Lab 2

**Queue**

Queue is a non-primitive linear data structure in which elements are inserted from one end called rear(tail) and removing the elements from the other end called front (head). Queue operates data in FIFO manner.

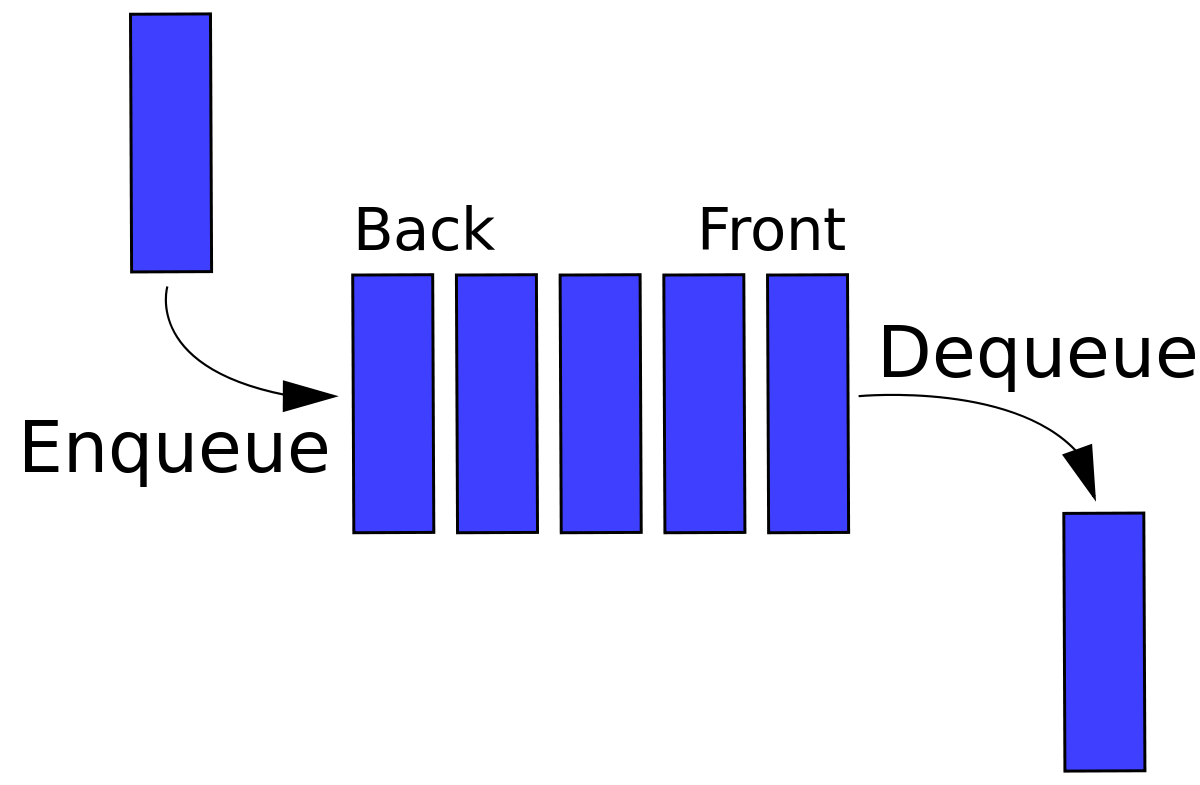


Fig: Queue

**Types of queue**

There are two types of queue. They are as follows:

1. Linear Queue
2. Circular Queue

* **Linear Queue:**

In this type of queue, data are added from one end called the head and removed from the other end called the tail. There are two types of operation that can be performed using a linear queue. They are:

* **Enqueue:**

Adding a new item to the front of the queue is an enqueue operation.

**Algorithm for enqueue operation**

Enqueue(LQueue,MAXSIZE,item,front,rear)

1. Start
2. Initialize front=0 and rear=-1
3. If rear=MAXSIZE-1 then display queue overflow and exit
4. Else rear=rear+1
5. Set LQueue[rear]=item
6. exit

* **Dequeue:**

Deleting an item from the end of the queue is a dequeue operation.

**Algorithm for dequeue operation**

Dequeue(LQueue,front,rear)

1. Start
2. If front>rear then display queue underflow and exit
3. Else, set data=LQueue[front]
4. Set front=front+1
5. Return data
6. exit

* **Circular Queue:**

In a circular queue the last element is connected back to the first element to make a circle. Elements are added at the tail end and removed from the front end. There are two types of operation that can be performed using a circular queue. They are:

* **Enqueue:**

Adding a new item to the front of the queue is an enqueue operation.

**Algorithm for enqueue operation**

Enqueue(CQueue,MAXSIZE,item,front,rear)

1. Start
2. Initialize front=-1 and rear=-1
3. If front=(rear+1)%MAXSIZE then display queue overflow and exit
4. Else goto step 5
5. If front=-1 then set front=0 else continue
6. rear=(rear+1)%MAXSIZE
7. Set CQueue[rear]=item
8. exit

* **Dequeue:**

Deleting an item from the end of the queue is a dequeue operation.

**Algorithm for dequeue operation**

Dequeue(CQueue,front,rear)

1. Start
2. If front=-1 then display queue underflow and exit
3. Else, set data=CQueue[front]
4. If front=rear then set front=rear=-1
5. Else front=(front+1)%MAXSIZE
6. Return data
7. exit

**Double ended Queue (Deque)**

It is the type of queue in which the insertion and deletion of item can be done from both of the ends (i.e from front as well as rear end). Operations that can be performed using a deque are listed below

* **Enqueue\_front**

Adding a new item to the front of the queue is an enqueue\_front operation.

**Algorithm for an enqueue\_front operation**

Enqueue\_front(Deque,MAXSIZE,item,front,rear)

1. Start
2. If front=(rear+1)%MAXSIZE then display queue is full and exit
3. Else if front=rear=-1 then set front=rear=0
4. Else if front=0 then set front=MAXSIZE-1
5. Else front=front-1
6. Set Deque[front]=item
7. exit

* **Enqueue\_rear**

Adding a new item to the end of the queue is an enqueue\_rear operation.

**Algorithm for an enqueue\_rear operation**

Enqueue\_rear(Deque,MAXSIZE,item,front,rear)

1. Start
2. If front=(rear+1)%MAXSIZE then display queue is full and exit
3. Else set rear=(rear+1)%MAXSIZE
4. Set Deque[rear]=item
5. exit

* **Dequeue\_front**

Removing an item from the front of the queue is a dequeue\_front operation.

**Algorithm for a dequeue\_front operation**

Dequeue\_front(MAXSIZE,front,rear,Deque)

1. Start
2. If front=-1 then display queue underflow and exit
3. Else if front=rear then set temp=Deque[front] and set front=-1 and rear=-1
4. Else set temp=Deque[rear] and set front=(front+1)%MAXSIZE
5. Return temp
6. exit

* **Dequeue\_rear**

Removing an item from the end of the queue is a dequeue\_rear operation.

**Algorithm for a dequeue\_rear operation**

Dequeue\_rear(MAXSIZE,front,rear,Deque)

1. Start
2. If front=-1 then display queue underflow and exit
3. Elseif front=rear then set temp=Deque[rear] and set front=-1 and rear=-1
4. Elseif rear=0 then set temp=Deque[rear] and set rear=MAXSIZE-1
5. Else set rear=rear-1
6. Return temp
7. exit