Aim-implementing queue using linked list

#include <iostream>

using namespace std;

// Node structure for linked list

struct Node {

int data;

Node\* next;

Node(int value) : data(value), next(nullptr) {}

};

class Queue {

private:

Node\* front;

Node\* rear;

public:

Queue() : front(nullptr), rear(nullptr) {}

~Queue() {

while (!isEmpty()) {

dequeue();

}

}

bool isEmpty() {

return front == nullptr;

}

void enqueue(int value) {

Node\* newNode = new Node(value);

if (rear == nullptr) {

front = rear = newNode;

} else {

rear->next = newNode;

rear = newNode;

}

cout << value << " enqueued successfully.\n";

}

void dequeue() {

if (isEmpty()) {

cout << "Queue is empty. Cannot dequeue.\n";

return;

}

Node\* temp = front;

cout << front->data << " dequeued successfully.\n";

front = front->next;

if (front == nullptr) {

rear = nullptr;

}

delete temp;

}

void display() {

if (isEmpty()) {

cout << "Queue is empty.\n";

return;

}

cout << "Queue elements: ";

Node\* temp = front;

while (temp != nullptr) {

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

temp = temp->next;

}

cout << "\n";

}

};

int main() {

Queue q;

int choice, value;

do {

cout << "\n1. Enqueue\n2. Dequeue\n3. Check if Empty\n4. Display\n5. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to enqueue: ";

cin >> value;

q.enqueue(value);

break;

case 2:

q.dequeue();

break;

case 3:

if (q.isEmpty())

cout << "Queue is empty.\n";

else

cout << "Queue is not empty.\n";

break;

case 4:

q.display();

break;

case 5:

cout << "Exiting program.\n";

break;

default:

cout << "Invalid choice. Try again.\n";

}

} while (choice != 5);

return 0;

}