

# queue (antrian)



## Queue bersifat **FIFO (First In First Out)**

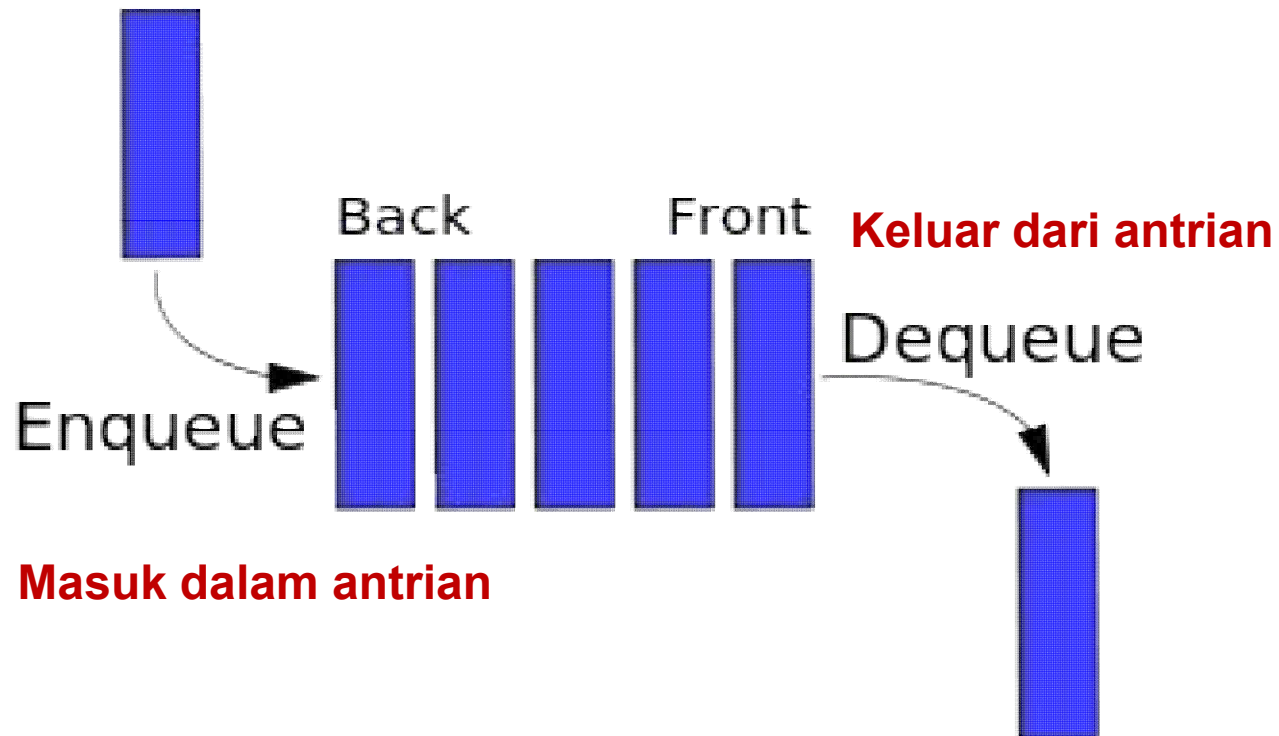
“elemen pertama yang ditempatkan pada queue adalah yang pertama dipindahkan”



# Implementation

- client-server systems
- operating systems: task queue, printing queue
- simulation and modeling: air traffic control, urban transport





# Representasi Antrian

# Operasi-operasi antrian

## ▪CREATE

Untuk menciptakan dan menginisialisasi queue dengan cara membuat Head dan Tail = -1

