**Department of Computing**

**School of Electrical Engineering and Computer Science**

**CS250 – Data Structures and Algorithms**



**Lab 7: Implementation of Queues in Different Problems**

**Submission Details**

|  |  |
| --- | --- |
| Name | CMS ID |
| Muhammad Umer | 345834 |
| Group | GP – 1 |
| Lab Engineer | Anum Asif |
| Faculty Member | Bostan Khan |
| Class | BEE12 |
| Date | 18/03/2024 |
| Time | 10:00 am – 12:50 pm |

**Table of Contents**

[2 Implementation of Queues in Different Problems 3](#_Toc161648333)

[2.1 Introduction 3](#_Toc161648334)

[2.2 Objectives 3](#_Toc161648335)

[2.3 Tools/Software Requirement 3](#_Toc161648336)

[2.4 Description 3](#_Toc161648337)

[2.5 Deliverables 3](#_Toc161648338)

[3 Lab Task 4](#_Toc161648339)

[4 Conclusion 7](#_Toc161648340)

# Implementation of Queues in Different Problems

## Introduction

This lab is based on queues and its implementation statically and dynamically.

## Objectives

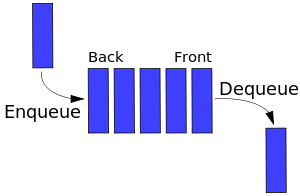
Objective of this lab is to get familiar with the queues and implement it in a programming language.

## Tools/Software Requirement

* Visual Studio C++

## Description

In computer science, a queue is a particular kind of abstract data type or collection in which the entities in the collection are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position and removal of entities from the front terminal position. This makes the queue a First-In-First-Out (FIFO) data structure.



The following sets of operation are generally supported by queue:

1. void Enqueue(element) – add an element at the rear end of the queue
2. element Dequeue() – removes and display the element from the front end of the queue
3. bool isEmpty() – checks if the queue is empty or not
4. bool isFull() – checks if the queue is full or not
5. void Clear() – release the memory allocated by queue
6. void FirstElement() – display the contents of first element of queue at front location

## Deliverables

Compile a single word document by filling in the solution parts and submit this file on LMS. The name of word document should follow this format. i.e., YourFullName(reg)\_Lab#. You must show the implementation of the tasks in a complete manner to get your work graded.

# Lab Task

You have to implement a waiting room management system in an emergency ward of a hospital. Your program will assign an id number to a patient in a first come first serve basis. The lower the id, the sooner the service will be provided to the patient.

Your program will contain the following methods:

**RegisterPatient():** This method assigns an id (which is auto-generated) to a patient and register him/her to the system.

**ServePatient():** This method calls a patient to provide hospital service to him/her.

**CancelAll():** This method cancels all appointments of the patients so that the doctor can go to lunch.

**CanDoctorGoHome():** This method returns true if no one is waiting, otherwise, returns false.

**ShowAllPatient():** This method shows all ids of the waiting patients in SORTED order.

(Hint: use the sorting methods learnt in class using the appropriate data-structure for each task) [Sorted according to their names]

Solve the above problem using an **array-based** *queue*.

Code

#include <iostream>

#include <string>

using *namespace* std;

*struct* Patient {

*int* id;

    string name;

};

*class* Queue {

*private:*

    Patient\* patients;

*int* front, rear, capacity;

*public:*

    Queue(*int* *size*) {

        patients = new Patient[*size*];

        front = rear = -1;

        capacity = *size*;

    }

    ~Queue() { delete[] patients; }

*void* RegisterPatient(string *name*) {

        if ((rear + 1) % capacity == front) {

            cout << "Queue is full" << endl;

            return;

        }

        Patient patient;

        patient.id = rear + 1;

        patient.name = *name*;

        if (front == -1) { *// if queue is empty*

            front = rear = 0;

        } else {

            rear = (rear + 1) % capacity;

        }

        patients[rear] = patient;

    }

*void* ServePatient() {

        if (front == -1) {

            cout << "Queue is empty" << endl;

            return;

        }

        cout << "Serving patient -> (" << patients[front].id << ", "

             << patients[front].name << ")" << endl;

        if (front == rear) {

            front = rear = -1;

        } else {

            front = (front + 1) % capacity;

        }

    }

*void* CancelAll() { front = rear = -1; }

*bool* CanDoctorGoHome() { return front == -1; }

*void* ShowAllPatient() {

        if (front == -1) {

            cout << "Queue is empty" << endl;

            return;

        }

        cout << "Patients in queue: " << endl;

        for (*int* i = 0; i <= rear; i++) {

            cout << "#" << patients[i].id << " -> " << patients[i].name

                 << endl;

        }

    }

};

*void* displayMenu() {

    cout << "\*\*== Emergency Ward Management System ==\*\*" << endl;

    cout << "1) -> Register Patient" << endl;

    cout << "2) -> Serve Patient" << endl;

    cout << "3) -> Cancel All" << endl;

    cout << "4) -> Can Doctor Go Home" << endl;

    cout << "5) -> Show All Patient" << endl;

    cout << "6) -> Exit" << endl;

}

*int* main() {

    Queue queue(5);

*int* choice;

    string name;

    displayMenu();

    while (true) {

        cout << "\nEnter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                cout << "Enter patient name: ";

                getline(cin >> ws, name);

                queue.RegisterPatient(name);

                break;

            case 2:

                queue.ServePatient();

                break;

            case 3:

                queue.CancelAll();

                break;

            case 4:

                cout << "Can doctor go home? -> "

                     << (queue.CanDoctorGoHome() ? "Yes" : "No") << endl;

                break;

            case 5:

                queue.ShowAllPatient();

                break;

            case 6:

                cout << "Exiting ..." << endl;

                return 0;

            default:

                cout << "Invalid choice" << endl;

        }

    }

    return 0;

}

Output

root@Zonularity:/home/zonularity/dsa# cd "/home/zonularity/dsa/lab\_7/" && g++ task.cpp -o task && "/home/zonularity/dsa/lab\_7/"task

\*\*== Emergency Ward Management System ==\*\*

1) -> Register Patient

2) -> Serve Patient

3) -> Cancel All

4) -> Can Doctor Go Home

5) -> Show All Patient

6) -> Exit

Enter your choice: 1

Enter patient name: Muhammad Umer

Enter your choice: 1

Enter patient name: Ahmed Mohsin

Enter your choice: 5

Patients in queue:

#0 -> Muhammad Umer

#1 -> Ahmed Mohsin

Enter your choice: 2

Serving patient -> (0, Muhammad Umer)

Enter your choice: 3

Enter your choice: 5

Queue is empty

Enter your choice: 4

Can doctor go home? -> Yes

Enter your choice: 6

Exiting ...

# Conclusion

In this lab, we successfully implemented a queue ADT using arrays. Making use of the First-in First-out (FIFO) property of the queue, we effectively realized an emergency ward management system in C++. Future improvements can include a Linked List-based implementation, although, practically, the number of beds in a hospital are fixed, for which the current implementation suffices.