp

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>

#define STACKSIZE 5

using namespace std;

class stack{
    int arr[STACKSIZE];
    int stack_top;

    public:
        stack(){
            stack_top = -1;
        }
        void push(int val){
            stack_top = stack_top+1;
            arr[stack_top] = val;
        }
        int pop(){
            int val;
            val = arr[stack_top];
            stack_top--;
            return val;
        }
        bool isEmpty(){
            if(stack_top == -1){
                return true;
            }else{
                return false;
            }
        }
        bool isFull(){
            if(stack_top == STACKSIZE-1){
                return true;
            }else{
                return false;
            }
        }
        int size(){
            return stack_top+1;
        }
        void display(){
            if(stack_top == -1){
                cout << "Not any Element present in the Stack";
            }
            else{
                cout << "Elements in the Stack is: ";
                for(int i = 0 ; i <= stack_top; i++){
                    cout << arr[i] << " ";
                }
                cout << endl;
            }
        }
};

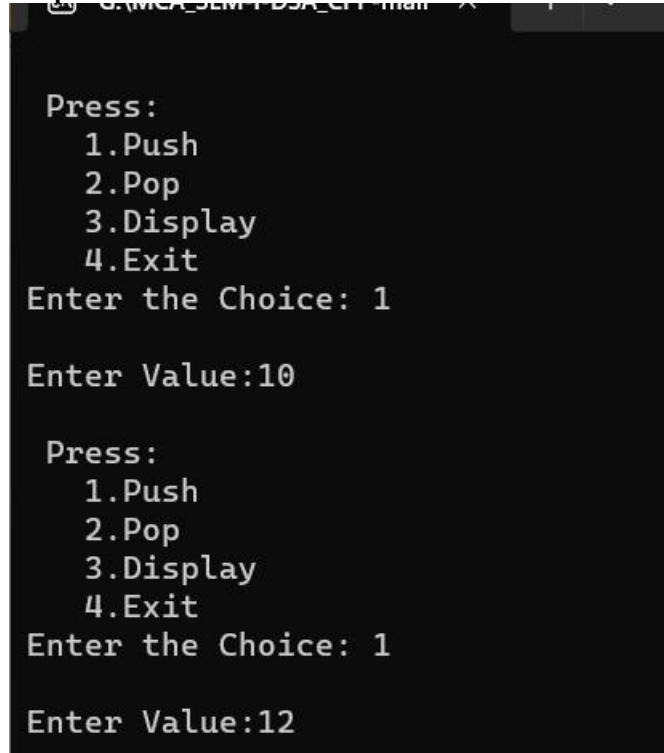
int main(){
    stack myStack;
    int val;
    int choice;
    while(1){
        cout << "\n Press:\n  1.Push\n  2.Pop\n  3.Display\n  4.Exit";
```

```

        cout << "\nEnter the Choice: ";
        cin >> choice;
        switch(choice){
            case 1:
                if(myStack.isFull() == false){
                    cout << "\nEnter Value:";
                    cin >> val;
                    myStack.push(val);
                }else{
                    cout << "\nStack is Full.";
                }
                break;
            case 2:
                if(myStack.isEmpty() == false){
                    val = myStack.pop();
                    cout << "\nValue is " << val;
                }
                else{
                    cout << "\nStack is Empty.";
                }
                break;
            case 3:
                myStack.display();
                break;
            case 4:
                exit(1);
        }
    }
    return 0;
}

```

Output:



```

C:\Users\user\Desktop\Stack\Stack.cpp
Press:
1.Push
2.Pop
3.Display
4.Exit
Enter the Choice: 1
Enter Value:10

