# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

**Class: BSCS 10AB**

**Lab 05 : Queue**

**CLO1: Understand the fundamentals of data structures and algorithms**

**Mashal Ashfaque**

**337203**

**CS10-A**

# Instructor: Dr. Yasir Faheem

# Lab Engineer: Aftab Farooq

# 

# Lab 05 : Linear and Circular Queue

**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**

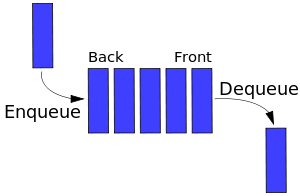
Visual Studio c++, Eclipse C++ IDE

**Helping Material**

Lecture slides, text book

**Description**

In [computer science](http://en.wikipedia.org/wiki/Computer_science), a queue is a particular kind of [abstract data type](http://en.wikipedia.org/wiki/Abstract_data_type) or [collection](http://en.wikipedia.org/wiki/Collection_(computing)) 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](http://en.wikipedia.org/wiki/FIFO_(computing)).



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

**Tasks:**

**Task 1:**

You are required to implement all operations of Queue ADT using an array-based linear queue.

#include <iostream>

using namespace std;

#define SIZE 5

class queue

{

public:

int items[SIZE];

int front = -1;int rear = -1;

int length;

void Enqueue(int element);// Add an element at the end of the queue

int Dequeue(); // Remove an element from the front of the queue

bool isEmpty(); // Check to see if queue is empty

bool isFull(); // Check to see if queue is full

int FirstElement();// Returns first element in queue without removing it

void Clear(); // Clear the queue

};

bool queue::isEmpty()

{

return rear == -1; //return true if its empty

}

bool queue::isFull()

{

return rear == SIZE-1; //return true if it is full

}

void queue::Enqueue(int val)

{

if (!isFull())

{

if(front==-1)

{

front = 0;

}

items[++rear] = val; //preincrement rear

}

else

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

}

int queue::Dequeue()

{

if (!isEmpty())

{

if (front == rear) //only 1 elemment left

{

rear = -1;

}

int flag = items[front ++];

return flag ;

}

else

{

cout << "Queue is Empty" << endl;

return -9999; //invalid

}

}

int queue::FirstElement()

{

if (!isEmpty())

{

return items[front];

}

else

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

}

void queue::Clear()

{

while (!isEmpty())

{

Dequeue();

}

}

int main()

{

queue my\_queue;

while (1)

{

cout << "Select the option on how you want to use queue.\n1-Enqueue\n2-Dequeue\n3-View Front Element\n4-Clear Queue" << endl;

int option;

cin >> option;

switch (option) {

case 1:

{

int val;

cout << "Enter a value"; cin >> val;

my\_queue.Enqueue(val);

break;

}

case 2:

cout << "Element that was dequeued: "<< my\_queue.Dequeue()<<endl;

break;

case 3:

cout<<my\_queue.FirstElement();

break;

case 4:

my\_queue.Clear();

break;

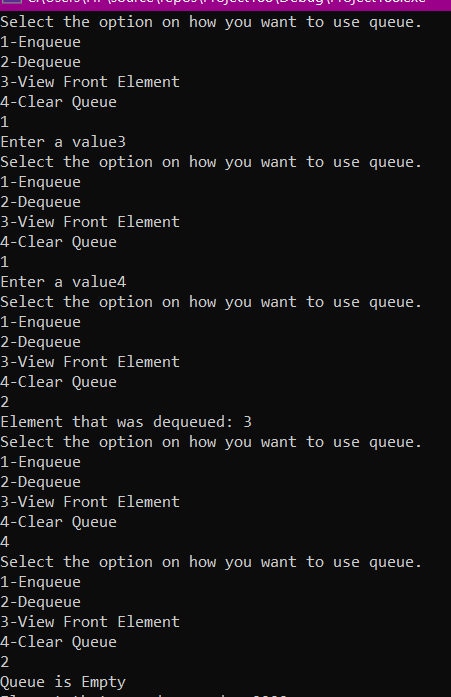
default:

break;

