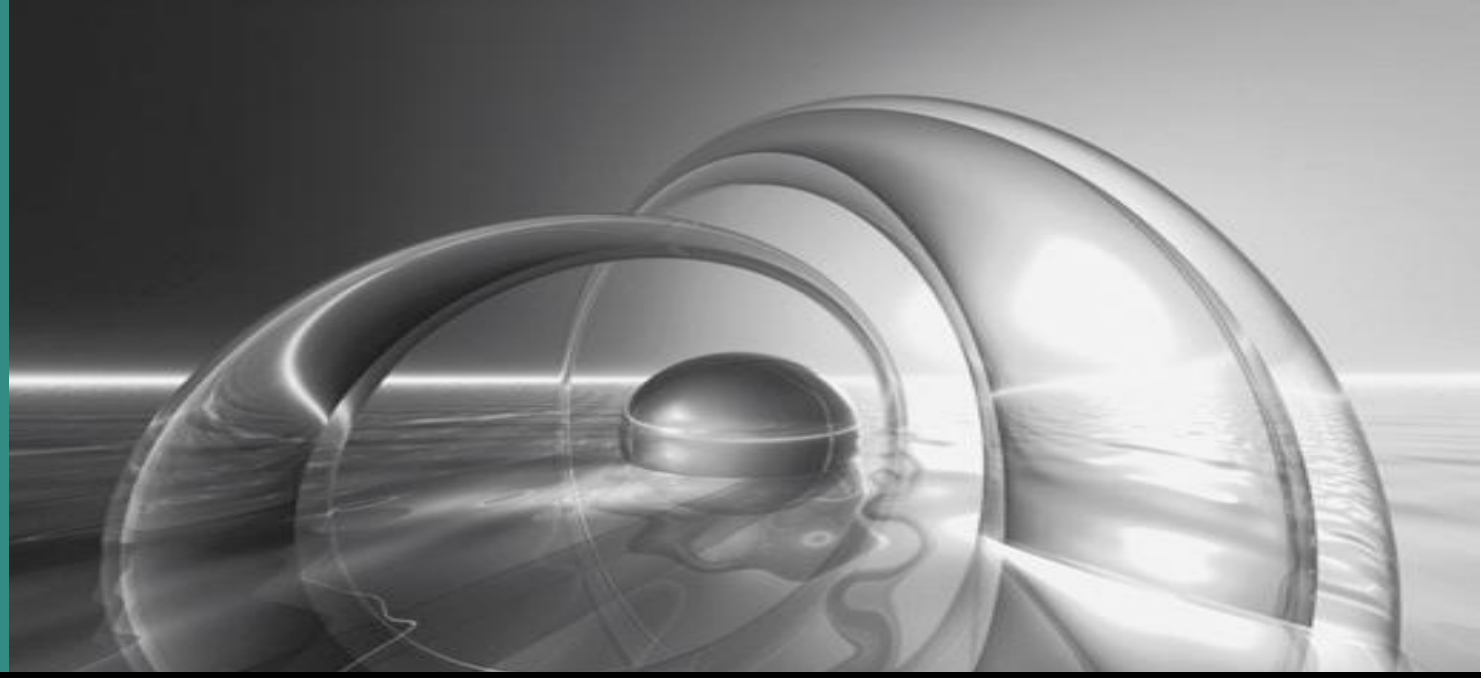


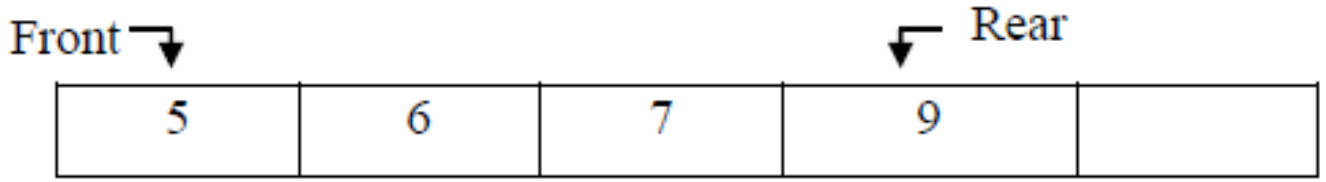
STRUKTUR DATA

QUEUE

QUEUE (ANTRIAN)

