Aim-circular queue implementation using array

#include <iostream>

using namespace std;

class Queue {

private:

int\* arr;

int front;

int rear;

int capacity;

int count;

public:

// Constructor

Queue(int size) {

arr = new int[size];

capacity = size;

front = 0;

rear = -1;

count = 0;

}

// Destructor

~Queue() {

delete[] arr;

}

// Enqueue operation

void enqueue(int value) {

if (isFull()) {

cout << "Queue is full. Cannot enqueue " << value << ".\n";

return;

}

rear = (rear + 1) % capacity;

arr[rear] = value;

count++;

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

}

// Dequeue operation

void dequeue() {

if (isEmpty()) {

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

return;

}

cout << arr[front] << " dequeued successfully.\n";

front = (front + 1) % capacity;

count--;

}

// Check if the queue is empty

bool isEmpty() {

return count == 0;

}

// Check if the queue is full

bool isFull() {

return count == capacity;

}

// Display queue elements

void display() {

if (isEmpty()) {

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

return;

}

cout << "Queue elements: ";

for (int i = 0; i < count; i++) {

cout << arr[(front + i) % capacity] << " ";

}

cout << "\n";

}

};

int main() {

int size;

cout << "Enter the size of the queue: ";

cin >> size;

Queue q(size);

int choice, value;

do {

cout << "\n1. Enqueue\n2. Dequeue\n3. Check if Empty\n4. Check if Full\n5. Display\n6. 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:

if (q.isFull())

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

else

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

break;

case 5:

q.display();

break;

case 6:

cout << "Exiting program.\n";

break;

default:

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

}

} while (choice != 6);

return 0;

}