}

}

}



**Task 2:**

You are required to implement all operations of Queue ADT using an array-based circular queue

#include<iostream>

using namespace std;

#define SIZE 5

class queue // queue

{

public:

int items[SIZE];

int front = 0, rear = -1;

int count = 0;

bool isFull()

{

return count == SIZE ;

}

bool isEmpty()

{

return count == 0;

}

void Enqueue(int x)

{

if (!isFull())

{

rear = (rear + 1) % SIZE;

items[rear] = x;

count++;

}

else

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

}

int Dequeue()

{

if (!isEmpty()) {

int flag = items[front];

front = (front + 1) % SIZE;

count--;

return flag;

}

else {

cout << "Queue Underflow" << endl;

}

return 0;

}

int firstElement()

{

if (!isEmpty())

return items[front];

else

return -9999; // invalid value

}

void clear()

{

while (!isEmpty())

Dequeue();

}

};

int main()

{

queue my\_queue;

while (1)

{

cout << "Select the option on how you want to use queue.\n1-Enqueue\n2-Dequeue\n3-View Front Element\n4-Clear Queue" << endl;

int option;

cin >> option;

switch (option) {

case 1:

{

int val;

cout << "Enter a value"; cin >> val;

my\_queue.Enqueue(val);

break;

}

case 2:

cout << "Element that was dequeued: " << my\_queue.Dequeue() << endl;

break;

case 3:

cout << my\_queue.firstElement();

break;

case 4:

my\_queue.clear();

break;

default:

break;

}

}

}

**Task 3:**

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 circular queue.

**CODE:**

#include<iostream>

#include <algorithm>

using namespace std;

#define SIZE 5

class queue // queue

{

public:

int items[SIZE];

int front = 0, rear = -1;

int count = 0;

bool isFull()

{

return count == SIZE ;

}

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

//Same as is\_empty()

{

return count == 0;

}

void Register\_Patient() //This method assigns an Id (which is auto-generated) to a patient and register him/her to the system.

//Similar to Enqueue

{

static int ID = 0;

ID++;

if (!isFull())

{

rear = (rear + 1) % SIZE;

items[rear] = ID;

count++;

}

else

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

}

int Serve\_Patient() //Similar to dequeue

//This method calls a patient to provide hospital service to him / her.

{

if (!CanDoctorGoHome()) {

int flag = items[front];

items[front] = -999; //place invalid id in that place

front = (front + 1) % SIZE;

count--;

return flag;

}

else {

cout << "Queue Underflow" << endl;

}

return 0;

}

int firstElement()

{

if (!CanDoctorGoHome())

return items[front];

else

return -9999; // invalid value

}

void Cancel\_all()

{

while (!CanDoctorGoHome())

Serve\_Patient();

}

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

{

int temp\_arr[SIZE];

//start printing from front till rear,

//being a circu.ar queue, front will not necessarily be at 0 index and lat will not always be on last index so considering that

for (int i = 0; i < SIZE; i++)

{

temp\_arr[i] = items[i];

}

sort(temp\_arr, temp\_arr + SIZE);

for (int i = 0; i < SIZE; i++)

{

if (temp\_arr[i] >= 0)

{

cout << temp\_arr[i] << "\t";

}

}

}

};

int main()

{

queue my\_queue;

while (1)

{

cout << "\nSelect the option on how you want to use queue.\n1-Register patient\n2-Serve patient\n3-View All in waiting queue\n4-Cancel All\n5-Can Doctor go home?" << endl;

int option;

cin >> option;

switch (option) {

case 1:

{

int val;

my\_queue.Register\_Patient();

break;

}

case 2:

{ cout << "The patient that was served: " << my\_queue.Serve\_Patient() << endl;

break;

}

case 3:

{

cout << "Patients currently in waiting line: \n";

my\_queue.ShowAllPatient();

break;

}

case 4:

{ my\_queue.Cancel\_all();

break; }

case 5:

