Riphah International University
I-14 Main Campus
Faculty of Computing

Class:	Fall-2024	**Subject:**	Data Structures & Algorithms
Course Code:	CS 2124	**Lab Instructor:**	Zeeshan Ali

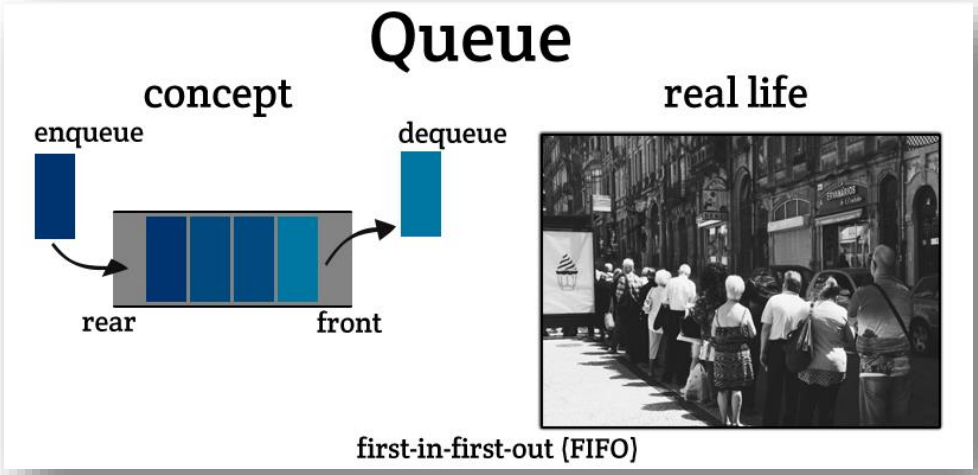
Learning Objective:

- What is Queue?
- Queue in C++.
- Operations of Queue
- Use of Queue
- Implementation of Queue in C++.

Queue:

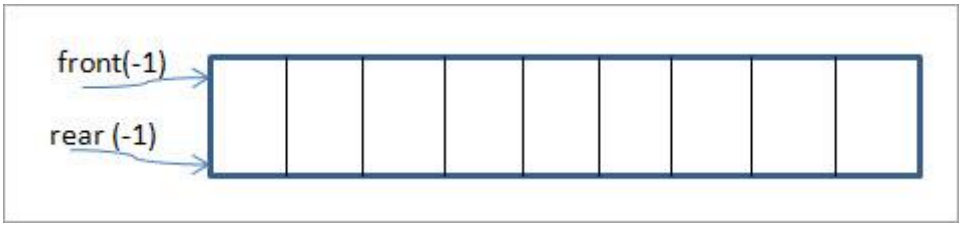
The queue is a basic data structure just like a stack. In contrast to stack that uses the LIFO approach, queue uses the FIFO (first in, first out) approach. With this approach, the first item that is added to the queue is the first item to be removed from the queue. Just like Stack, the queue is also a linear data structure.

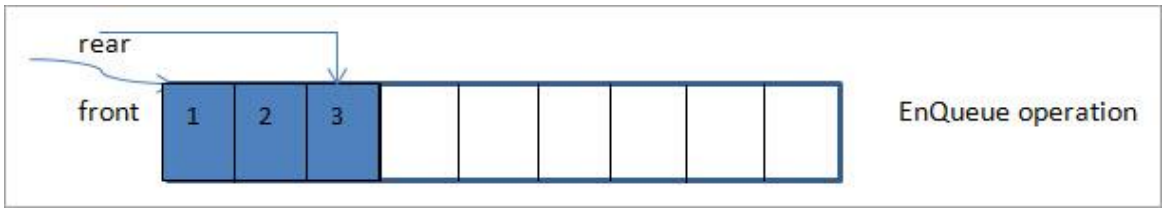
In a real-world analogy, we can imagine a bus queue where the passengers wait for the bus in a queue or a line. The first passenger in the line enters the bus first as that passenger happens to be the one who had come first.

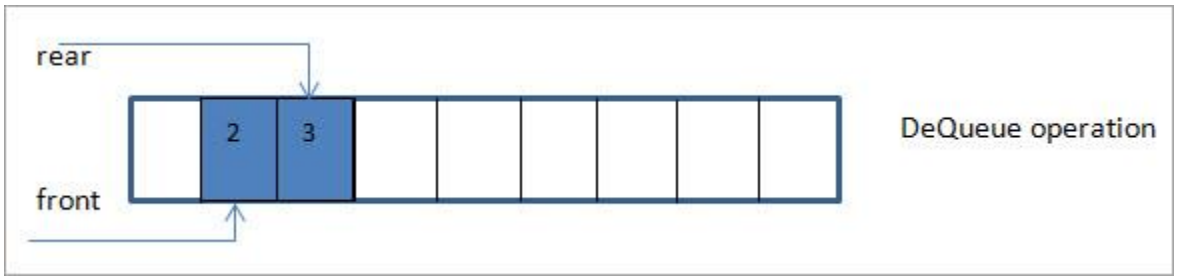


Queue in C++

In software terms, the queue can be viewed as a set or collection of elements as shown below. The elements are arranged linearly.







Operations of Queue

- **EnQueue:** Adds an item to the queue. Addition of an item to the queue is always done at the rear of the queue.
- **DeQueue:** Removes an item from the queue. An item is removed or de-queued always from the front of the queue.

Use of Queue

- CPU scheduling
- Synchronization
- Handling of interrupts in real-time systems

Implementation of Queue using link list in C++.

QueueLink.cpp

```
#include<iostream>

using namespace std;

struct Node
{
    int data;
    Node *next ;
};
class Queue
{
    Node *front, *rear;

public:
    Queue()
    {
        front = rear = NULL;  //   Initially

    }
    void Enqueue(int data)    // for insertion from rear
    {
        Node *newnode;
        newnode = new Node;
```

Compile Log | Debug | Find Results | Close

```
int main()

{
    Queue Q1;
    Q1.Enqueue(10);
    Q1.Enqueue(20);
    Q1.Enqueue(30);
    Q1.Enqueue(40);
    cout<<"Queue after Enqueue :"<<endl;
    Q1.display();
    Q1.Dequeue();
    cout<<"Queue after Dequeue :"<<endl;;
    Q1.display();
}
```

Compile Log | Debug | Find Results | Close

```
E:\00 Ripha Uni\CS3 Data Structures & Algorithms(Male)\C
Queue after Enqueue :
10      20      30      40
Queue after Dequeue :
20      30      40

--------------------------------
Process exited after 0.4542 seconds with r
Press any key to continue . . .
```

Task

1. Implement Class Queue, its data members, and operation listed below.

- Queue ()

A non-parameterized constructor that creates an empty queue. Where should the front

and rear of an empty queue point to?

- enqueue ()

Inserts the element at the rear of queue.

- dequeue ()

Removes the element from the front of queue.

- isEmpty ()

Returns True if queue is empty else returns False.

- display ()

Display all the elements of Queue

2. Take a single string as input. Using this input string, you have to create multiple queues in which each queue will comprise of separate word appeared in input string. At the end, you will again concatenate all queues to a single queue and return it to user.

Example:

String = “Data Structure and Algorithms”

Q1 = D → a → t → a

Q2 = S → t → r → u → c → t → u → r→e

Q3 = a → n → d

Q4 = A → l → g → o

At the end concatenate all queues.

Q1→Q2→Q3→Q4