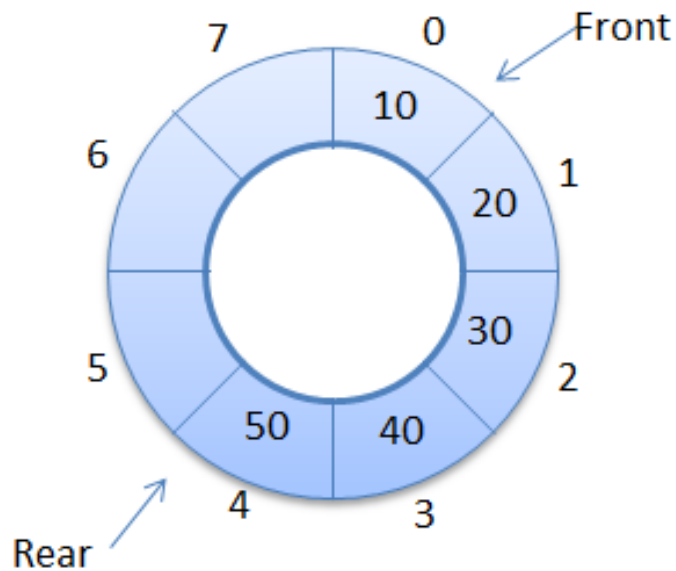
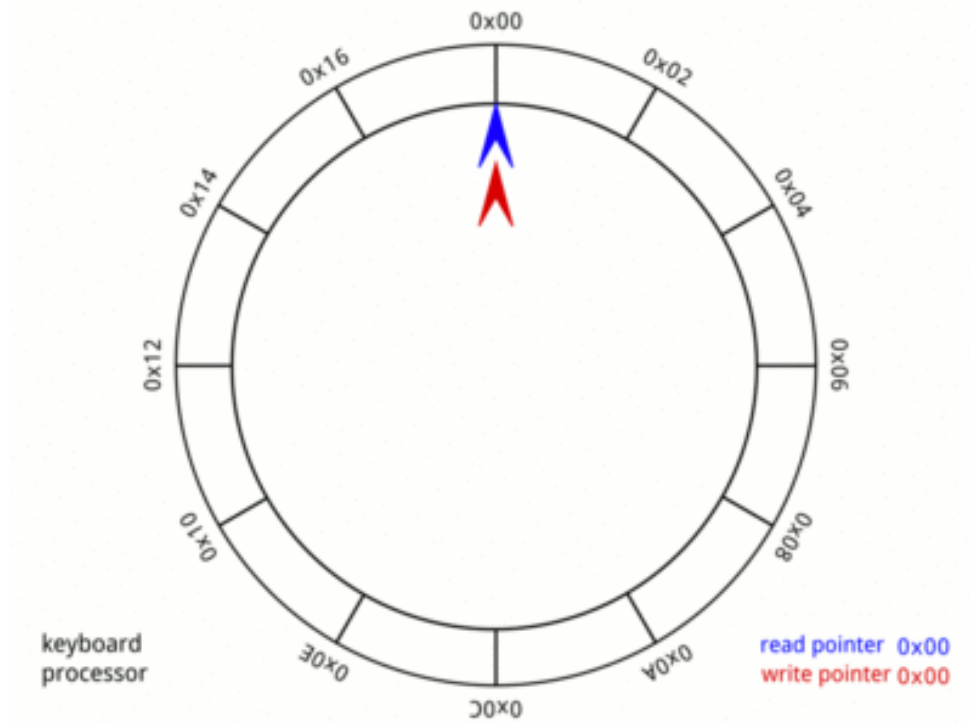
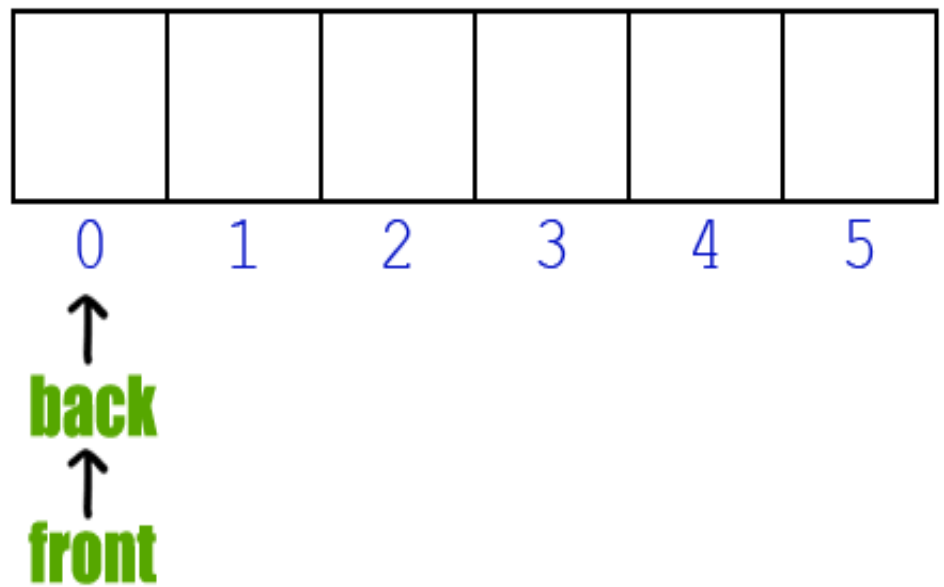
Circular Queue