{

if (my\_queue.CanDoctorGoHome())

{

cout << "Yes.";

}

else cout << "No.Patients still in queue.";

break; }

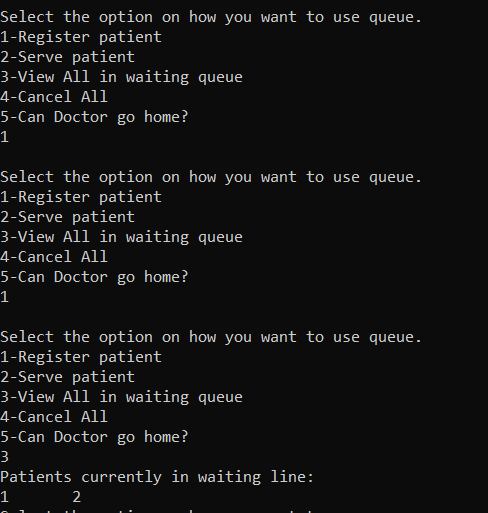
default:

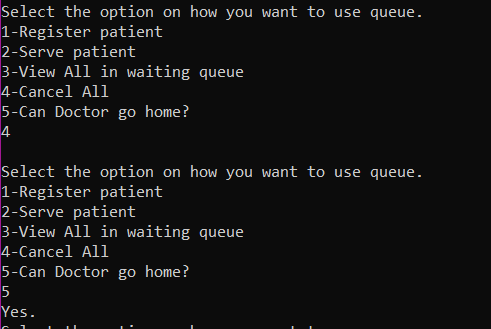
break;

}

}

}





**Task 4:**

Take an array of non-negative integers. Find the largest multiple of 3 that can be formed from array elements.   
For example, if the input array is {8, 1, 9}, the output should be “9 8 1”, and if the input array is {8, 1, 7, 6, 0}, output should be “8 7 6 0”.

**Hint :**   
1. Sort the array in non-decreasing order.  
2. Take three queues. One for storing elements which on dividing by 3 gives remainder as 0.The second queue stores digits which on dividing by 3 gives remainder as 1. The third queue stores digits which on dividing by 3 gives remainder as 2. Call them as queue0, queue1 and queue2  
3. Find the sum of all the digits.  
4. Three cases arise:   
……4.1 The sum of digits is divisible by 3. Dequeue all the digits from the three queues. Sort them in non-increasing order. Output the array.  
……4.2 The sum of digits produces remainder 1 when divided by 3.   
Remove one item from queue1. If queue1 is empty, remove two items from queue2. If queue2 contains less than two items, the number is not possible.  
……4.3 The sum of digits produces remainder 2 when divided by 3.   
Remove one item from queue2. If queue2 is empty, remove two items from queue1. If queue1 contains less than two items, the number is not possible.  
5. Finally empty all the queues into an auxiliary array. Sort the auxiliary array in non-increasing order. Output the auxiliary array.

**Code:**

#include <iostream>

#include <algorithm>

using namespace std;

#define SIZE 5

void printarr(int arr[3]);

class queue // queue

{

public:

int items[SIZE];

int front = 0, rear = -1;

int count = 0;

bool isFull()

{

return count == SIZE;

}

bool isEmpty()

{

return count == 0;

}

void Enqueue(int x)

{

if (!isFull())

{

rear = (rear + 1) % SIZE;

items[rear] = x;

count++;

}

else

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

}

int Dequeue()

{

if (!isEmpty()) {

int flag = items[front];

front = (front + 1) % SIZE;

count--;

return flag;

}

else {

cout << "Queue Underflow" << endl;

}

return 0;

}

int firstElement()

{

if (!isEmpty())

return items[front];

else

return -9999; // invalid value

}

void clear()

{

while (!isEmpty())

Dequeue();

}

};

void populateAux(int aux[], queue queue0, queue queue1,queue queue2, int\* top)

{

// Put all items of first queue in aux[]

while (!queue0.isEmpty()) {

aux[(\*top)++] = queue0.firstElement();

queue0.Dequeue();

