

|  |
| --- |
| **LAB 9** of DSA LAB |

**Submitted by:**

**Name:** Muhammad shuraim

**Roll no:** SU92-S24-BSSEM-103

**Section:** 3A

**Submitted to**

Sir Rasikh Ali

**Question 1**

**STACK Using Array**

#include <iostream>

using namespace std;

class StackArray {

private:

int top;

int arr[100];

int size;

public:

StackArray(int s = 100) {

top = -1;

size = s;

}

void push(int val) {

if (top == size - 1) {

cout << "Stack Overflow\n";

return;

}

arr[++top] = val;

}

void pop() {

if (top == -1) {

cout << "Stack Underflow\n";

return;

}

top--;

}

void display() {

if (top == -1) {

cout << "Stack is empty\n";

return;

}

cout << "Stack (Top -> Bottom): ";

for (int i = top; i >= 0; i--) {

cout << arr[i] << " ";

}

cout << "\n";

}

};

int main() {

StackArray sa;

sa.push(10);

sa.push(20);

sa.push(30);

sa.display();

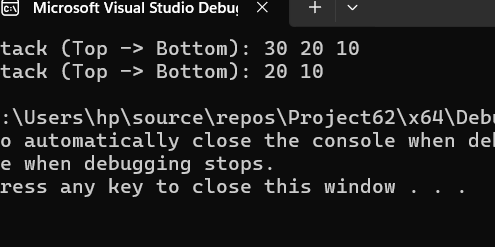
sa.pop();

sa.display();

return 0;

}

**OUTPUT**



STACK USING LINKLIST

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int val) {

data = val;

next = nullptr;

}

};

class StackLinkedList {

private:

Node\* top;

public:

StackLinkedList() {

top = nullptr;

}

void push(int val) {

Node\* newNode = new Node(val);

newNode->next = top;

top = newNode;

}

void pop() {

if (top == nullptr) {

cout << "Stack Underflow\n";

return;

}

Node\* temp = top;

top = top->next;

delete temp;

}

void display() {

if (top == nullptr) {

cout << "Stack is empty\n";

return;

}

Node\* temp = top;

cout << "Stack (Top -> Bottom): ";

while (temp != nullptr) {

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

temp = temp->next;

}

cout << "\n";

}

};

int main() {

StackLinkedList sl;

sl.push(100);

sl.push(200);

sl.push(300);

sl.display();

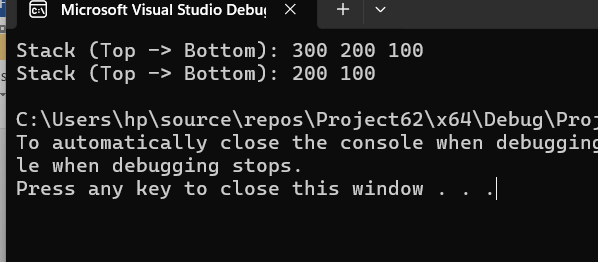
sl.pop();

sl.display();

return 0;

}

Output



EXPLANATION OF BOTH

✅ Stack Using Array

* Uses a fixed-size array (e.g., int arr[100]).
* top keeps track of the last inserted element.
* push() adds element at top + 1.
* pop() removes the element at top.
* display() shows elements from top to bottom.
* Limitation: Fixed size (can overflow).

✅ Stack Using Linked List

* Uses nodes (each has data and next pointer).
* top is a pointer to the latest node.
* push() creates a new node and links it at the top.
* pop() deletes the top node and moves top to the next.
* display() prints from top to end.
* Advantage: No fixed size (uses dynamic memory).