Press:
1.Push
2.Pop
3.Display
4.Exit
Enter the Choice: 1
Enter Value:12

```

```
Press:
  1.Push
  2.Pop
  3.Display
  4.Exit
Enter the Choice: 1
```

```
Enter Value:14
```

```
Press:
  1.Push
  2.Pop
  3.Display
  4.Exit
Enter the Choice: 1
```

```
Enter Value:1
```

```
Press:
  1.Push
  2.Pop
  3.Display
  4.Exit
Enter the Choice: 16
```

```
Press:
  1.Push
  2.Pop
  3.Display
  4.Exit
Enter the Choice: 1
```

```
Enter Value:18
```

```
Press:
  1.Push
  2.Pop
  3.Display
  4.Exit
Enter the Choice: 1
```

```
Stack is Full.
```

```
Press:
  1.Push
  2.Pop
  3.Display
  4.Exit
Enter the Choice: 20
```

Stack is Full.

Press:

- 1.Push
- 2.Pop
- 3.Display
- 4.Exit

Enter the Choice: 20

Press:

- 1.Push
- 2.Pop
- 3.Display
- 4.Exit

Enter the Choice: 1

Stack is Full.

Press:

- 1.Push
- 2.Pop
- 3.Display
- 4.Exit

Enter the Choice: |

Press:

- 1.Push
- 2.Pop
- 3.Display
- 4.Exit

Enter the Choice: 2

Value is 18

Press:

- 1.Push
- 2.Pop
- 3.Display
- 4.Exit

Enter the Choice: 3

Elements in the Stack is: 10 12 14 1

Press:

- 1.Push
- 2.Pop
- 3.Display
- 4.Exit

Enter the Choice: |

Write a C++ program to demonstrate working of Stack using Linked List.

Code:

```
#include <bits/stdc++.h>
#include<iostream>
#include<conio.h>
#include<stdlib.h>
using namespace std;

class Node {
public:
    int data;
    Node* link;

    Node(int n)
    {
        this->data = n;
        this->link = NULL;
    }
};

class Stack {
    Node* top;

public:
    Stack() { top = NULL; }

    void push(int data)
    {
        Node* temp = new Node(data);

        if (!temp) {
            cout << "\nStack Overflow";
            exit(1);
        }

        temp->data = data;
        temp->link = top;
        top = temp;
    }

    bool isEmpty()
    {
        return top == NULL;
    }

    int peek()
    {
        if (!isEmpty())
            return top->data;
        else
            exit(1);
    }

    void pop()
    {
        Node* temp;

        if (top == NULL) {
            cout << "\nStack Underflow" << endl;
            exit(1);
        }
        else {
            temp = top;
            top = top->link;
            free(temp);
        }
    }
}
```

```

void display()
{
    Node* temp;

    if (top == NULL) {
        cout << "\nStack Underflow";
        exit(1);
    }
    else {
        temp = top;
        while (temp != NULL) {
            cout << temp->data;
            temp = temp->link;
            if (temp != NULL)
                cout << " -> ";
        }
    }
};

int main()
{
    Stack s;

    s.push(11);
    s.push(22);
    s.push(33);
    s.push(44);
    s.display();

    cout << "\nTop element is " << s.peek() << endl;

    s.pop();
    s.pop();

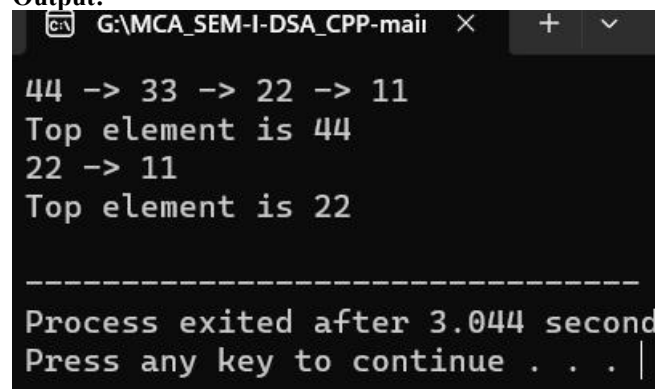
    s.display();

    cout << "\nTop element is " << s.peek() << endl;

    return 0;
}

```

Output:



```

G:\MCA_SEM-I-DSA_CPP-main
44 -> 33 -> 22 -> 11
Top element is 44
22 -> 11
Top element is 22