}

// Put all items of second queue in aux[]

while (!queue1.isEmpty()) {

aux[(\*top)++] = queue1.firstElement ();

queue1.Dequeue();

}

// Put all items of third queue in aux[]

while (!queue2.isEmpty()) {

aux[(\*top)++] = queue2.firstElement();

queue2.Dequeue();

}

}

int main()

{

queue q0, q1, q2; //declaring 3 queues

int arr[10], answer[10] = { 0 };

int length, sum = 0;

cout << "Enter number of digits: (max 10)";

cin >> length;

while(length>10)

{

cout << "Enter number of digits: (max 10)";

cin >> length;

}

//Taking input

for (int i = 0; i < length; i++)

{

cout << "Enter digit no." << i + 1 << ": ";

cin >> arr[i];

}

//Bubble sort to sort array in non decreasing order

for (int i = 0; i < length - 1; i++)

{

for (int j = 0; j < length - i - 1; j++)

{

//Swapping

if (arr[j] > arr[j + 1]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

//Storing digits in the three queues according to the instructions

for (int i = 0; i < length; i++)

{

if (arr[i] % 3 == 0) { //if digit completely divisible by 3

q0.Enqueue(arr[i]);

}

else if (arr[i] % 3 == 1) { //if remainder is 1

q1.Enqueue(arr[i]);

}

else if (arr[i] % 3 == 2) { //if remainder is 2

q2.Enqueue(arr[i]);

}

}

//Taking sum

for (int i = 0; i < length; i++)

{

sum += arr[i];

}

if (sum % 3 == 1) {

if (!q1.isEmpty())

{

q1.Dequeue();

}

else if (!q2.isEmpty())

{

q2.Dequeue();

if (!q2.isEmpty())

{

q2.Dequeue();

}

else

{

cout << "Number not possible\n";

return 0;

}

}

else

{

cout << "Number not possible\n";

return 0;

}

}

else if (sum % 3 == 2) {

if (!q2.isEmpty())

{

q2.Dequeue();

}

else if (!q1.isEmpty())

{

q1.Dequeue();

if (!q1.isEmpty())

{

q1.Dequeue();

}

else

{

cout << "Number not possible\n";

return 0;

}

}

else

{

cout << "Number not possible\n";

return 0;

}

}

int counter = 0;

while (!q0.isEmpty())

{

answer[counter] = q0.Dequeue();

counter++;

}

while (!q1.isEmpty())

{

answer[counter] = q1.Dequeue();

counter++;

}

while (!q2.isEmpty())

{

answer[counter] = q2.Dequeue();

counter++;

}

//Bubble sort

for (int i = 0; i < counter - 1; i++)

{

for (int j = 0; j < counter - i - 1; j++)

{

//Swapping

if (answer[j] < answer[j + 1]) {

int temp = answer[j];

answer[j] = answer[j + 1];

answer[j + 1] = temp;

}

}

}

//output

cout << "the resultant array is: {";

for (int i = 0; i < counter; i++)

{

cout << answer[i] << " ";

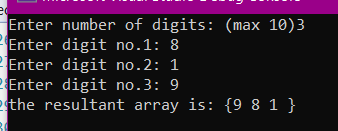
}

cout << "}" << endl;

return 0;

}

**Output:**



**Deliverables:**

Compile a single word document by filling in the solution part and submit this Word file on LMS. The name of word document should follow this format. i.e. **YourFullName(reg)\_Lab#.** This lab grading policy is as follows: The lab is graded between 0 to 10 marks. The submitted solution can get a maximum of 5 marks. At the end of each lab or in the next lab, there will be a viva related to the tasks. The viva has a weightage of 5 marks. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS. In case of any problems discuss it by emailing it to [aftab.farooq@seecs.edu.pk](mailto:aftab.farooq@seecs.edu.pk).

**Note:** Students are required to upload the lab on LMS before deadline.

Use proper indentation and comments. Lack of comments and indentation will result in deduction of marks.