LAB 9

CSE225L

Stack (Array—Based)

In this lab, we will:

- Design and implement the Stack ADT using an array-based structure.
- Create the **StackType** class with methods for Push, Pop, Top, and status checks (IsFull, IsEmpty).
- Test the stack by performing operations such as inserting, removing, and retrieving elements, and handling stack overflow and underflow conditions.
- Use the stack to validate balanced parentheses in input strings.

STACK (ARRAY–BASED)

stacktype.h

```
#ifndef STACKTYPE_H
#define STACKTYPE_H

const int SIZE = 5;

// Exception class thrown by Push when the stack is full
class FullStack {};

// Exception class thrown by Pop and Top when the stack is empty
class EmptyStack {};

template<class T>
class StackType
{
private:
    T* data;
    int top;
public:
    StackType();
    ~StackType();
    bool IsFull();
    bool IsEmpty();
    void Push(T);
    void Pop();
    T Top();
};

#endif // STACKTYPE_H
```

stacktype.cpp

```
#include <iostream>
#include "stacktype.h"

using namespace std;

template<class T>
StackType<T>::StackType()
{
    data = new T[SIZE];
    top = -1;
}

template<class T>
StackType<T>::~StackType()
{
    delete[] data;
}

template<class T>
bool StackType<T>::IsEmpty()
{
    return (top == -1);
}
```

```
template<class T>
bool StackType<T>::IsFull()
{
    return (top == SIZE - 1);
}

template<class T>
void StackType<T>::Push(T value)
{
    try
    {
        if (IsFull())
            throw FullStack();
        else
        {
            top++;
            data[top] = value;
        }
    }
    catch (FullStack e)
    {
        cout << "Error: Stack is full" << endl;
    }
}

template<class T>
void StackType<T>::Pop()
{
    try
    {
        if (IsEmpty())
            throw EmptyStack();
        else
            top--;
    }
    catch (EmptyStack e)
    {
        cout << "Error: Stack is empty" << endl;
    }
}

template<class T>
T StackType<T>::Top()
{
    try
    {
        if (IsEmpty())
            throw EmptyStack();
        else
            return data[top];
    }
    catch (EmptyStack e)
    {
        cout << "Error: Stack is empty" << endl;
    }
}
```

STACK (ARRAY—BASED)

TASKS:

Instructions:

- Create the driver file (main.cpp) and perform the following tasks.
- You cannot make any changes to the header (**.h**) or source (**.cpp**) files of the **StackType** class.

Operation	Input Values	Expected Output
Create a stack of integers		
Check if the stack is empty		Stack is Empty
Push four items	5, 7, 4, 2	
Check if the stack is empty		Stack is not Empty
Check if the stack is full		Stack is not full
Print the values in the stack (in the order the values are given)		5, 7, 4, 2
Push another item	3	
Print the values in the stack		5, 7, 4, 2, 3
Check if the stack is full		Stack is full
Pop two items		
Print top item		4
Take strings of parentheses as input from the user and use a stack to check if each string is balanced.	( )	Balanced
	(( )) ( ) (( ) ( )) ( )	Balanced
	(( )) ( ) ((( )	Not Balanced
	(( )))) ((( )	Not Balanced
	(( ))))))))	Not Balanced