## ▪ISEMPTY

Untuk memeriksa apakah queue kosong

## ▪ISFULL

Untuk memeriksa apakah queue sudah penuh



## Operasi-operasi antrian

- **ENQUEUE**

Untuk menambahkan item pada posisi paling belakang

- **DEQUEUE**

Untuk menghapus item dari posisi paling depan

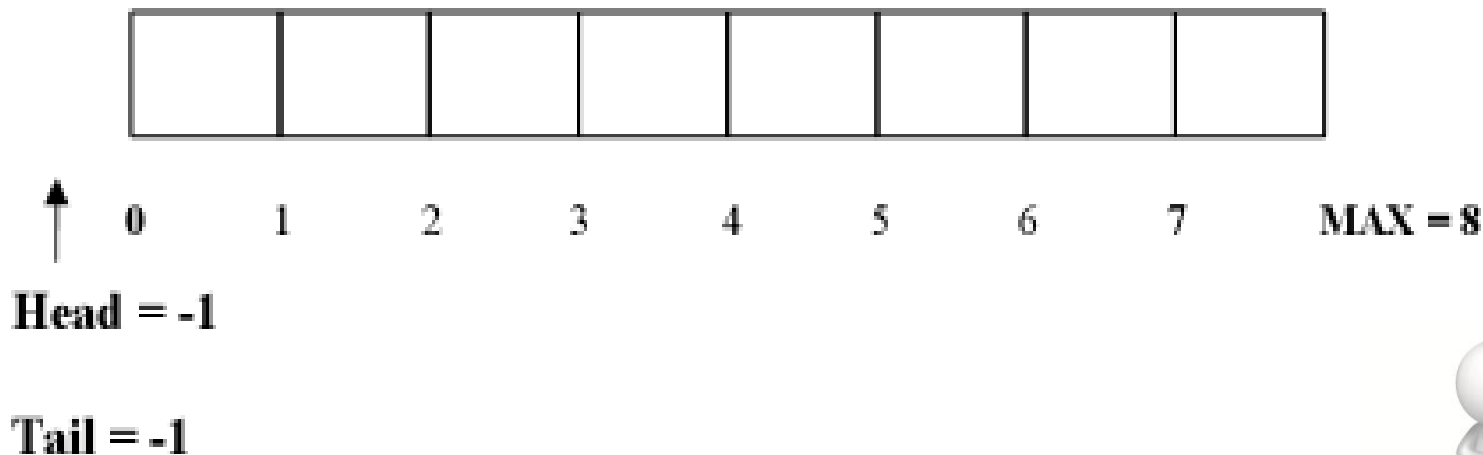
- **CLEAR**

Untuk mengosongkan queue



# Queue Linier Array

- Terdapat satu buah pintu masuk di suatu ujung dan satu buah pintu keluar di ujung satunya
- Sehingga membutuhkan 2 variabel: **Head dan Tail**



## DEKLARASI QUEUE

```
#define MAX 8
typedef struct{
    int data[MAX];
    int head;
    int tail;
} Queue;

Queue antrian;
```

## Queue (2)



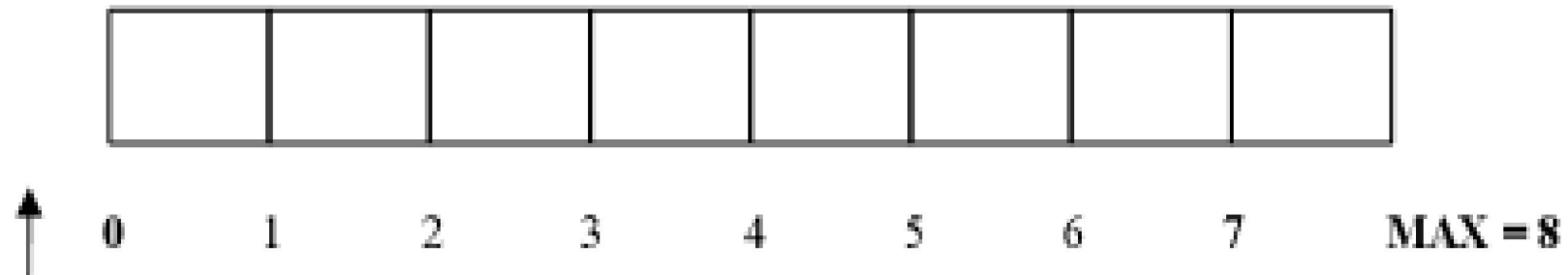
- Operasi-operasi:

### Create()

- Untuk menciptakan dan menginisialisasi Queue
- Dengan cara membuat **Head dan Tail = -1**



