KATHMANDU UNIVERSITY

DHULIKHEL, KAVRE

School of Engineering

Lab Report No.: 03

Data Structures and Algorithm (COMP 202)

Submitted By:

Aayush Pokhrel

Roll No:42

Group: CE

Level: 2nd Year/1st Sem

Submitted To

Dr. Rajani Chulyadyo

Department of Computer Science And Engineering

Objective: Implement Queue data structure using an array as well as a linked list.

Using:

(a) enqueue(element): Adds an element into the queue

(b) dequeue(): Removes an element from the queue

(c) isEmpty(): Checks if the queue is empty

(d) isFull(): Checks if the queue is full

(e) front(): Gives the element at the front

(f) back(): Gives the element at the rear

Description

About Queue

A queue is a common data structure that follows the First-In-First-Out (FIFO) principle. It is similar to a queue of people waiting in line, where the person who arrives first is the first one to be served.

In a queue, elements are added at one end called the rear or back, and they are removed from the other end called the front or head. This behaviour is known as enqueue (adding an element) and dequeue (removing an element), respectively.

We Implemented It using Two methods-:

1)Using Array

That's the basic concept of implementing a queue using a linked list in a nutshell. We should handle edge cases appropriately and ensure the correct order of operations to maintain the FIFO (First-In-First-Out) property of the queue.

Initialize an array to store the elements and two variables to track the front and end indices. Enqueue an element by adding it at the end of the array and updating the end index. Dequeue an element by removing the element at the front of the array, shifting the remaining elements, and updating the front index. You can also implement additional operations, such as checking if the queue is empty, retrieving the front element, or getting the size of the queue.

2)Using Linked List

Define a structure for the nodes in the linked list, which should include the data and a pointer to the next node. Create two pointers, "front" and "rear," and initialize them as NULL. To enqueue an element, create a new node, assign the data, and update the pointers accordingly. To dequeue an element, remove the node at the front and update the pointers. To check if the queue is empty, check if the "front" pointer is NULL. Optionally, provide operations to peek at the front element without removing it and to get the size of the queue.

Main.cpp Code:

```
int main()
{
    // implementing queue using array
    cout << "Queue using Array:" << endl;
    ArrayQueue A(5);

    A.enqueue(5);
    A.enqueue(7);
    A.enqueue(9);
    cout << "Front element: " << A.getFront() << endl;
    cout << "Back element: " << A.getRear() << endl;

    A.dequeue();
    cout << "Front element after dequeue: " << A.getFront() << endl;
    cout << "Back element after dequeue: " << A.getRear() << endl;

    A.enqueue(3);
    A.enqueue(0);
    cout << "Front element after enqueue: " << A.getFront() << endl;
    cout << "Back element after enqueue: " << A.getRear() << endl;

    // Queue using Linked List
    cout << endl
         << "Queue using Linked List:" << endl;

    Queue Li1;

    Li1.enqueue(5);
    Li1.enqueue(7);
    Li1.enqueue(9);
    cout << "Front element: " << Li1.getFront() << endl;
    cout << "Back element: " << Li1.getRear() << endl;

    Li1.dequeue();
```

```
    cout << "Front element after dequeue: " << Li1.getFront() << endl;
    cout << "Back element after dequeue: " << Li1.getRear() << endl;

    Li1.enqueue(3);
    Li1.enqueue(0);
    cout << "Front element after enqueue: " << Li1.getFront() << endl;
    cout << "Back element after enqueue: " << Li1.getRear() << endl;

    return 0;
}
```

Output Of The Above Problem:

```
Queue using Array:
Front element: 5
Back element: 9
Front element after dequeue: 7
Back element after dequeue: 9
Front element after enqueue: 3
Back element after enqueue: 0

Queue using Linked List:
Front element: 5
Back element: 9
Front element after dequeue: 7
Back element after dequeue: 9
Front element after enqueue: 3
Back element after enqueue: 0
```

Github Link: [1014Aayush/LabWork (github.com)]