A double-ended queue (deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a one dimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque.   
  
  
#include <iostream>

using namespace std;

#define MAX 10 // Maximum size of the deque

class Deque {

private:

int arr[MAX]; // Array to store deque elements

int front; // Index of the front of the deque

int rear; // Index of the rear of the deque

public:

// Constructor to initialize deque

Deque() {

front = -1;

rear = -1;

}

// Check if deque is full

bool isFull() {

return (front == 0 && rear == MAX - 1) || (rear + 1 == front);

}

// Check if deque is empty

bool isEmpty() {

return front == -1;

}

// Add element to the front of the deque

void addFront(int value) {

if (isFull()) {

cout << "Deque is full. Cannot add element at the front.\n";

return;

}

if (isEmpty()) {

front = rear = 0;

} else if (front == 0) {

front = MAX - 1;

} else {

front--;

}

arr[front] = value;

cout << "Added " << value << " to the front.\n";

}

// Add element to the rear of the deque

void addRear(int value) {

if (isFull()) {

cout << "Deque is full. Cannot add element at the rear.\n";

return;

}

if (isEmpty()) {

front = rear = 0;

} else if (rear == MAX - 1) {

rear = 0;

} else {

rear++;

}

arr[rear] = value;

cout << "Added " << value << " to the rear.\n";

}

// Delete element from the front of the deque

void deleteFront() {

if (isEmpty()) {

cout << "Deque is empty. Cannot delete element from the front.\n";

return;

}

cout << "Deleted " << arr[front] << " from the front.\n";

if (front == rear) {

front = rear = -1; // Reset deque

} else if (front == MAX - 1) {

front = 0;

} else {

front++;

}

}

// Delete element from the rear of the deque

void deleteRear() {

if (isEmpty()) {

cout << "Deque is empty. Cannot delete element from the rear.\n";

return;

}

cout << "Deleted " << arr[rear] << " from the rear.\n";

if (front == rear) {

front = rear = -1; // Reset deque

} else if (rear == 0) {

rear = MAX - 1;

} else {

rear--;

}

}

// Display the elements of the deque

void display() {

if (isEmpty()) {

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

return;

}

cout << "Deque elements: ";

int i = front;

while (true) {

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

if (i == rear) break;

i = (i + 1) % MAX;

}

cout << endl;

}

};

int main() {

Deque deque;

int choice, value;

do {

cout << "\nDeque Operations Menu:\n";

cout << "1. Add Front\n";

cout << "2. Add Rear\n";

cout << "3. Delete Front\n";

cout << "4. Delete Rear\n";

cout << "5. Display Deque\n";

cout << "6. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to add at front: ";

cin >> value;

deque.addFront(value);

break;

case 2:

cout << "Enter value to add at rear: ";

cin >> value;

deque.addRear(value);

break;

case 3:

deque.deleteFront();

break;

case 4:

deque.deleteRear();

break;

case 5:

deque.display();

break;

case 6:

cout << "Exiting program. Goodbye!\n";

break;

default:

cout << "Invalid choice. Please try again.\n";

}

} while (choice != 6);

return 0;

}