## Queue (3)



**Head = -1**

Antrian pertama kali

**Tail = -1**

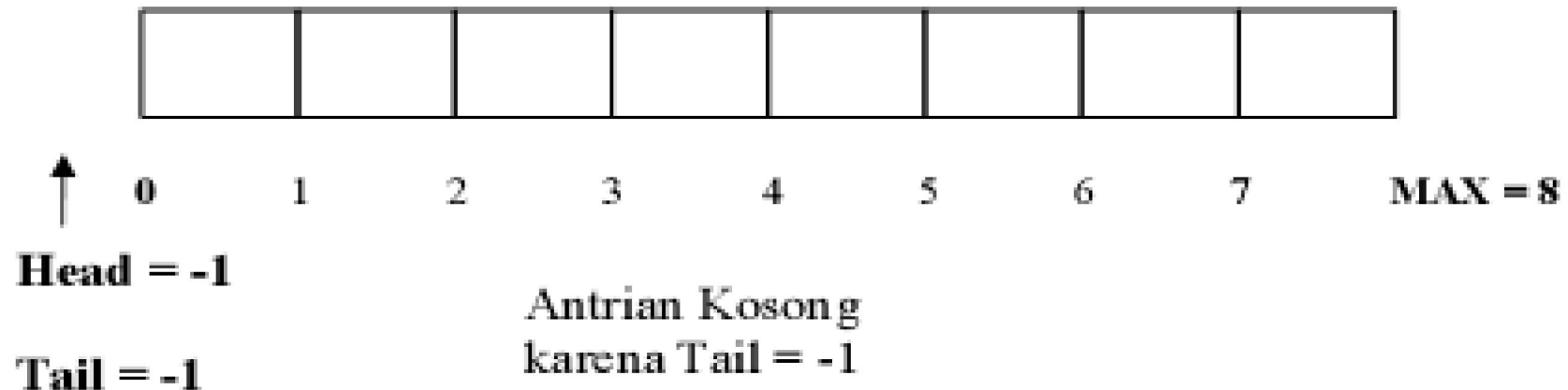
```
void Create() {  
    antrian.head=antrian.tail=-1;  
}
```

# Queue (4)

- **IsEmpty()**
  - Untuk memeriksa apakah **Antrian sudah penuh atau belum**
  - Dengan cara memeriksa nilai Tail, jika Tail = -1 maka **empty**
  - Kita tidak memeriksa Head, karena **Head adalah tanda untuk kepala antrian** (elemen pertama dalam antrian) yang tidak akan berubah-ubah
  - Pergerakan pada Antrian terjadi dengan penambahan elemen Antrian kebelakang, yaitu menggunakan **nilai Tail**



# Queue (5)



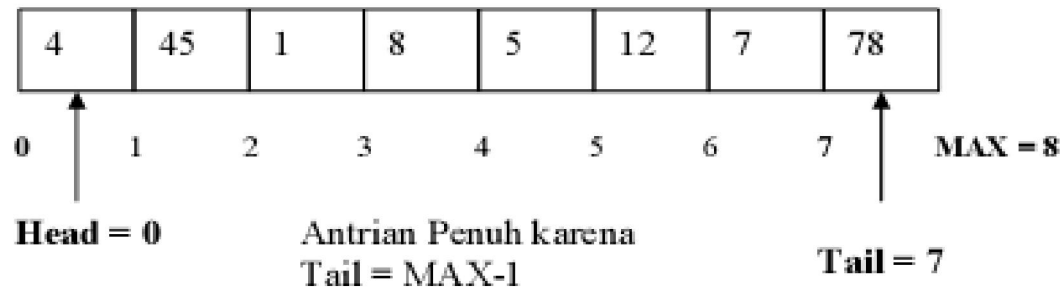
```
int isEmpty(){
    if(antrian.tail==-1)
        return 1;
    else
        return 0;
}
```



# Queue (6)

## Fungsi IsFull

- Untuk mengecek apakah Antrian sudah **penuhi atau belum**
- Dengan cara mengecek nilai Tail, **jika Tail  $\geq$  MAX-1** (karena MAX-1 adalah batas elemen array pada C) berarti **sudah penuh**



```
int IsFull(){
    if(antrian.tail==MAX-1) return 1;
    else return 0;
}
```