- To solve this problem, queues implement wrapping around. Such queues are called Circular Queues.
- Both the front and the rear pointers wrap around to the beginning of the array.
- It is also called as “**Ring buffer**”.
- Items can inserted and deleted from a queue in O(1) time.


0
Front
1
2
3
4
Rear
5
6
7
10
20
30
40
50


0x00
0x02
0x04
0x06
0x08
0x0A
0x0C
0x0E
0x10
0x12
0x14
0x16
keyboard
processor
read pointer 0x00
write pointer 0x00


0
1
2
3
4
5
back
front

Enqueue

```
void enqueue(int x)
{
    if(front==(rear+1)%size)
        printf(“queue overflow");
   else if(front==-1&&rear==-1)
        front=rear=0;
   else
        rear=(rear+1)%size;
   queue[rear]=x;
}
```

Dequeue

```
void dequeue()
{
   if(front==-1&&rear==-1)
        printf(“queue underflow");
   else if(front==rear)
   {
        printf(“deleted ele=%d”,queue[front]);
        front=rear=-1;
   }
   else
   {
        printf(“deleted ele=%d”,queue[front]);
        front=(front+1)%size;
   }
 }
```

Display

```
void display()
{int i;
if(rear<front)
{
for(i=front;i<size;i++)
        printf("%d\t", queue[i]);
for(i=0;i<=rear;i++)
        printf("%d\t", queue[i]);
}
else
   for(i=front;i<=rear;i++)
        printf("%d\t", queue[i]);
  }
```