-----
Process exited after 3.044 seconds
Press any key to continue . . .

```

Demonstrate application of stack “evaluation of postfix expression”.

Code:

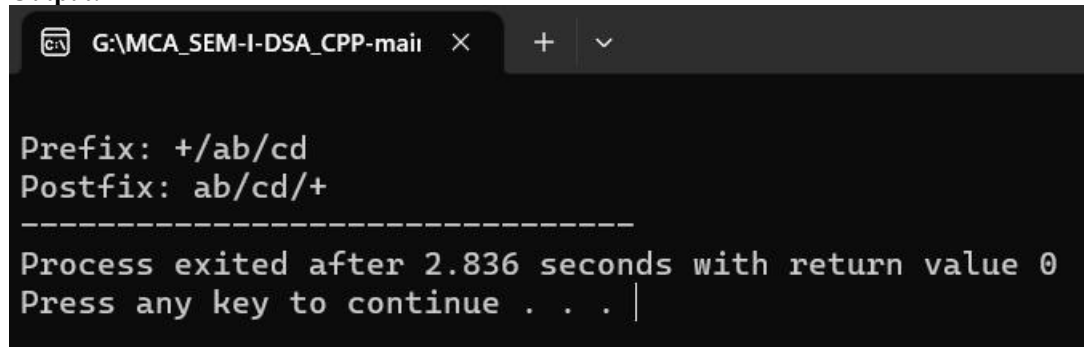
```
#include<iostream>
#include<stack>
using namespace std;
bool isOperator(char x){
    switch(x){
        case '+':
        case '-':
        case '*':
        case '/':
            return true;
    }
    return false;
}

string prefixConvertToPostfix(string prefix){
    stack <string> expression;
    int length = prefix.size();

    for(int i = length-1; i >= 0; i--){
        if(isOperator(prefix[i])){
            string op1 = expression.top();
            expression.pop();
            string op2 = expression.top();
            expression.pop();
            string temp = op1 + op2 + prefix[i];
            expression.push(temp);
        }
        else{
            expression.push(string(1,prefix[i]));
        }
    }
    return expression.top();
}

int main(){
    string prefix = "+/ab/cd";
    cout << "\nPrefix: " << prefix;
    cout << "\nPostfix: " << prefixConvertToPostfix(prefix);
    return 0;
}
```

Output:



```
G:\MCA_SEM-I-DSA_CPP-main  ×  +  ∨

Prefix: +/ab/cd
Postfix: ab/cd/+
-----
Process exited after 2.836 seconds with return value 0
Press any key to continue . . . |
```

Demonstrate application of stack “balancing of parenthesis”.

Code:

```
#include<iostream>
#include<conio.h>
#include<stdlib.h>

#define STACKSIZE 5

using namespace std;

class stack{
    int arr[STACKSIZE];
    int stack_top;

public:
    stack(){
        stack_top = -1;
    }
    void push(int val){
        stack_top = stack_top+1;
        arr[stack_top] = val;
    }
    int pop(){
        int val;
        val = arr[stack_top];
        stack_top--;
        return val;
    }
    bool isEmpty(){
        if(stack_top == -1){
            return true;
        }else{
            return false;
        }
    }
    bool isFull(){
        if(stack_top == STACKSIZE-1){
            return true;
        }else{
            return false;
        }
    }
    int size(){
        return stack_top+1;
    }
    void display(){
        if(stack_top == -1){
            cout << "Not any Element present in the Stack";
        }
        else{
            cout << "Elements in the Stack is: ";
            for(int i = 0 ; i <= stack_top; i++){
                cout << arr[i] << " ";
            }
            cout << endl;
        }
    }
};

int main(){
    stack myStack;
    char exp[] = {'<','{','[','(',')',']','}', '>','\0'};
    int size = 0;
    int op1, op2, res, i;
    int isValid = 1;
    char test;
    for(i = 0; exp[i] != '\0'; i++){
        test = exp[i];
```



```

char comp;
if(test == '{' || test == '<' || test == '[' || test == '('){
    myStack.push(test);
}
else{
    if(myStack.isEmpty() == true){
        isValid = 0;
        break;
    }
    else{
        comp = myStack.pop();
        if(test == '}' && comp != '{'){
            isValid = 0;
            break;
        }
        if(test == '>' && comp != '<'){
            isValid = 0;
            break;
        }
        if(test == ']' && comp != '['){
            isValid = 0;
            break;
        }
        if(test == ')' && comp != '('){
            isValid = 0;
            break;
        }
    }
}
}
if(isValid == 1 && myStack.isEmpty() == true){
    cout << "\nValid Parenthesis!!!";
}
else{
    cout << "\nNot valid Parenthesis!!!";
}
return 0;
}

```

Output:

```

G:\MCA_SEM-I-DSA_CPP-mai  X  +  v
Not valid Parenthesis!!!
-----
Process exited after 2.324 seconds with return value 0
Press any key to continue . . . |

```

```

G:\MCA_SEM-I-DSA_CPP-mai  X  +  v
Valid Parenthesis!!!
-----
Process exited after 1.2 seconds with return value 0
Press any key to continue . . . |

```