THE UNIVERSITY OF AZAD JAMMU AND KASHMIR, MUZAFFARABAD

Department of Software Engineering

Submitted to:	Engr. Sidra Rafique
Course Title:	Data Structure and Algorithm
Course Code:	CS-2101
Session:	2024-2028
Lab No:	07
Roll No:	2024-SE-15
Submitted By:	Shahzad Ahmed Awan
Submission date:	January 8, 2026

Table of Contents

Table of Figures

Lab 07 – Stack Implementation

1. Objective

To implement Stack data structure using:

1. **Array**
2. **Linked List**

Perform basic stack operations (push, pop, peek, display) demonstrating the LIFO principle.

2. Background

- Stack is a LIFO (Last-In-First-Out) data structure.
- Array-based stack has a fixed size.
- Linked list-based stack is dynamic in size.

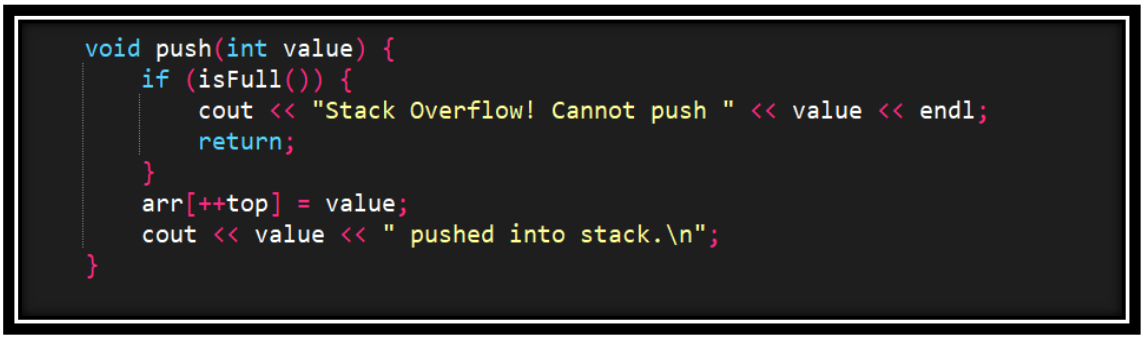
3. Stack Using Array

3.1 Algorithm

- Initialize an array and a top pointer.
- Push elements at top, pop from top.
- Peek to view top element without removing.
- Display elements from top to bottom.

3.2 Functions code when array implementation (file attached)

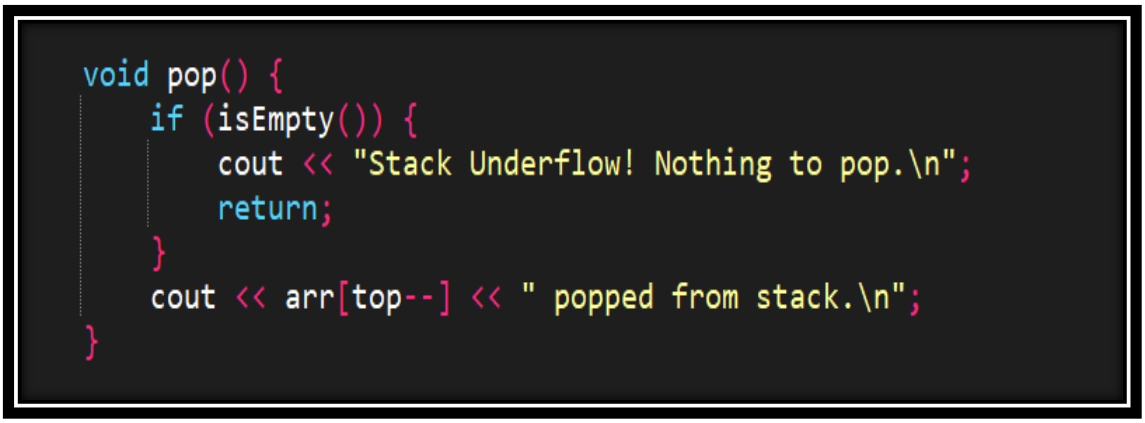
- **Push Function** – Adds an element at the top of the stack.



```
void push(int value) {
    if (isFull()) {
        cout << "Stack Overflow! Cannot push " << value << endl;
        return;
    }
    arr[++top] = value;
    cout << value << " pushed into stack.\n";
}
```

Figure 1: Push function in Array implementation of stack

- **Pop Function** – Removes the top element from the stack.



```
void pop() {
    if (isEmpty()) {
        cout << "Stack Underflow! Nothing to pop.\n";
        return;
    }
    cout << arr[top--] << " popped from stack.\n";
}
```

Figure 2:: Pop function in Array implementation of stack

- **Peek Function** – Returns the top element without removing it.

```
int peek() {
    if (isEmpty()) return -1;
    return arr[top];
}
```

Figure 3: Peek function in Array implementation of stack

- **Display Function** – Displays all elements from top to bottom.

```
void display() {
    if (isEmpty()) {
        cout << "Stack is empty.\n";
        return;
    }
    cout << "Stack elements (top -> bottom): ";
    for (int i = top; i >= 0; i--) cout << arr[i] << " ";
    cout << endl;
}
```

Figure 4 : Push function in Array implementation of stack

---

4. Stack Using Linked List

4.1 Algorithm

- Initialize top pointer as nullptr.
- Push elements by creating a new node at top.
- Pop elements by removing the top node.
- Peek to view top element.
- Display elements from top to bottom.

4.2 Functions when using LinkedList implementation

- **Push Function** – Adds an element at the top of the stack.

```
void push(int value) {
    Node* temp = new Node();
    temp->data = value;
    temp->next = top;
    top = temp;
    cout << value << " pushed into stack.\n";
}
```

Figure 5:: Push function in LinkedList implementation of stack

- **Pop Function** – Removes the top element from the stack.

```
void pop() {
    if (isEmpty()) {
        cout << "Stack Underflow! Nothing to pop.\n";
        return;
    }
    Node* temp = top;
    cout << top->data << " popped from stack.\n";
    top = top->next;
    delete temp;
}
```

Figure 6: Pop function in LinkedList implementation of stack

- **Peek Function** – Returns the top element without removing it.

```
int peek() {
    if (isEmpty()) return -1;
    return top->data;
}
```

Figure 7: Peek function in LinkedList implementation of stack

- **Display Function** – Displays all elements from top to bottom.

```
void display() {
    if (isEmpty()) {
        cout << "Stack is empty.\n";
        return;
    }
    cout << "Stack elements (top -> bottom): ";
    Node* temp = top;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}
```

Figure 8: Display function in LinkedList implementation of stack

5. Conclusion

- Successfully implemented stack using Array and Linked List.
- Demonstrated LIFO principle and dynamic/static stack behavior.
- Stack operations are working correctly with proper interactive output.