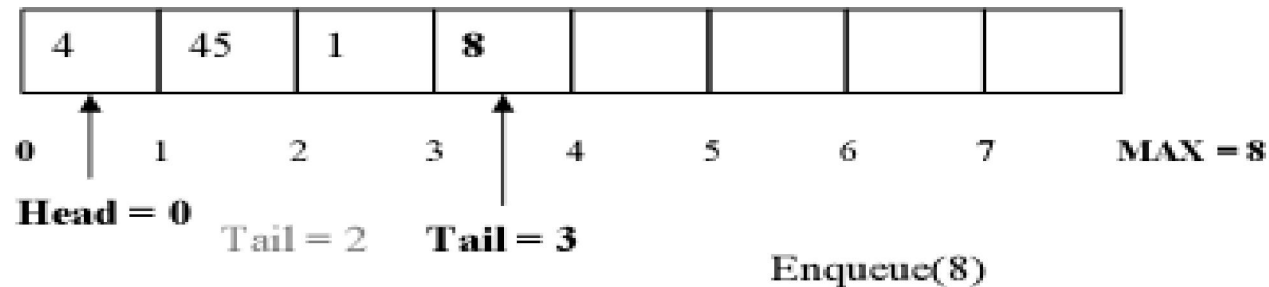
# Queue (7)

## Enqueue

- Untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling **belakang**
- Penambahan elemen selalu menggerakkan variabel Tail dengan cara **increment counter Tail** terlebih dahulu



# Queue (8)



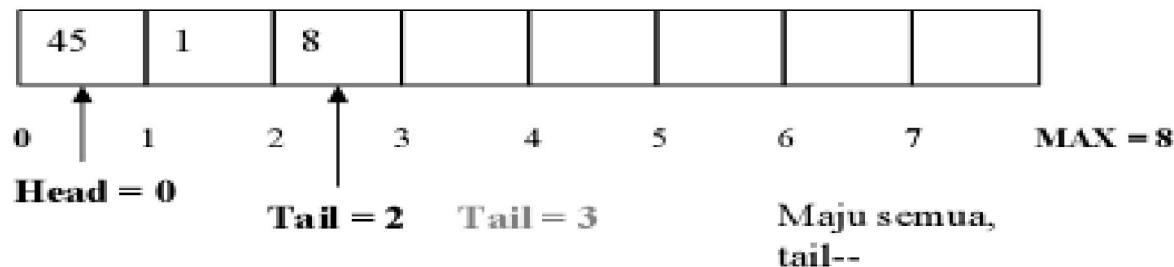
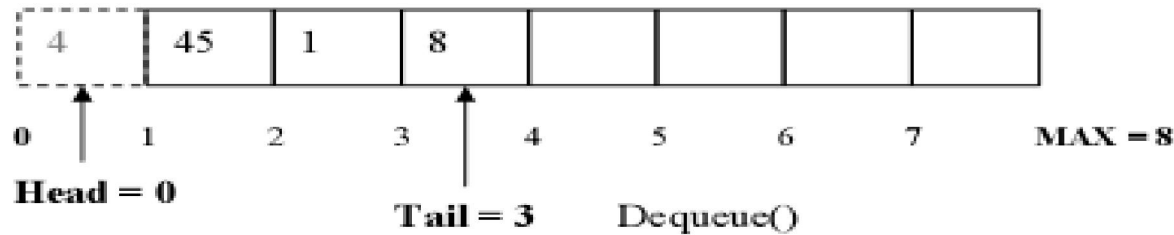
```
void Enqueue(int data){
    if(IsEmpty()==1){
        antrian.head=antrian.tail=0;
        antrian.data[antrian.tail]=data;
        printf("%d masuk!",antrian.data[antrian.tail]);
    } else
    if(IsFull()==0){
        antrian.tail++;
        antrian.data[antrian.tail]=data;
        printf("%d masuk!",antrian.data[antrian.tail]);
    }
}
```

## Queue (9)

- **Dequeue()**
  - Digunakan untuk **menghapus** elemen terdepan/pertama (head) dari Antrian
  - Dengan cara **menggeser** semua elemen antrian kedepan dan **mengurangi Tail dgn 1**
  - Penggeseran dilakukan dengan **menggunakan looping**



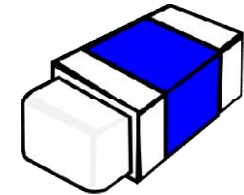
# Queue (10)



```
int Dequeue() {
    int i;
    int e = antrian.data[antrian.head];
    for(i=antrian.head; i<=antrian.tail-1; i++){
        antrian.data[i] = antrian.data[i+1];
    }
    antrian.tail--;
    return e;
}
```



