

# Struktur Data Queue

Universitas Muslim Indonesia

Oleh  
**Lutfi Budi Ilmawan**

# Queue / Antrian

- Merupakan *list* di mana penambahan dan pengambilan elemen dilakukan pada sisi yang berbeda. Satu sisi untuk penambahan, dan sisi lainnya untuk pengambilan elemen.
- Menggunakan aturan FIFO (First In First Out), yaitu elemen yang pertama masuk akan pertama kali diambil atau dilayani.
- Front adalah posisi paling bawah/depan dari elemen queue
- Rear adalah posisi teratas/belakang dari elemen pada queue.

# Operasi Dasar

Operasi utama:

- Enqueue : proses penambahan elemen baru pada queue
- Dequeue : proses pengambilan elemen pada queue

Operasi penunjang:

- createQueue : membuat queue baru
- isEmpty : proses pengecekan pada sebuah queue bahwa queue tersebut telah terisi atau tidak
- isFull : proses pengecekan pada sebuah queue bahwa queue tersebut telah mencapai kapasitasnya
- destroyQueue : mengosongkan isi dari queue

# Contoh

Queue Q dengan kapasitas  $n = 5$



Front = 0

Rear = -1

jmlItem = 0

# Contoh

Enqueue(Q, a)



Front = 0

Rear = 0

jmlItem = 1

# Contoh

Enqueue(Q, b)



Front = 0

Rear = 1

jmlItem = 2

# Contoh

Enqueue(Q, c)



Front = 0

Rear = 2

jmlItem = 3

# Contoh

Dequeue(Q, x)



Front = 1

Rear = 2

jmlItem = 2

x = 'a'



# Contoh

Enqueue(Q, d)



Front = 1

Rear = 3

jmlItem = 3

# Contoh

Enqueue(Q, e)



Front = 1

Rear = 4

jmlItem = 4

# Contoh

Dequeue(Q, x)



Front = 2

Rear = 4

jmlItem = 3

x = 'b'

# Contoh

Dequeue(Q, x)



Front = 3

Rear = 4

jmlItem = 2

x = 'c'

# Deklarasi

```
const int MAX=10;  
struct qType{  
    char elements[MAX];  
    int front;  
    int rear;  
    int jmlItem;  
};  
qType q;
```

# CreateQueue dan DestroyQueue

```
void createQueue(qType &queue){  
    queue.front = 0;  
    queue.rear = -1;  
    queue.jmlItem = 0;  
}
```

# isEmpty

```
bool isEmpty(qType queue){  
    return (queue.jmlItem == 0);  
}
```

# isFull

```
bool isFull(qType queue){  
    return (queue.jmlItem == Max);  
}
```



# Enqueue

```
void enqueue(qType &queue, char NewElement){  
    queue.rear = queue.rear + 1;  
    queue.jmlItem = queue.jmlItem + 1;  
    queue.elements[queue.rear] = NewElement;  
}
```

# Deque

```
void deQueue(qType &queue, char &PoppedElement){  
    PoppedElement = queue.Elements[queue.front];  
    queue.front = queue.front + 1;  
    queue.jmlItem = queue.jmlItem - 1;  
}
```