**✅ C++ Program: Queue Implementation using Array**

#include <iostream>

using namespace std;

#define SIZE 5 // maximum size of the queue

class Queue {

private:

int items[SIZE];

int front, rear;

public:

Queue() {

front = -1;

rear = -1;

}

// Enqueue operation

void enqueue(int element) {

if (rear == SIZE - 1) {

cout << "Queue is full (Overflow)!" << endl;

return;

}

if (front == -1)

front = 0;

rear++;

items[rear] = element;

cout << "Inserted -> " << element << endl;

}

// Dequeue operation

void dequeue() {

if (front == -1 || front > rear) {

cout << "Queue is empty (Underflow)!" << endl;

return;

}

cout << "Deleted -> " << items[front] << endl;

front++;

}

// Display operation

void display() {

if (front == -1 || front > rear) {

cout << "Queue is empty!" << endl;

return;

}

cout << "Queue elements: ";

for (int i = front; i <= rear; i++) {

cout << items[i] << " ";

}

cout << endl;

}

};

// Main function to test the queue

int main() {

Queue q;

q.enqueue(10);

q.enqueue(20);

q.enqueue(30);

q.display();

q.dequeue();

q.display();

q.enqueue(40);

q.enqueue(50);

q.enqueue(60); // Should show overflow

q.display();

return 0;

}

**✅ C++ Program: Queue Implementation using Linked List**

#include <iostream>

using namespace std;

// Node structure

struct Node {

int data;

Node\* next;

};

// Queue class

class Queue {

private:

Node\* front;

Node\* rear;

public:

Queue() {

front = rear = nullptr;

}

// Enqueue operation

void enqueue(int value) {

Node\* newNode = new Node();

newNode->data = value;

newNode->next = nullptr;

if (rear == nullptr) {

front = rear = newNode;

} else {

rear->next = newNode;

rear = newNode;

}

cout << "Inserted -> " << value << endl;

}

// Dequeue operation

void dequeue() {

if (front == nullptr) {

cout << "Queue is empty (Underflow)!" << endl;

return;

}

Node\* temp = front;

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

front = front->next;

// If the queue becomes empty

if (front == nullptr) {

rear = nullptr;

}

delete temp;

}

// Display operation

void display() {

if (front == nullptr) {

cout << "Queue is empty!" << endl;

return;

}

Node\* temp = front;

cout << "Queue elements: ";

while (temp != nullptr) {

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

temp = temp->next;

}

cout << endl;

}

// Destructor to free memory

~Queue() {

while (front != nullptr) {

dequeue();

}

}

};

// Main function

int main() {

Queue q;

q.enqueue(5);

q.enqueue(10);

q.enqueue(15);

q.display();

q.dequeue();

q.display();

q.enqueue(20);

q.enqueue(25);

q.display();

return 0;

}