# Queue (11)



- **Clear()**
  - Untuk **menghapus elemen-elemen Antrian** dengan cara membuat Tail dan Head = -1
  - Penghapusan elemen-elemen Antrian sebenarnya tidak menghapus array-nya, namun hanya **mengeset indeks pengaksesan-nya ke nilai -1** sehingga elemen-elemen Antrian tidak lagi terbaca



# Queue (12)



↑ 0 1 2 3 4 5 6 7 MAX = 8

**Head = -1**

**Tail = -1**

```
void Clear() {  
    antrian.head=antrian.tail=-1;  
    printf("data clear");  
}
```



# Queue (13)

- **Tampil()**

- Untuk **menampilkan** nilai-nilai elemen Antrian
- Menggunakan **looping dari head s/d tail**



```
void Tampil(){
    if(IsEmpty()==0){
        for(int i=antrian.head;i<=antrian.tail;i++){
            printf("%d ",antrian.data[i]);
        }
    }else printf("data kosong!\n");
}
```

```
#include <stdio.h>
#include <conio.h>
#include <iostream>
#define MAX 8
using namespace std;
```

```
struct Queue
{
    int data[MAX];
    int head;
    int tail;
};
Queue antrian;
```

1

```
void Create()
{
    antrian.head=antrian.tail=-1;
}

int IsEmpty()
{
    if(antrian.tail==-1)
        return 1;
    else
        return 0;
}

int IsFull()
{
    if(antrian.tail==MAX-1)
        return 1;
    else
        return 0;
}
```

2

```
void Enqueue(int data)
```

```
{
```

```
  if(IsEmpty()==1)
```

```
  {
```

```
    antrian.head=antrian.tail=0;
```

```
    antrian.data[antrian.tail]=data;
```

```
    cout<<antrian.data[antrian.tail];
```

```
  }
```

```
  else
```

**//kondisi lainnya jika penuh() sama dengan 0 maka antrian.ekor  
ditambah 1**

```
  {
```

```
    antrian.tail++;
```

```
    antrian.data[antrian.tail]=data;
```

```
    cout<<antrian.data[antrian.tail];
```

```
  }
```

```
}
```

## **int Dequeue()**

```
{  
    int i;  
    int e=antrian.data[antrian.head];  
    for(i=antrian.head;i<=antrian.tail-1;i++)  
    {  
        antrian.data[i]=antrian.data[i+1];  
    }  
    antrian.tail--;  
    return e;  
}
```

4

## **void Clear()**

```
{  
    antrian.head=antrian.tail=-1;  
    cout<<"Data Clear";  
}
```

5

## **void Tampil()**

```
{  
    if (IsEmpty()==0)  
        for (int i=antrian.head;i<=antrian.tail;  
i++)  
            cout<<antrian.data[i]<<" ";  
    else  
        cout<<"Data Kosong\n";  
}
```

```
int main()
{
    int pilih;
    int data;
    Create();
    do
    {
        clrscr();
        cout<<"\n=====MENU PILIHAN=====\\n";
        cout<<"1. Enqueue\\n";
        cout<<"2. Dequeue\\n";
        cout<<"3. Tampil\\n";
        cout<<"4. Clear\\n";
        cout<<"5. Keluar\\n";
        cout<<"-----\\n";
        cout<<"Masukkan Pilihan Anda -> ";
        cin>>pilih;
```

```
switch(pilih)
```

```
{
```

```
case 1:
```

```
    cout<<"Data : ";cin>>data;
```

```
    Enqueue(data);
```

```
    break;
```

```
case 2:
```

```
    if (IsEmpty()==0)
```

```
        cout<<"Elemen yang keluar : "<<Dequeue();
```

```
    else
```

```
        cout<<"Data kosong"<<endl;
```

```
        break;
```

6

```
case 3:
```

```
    Tampil();
```

```
    break;
```

```
case 4:
```

```
    Clear();
```

```
    break;
```

```
case 5:
```

```
    break;
```

```
}
```

```
    getch();
```

```
    } while(pilih!=5);
```

```
return 0;
```

```
}
```

7





# Ada pertanyaan?





**Terima kasih, sampai jumpa !**