Deletion
Insertion
Insertion
Deletion
Front
Rear

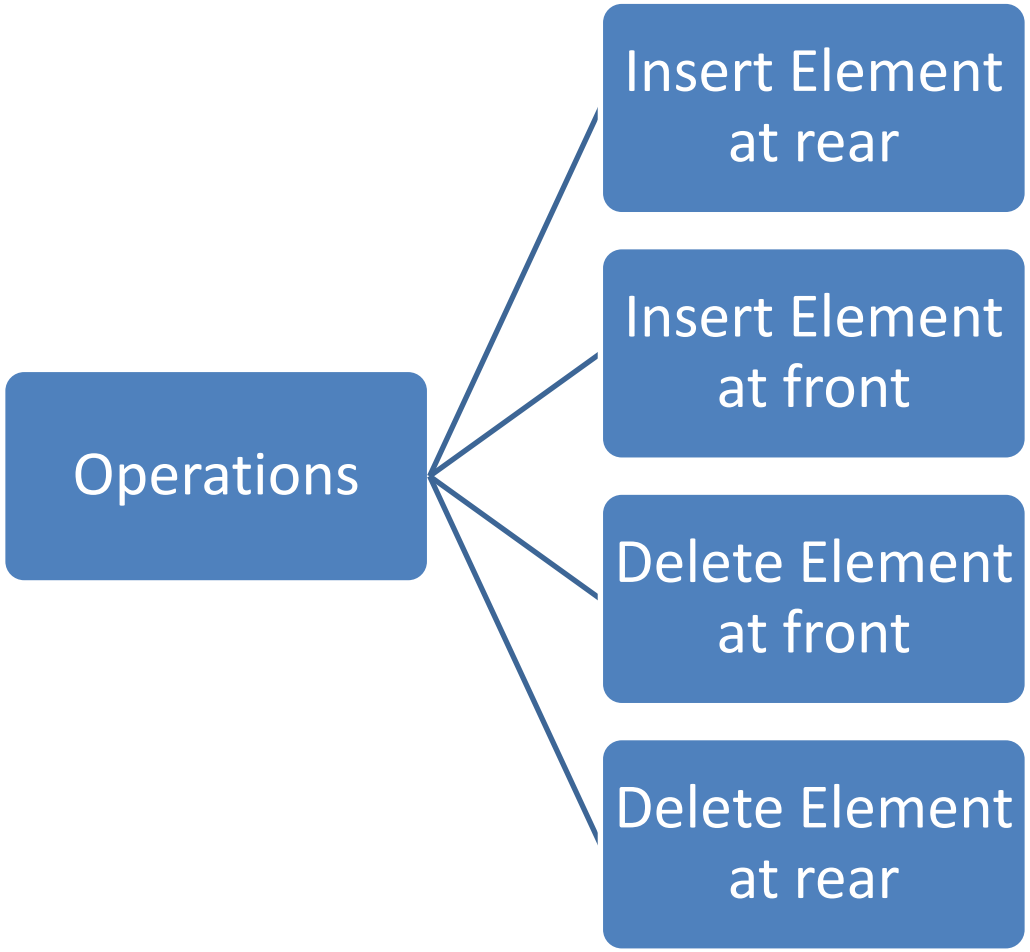
DEQUE

- A Deque or deck is a double-ended queue.
- Allows elements to be added or removed on either the ends.

Types

- Input Restricted
  - Elements can be inserted only at one end.
  - Elements can be removed from both the ends.
- Output Restricted
  - Elements can be removed only at one end.
  - Elements can be inserted from both the ends.

Operations


Operations
Insert Element at rear
Insert Element at front
Delete Element at front
Delete Element at rear

Enqueue Rear

```
void enqueue_rear(int x)
{
 if(rear==max-1)
{
    printf(“queue overflow");
    return;
}
 else if(front==-1&&rear==-1)
    front=rear=0;
else
    rear=rear+1;
q[rear]=x;
}
```

Dequeue Front

```
void dequeue_front()
{
 if(front==-1&&rear==-1)
{
    printf("underflow");
    return;
}
else if(front==rear)
          front=rear=-1;
 else
    front=front+1;
 }
```

Dequeue Rear

```
void dequeue_rear()
{
 if(rear==-1)
    printf(“queue underflow");
 else
    rear=rear-1;
}
```

Enqueue Front

```
void enqueue_front(int x)
{
   if(front<=0)
        printf("Cannot Insert in front");
   else
   {
        front=front-1;
        q[front]=x;
   }
}
```

Application

- Printing Job Management
- Packet Forwarding in Routers
- CPU Scheduling
  - Operating systems often maintain a queue of processes that are ready to execute or that are waiting for a particular event to occur
- Message queue in Windows
  - Computer systems must often provide a “holding area” for messages between two processes, two programs, or even two systems
- I/O buffer
  - When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes