

|  |
| --- |
| **LAB 11** of DSA LAB |

**Submitted by:**

**Name:** Muhammad shuraim

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

**Section:** 3A

**Submitted to**

Sir Rasikh Ali

**Question 1**

**QUEUE Using Array**

#include <iostream>

using namespace std;

class QueueArray {

private:

int front, rear;

int arr[100];

int size;

public:

QueueArray(int s = 100) {

size = s;

front = rear = -1;

}

void enqueue(int val) {

if (rear == size - 1) {

cout << "Queue Overflow\n";

return;

}

if (front == -1) front = 0;

arr[++rear] = val;

}

void dequeue() {

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

cout << "Queue Underflow\n";

return;

}

front++;

}

void display() {

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

cout << "Queue is empty\n";

return;

}

cout << "Queue: ";

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

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

}

cout << "\n";

}

};

int main() {

QueueArray qa;

qa.enqueue(10);

qa.enqueue(20);

qa.enqueue(30);

qa.display();

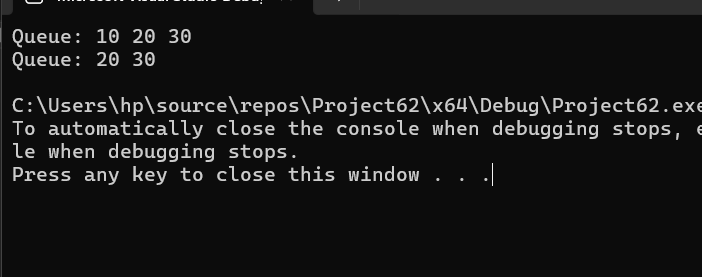
qa.dequeue();

qa.display();

return 0;

}

**OUTPUT**

****

QUEUE USING LINKLIST

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int val) {

data = val;

next = nullptr;

}

};

class QueueLinkedList {

private:

Node\* front;

Node\* rear;

public:

QueueLinkedList() {

front = rear = nullptr;

}

void enqueue(int val) {

Node\* newNode = new Node(val);

if (rear == nullptr) {

front = rear = newNode;

}

else {

rear->next = newNode;

rear = newNode;

}

}

void dequeue() {

if (front == nullptr) {

cout << "Queue Underflow\n";

return;

}

Node\* temp = front;

front = front->next;

if (front == nullptr) rear = nullptr;

delete temp;

}

void display() {

if (front == nullptr) {

cout << "Queue is empty\n";

return;

}

cout << "Queue: ";

Node\* temp = front;

while (temp != nullptr) {

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

temp = temp->next;

}

cout << "\n";

}

};

int main() {

QueueLinkedList ql;

ql.enqueue(100);

ql.enqueue(200);

ql.enqueue(300);

ql.display();

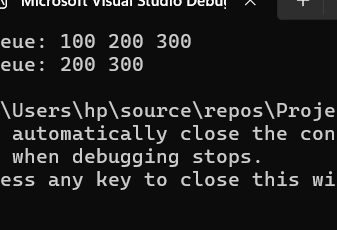
ql.dequeue();

ql.display();

return 0;

}

Output



EXPLANATION OF BOTH

* ✅ Queue Using Array
* Uses a fixed-size array arr[100].
* front points to the first element.
* rear points to the last element.
* enqueue() adds at rear + 1.
* dequeue() removes from front by incrementing it.
* Limitation: Fixed size, no reuse of space unless circular queue is used.
* ✅ Queue Using Linked List
* Uses nodes with data and next.
* front points to the first node.
* rear points to the last node.
* enqueue() creates a new node at the end.
* dequeue() deletes the node at the front.
* Advantage: Dynamic size (no overflow unless memory full).