**QUEUE**

A queue is a data structure used for storing data (similar to Linked Lists and Stacks). In queue, the order in which data arrives is important. In general, a queue is a line of people or things waiting to be served in sequential order starting at the beginning of the line or sequence.

**Definition: *A queue*** is an ordered list in which insertions are done at one end (*rear*) and

deletions are done at other end (*front*). The first element to be inserted is the first one to be

deleted. Hence, it is called First in First out (FIFO) or Last in Last out (LILO) list.

Similar to *Stacks*, special names are given to the two changes that can be made to a queue.

**Main Queue Operations**

* **EnQueue:** this is the process of inserting an element to a queue.
* **DeQueue:** this is the process of removing an element from the queue.

The exceptions involved in a queue are:

* **Underflow**: this is the process of dequeueing an empty set.
* **Overflow:** this is the process of enqueueing an element in a full queue.

**Diagramatic Representation of Enqueue and Dequeue**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |

Elements ready to

Be served

(DeQueue)

Front Rear New elements

ready to enter

the queue

(EnQueue)

**CONCEPT OF A QUEUE**

The concept of a queue can be explained by observing a line at a reservation counter. When we enter the line we stand at the end of the line and the person who is at the front of the line is the one who will be served next. He will exit the queue and be served.

As this happens, the next person will come at the head of the line, will exit the queue and will be

served. As each person at the head of the line keeps exiting the queue, we move towards the head of the line. Finally we will reach the head of the line and we will exit the queue and be served.

This behavior is very useful in cases where there is a need to maintain the order of arrival.

Another example is the queue at the ATM.

**APPLICATION AREAS OF THE QUEUE**

* Operating systems schedule jobs (with equal priority) in the order of arrival (e.g., a print queue).
* Simulation of real-world queues such as lines at a ticket counter or any other **firstcome first-served** scenario requires a queue.
* Multiprogramming.
* Asynchronous data transfer (file IO, pipes, sockets).
* Waiting times of customers at call center.
* Determining number of cashiers to have at a supermarket.

**IMPLEMENTATION OF QUEUE**

There are many ways (similar to Stacks) of implementing queue operations and some of the

commonly used methods are listed below.

* Circular array based implementation
* Linked list implementation

**Circular Array Based Implementation**

**Rear**

**Front**

This implementation of this Queue uses an array. In this array, elements are added circularly and two variables are used to keep track of the start element and end element. Generally, front is used to indicate the start element and rear is used to indicate the end element in the queue. The array storing the queue elements may become full. An EnQueue operation will then throw a full queue exception. Similarly, if we try deleting an element from an empty queue it will throw empty queue exception.

**LINKED LIST IMPLEMENTATION**

Another way of implementing queues is by using Linked lists. EnQueue operation is implemented by inserting an element at the end of the list. DeQueue operation is implemented by deleting anelement from the beginning of the list.

40

7

15

4

**Front** **Rear**