- **Queue (antrian) adalah barisan elemen yang apabila elemen ditambah,**

  maka penambahannya berada pada posisi belakang (rear) dan jika dilakukan pengambilan elemen dilakukan di elemen paling depan (front).
  FIFO (First In First Out)



Operasi Dasar

- **Enqueue : proses penambahan atau memasukkan satu elemen di belakang**
- **Dequeue : proses pengambilan atau mengeluarkan satu elemen di posisi depan**

Representasi Queue (Array Statis)

1. deklarasi:

```
const
  maksqueue=...
type
  typequeue= array[1..maksqueue]  of typedata

varqueue : typequeue
front, rear: integer
```

Representasi Queue (Array Statis)

2. **Penciptaan (create queue)**

   proses pemberian nilai 0 untuk variabel penunjuk depan (front) dan variabel penunjuk belakang (rear) dari queue.

```
front ← 0
rear ← 0
```

Representasi Queue (Array Statis)

3. **Fungsi kosong**

   digunakan untuk memeriksa apakah keadaan queue tidak memiliki elemen.
   Fungsi kosong didapatkan dengan memeriksa penunjuk rear dari queue.
   Jika penunjuk rear bernilai nol (0), maka berarti queue kosong dan jika tidak nol, maka berarti queue mempunyai elemen.

Representasi Queue (Array Statis)

Function Kosong (**Input** Rear : **integer**) → **Boolean**

{I.S : penunjuk Rear pada queue sudah terdefinisi}

{F.S : menghasilkan fungsi kosong}

Kamus:

Algoritma :

```
Kosong←false
If (Rear =0) then
    Kosong←True
EndIf
EndFunction
```

Representasi Queue (Array Statis)

4. **Fungsi penuh**

   digunakan untuk memeriksa apakah suatu queue telah penuh atau belum. Fungsi ini diperlukan ketika proses enqueue.
   Fungsi ini akan bernilai benar (true) jika penunjuk rear sama dengan nilai maksimum queue, jika tidak sama berarti queue belum penuh.

Representasi Queue (Array Statis)

```
Function Penuh (Input  Rear : Integer) → Boolean
{I.S : penunjuk Rear pada queue sudah terdefinisi}
{F.S : menghasilkan fungsi penuh}
Kamus:

Algoritma :
        Penuh←false
        If  (Rear =MaksQueue) Then
            Penuh←True
        EndIf
EndFunction
```

Representasi Queue (Array Statis)

5. **Proses Enqueue**

   (memeriksa queue kosong/ tidak dan queue penuh/ tidak)

   - **Proses untuk penambahan di posisi rear**
   - **Penambahan dilakukan jika kondisi queue tidak penuh**
   - **Jika keadaaan masih kosong, maka posisi front dan rear bernilai 1, tetapi jika sudah memiliki elemen maka nilai rear harus bertambah 1**
   - **Kemudian data baru disimpan di queue pada posisi rear**

Representasi Queue (Array Statis)

```
Procedure Enqueue (I/O front,rear: integer,
                           input elemen: tipedata)
{I.S: penunjuk front dan rear serta data yang akan dimasukkan
    ke queue sudah terdefinisi}
{F.S: menghasilkan queue yang sudah bertambah satu data}
Kamus:

Algoritma:
    if (kosong(rear)) then
        front ← 1
        rear ← 1
    else
        if (not penuh(rear)) then
                rear ← rear+1
        endif
    endif
        queue(rear) ← elemen
Endprocedure
```

Representasi Queue (Array Statis)

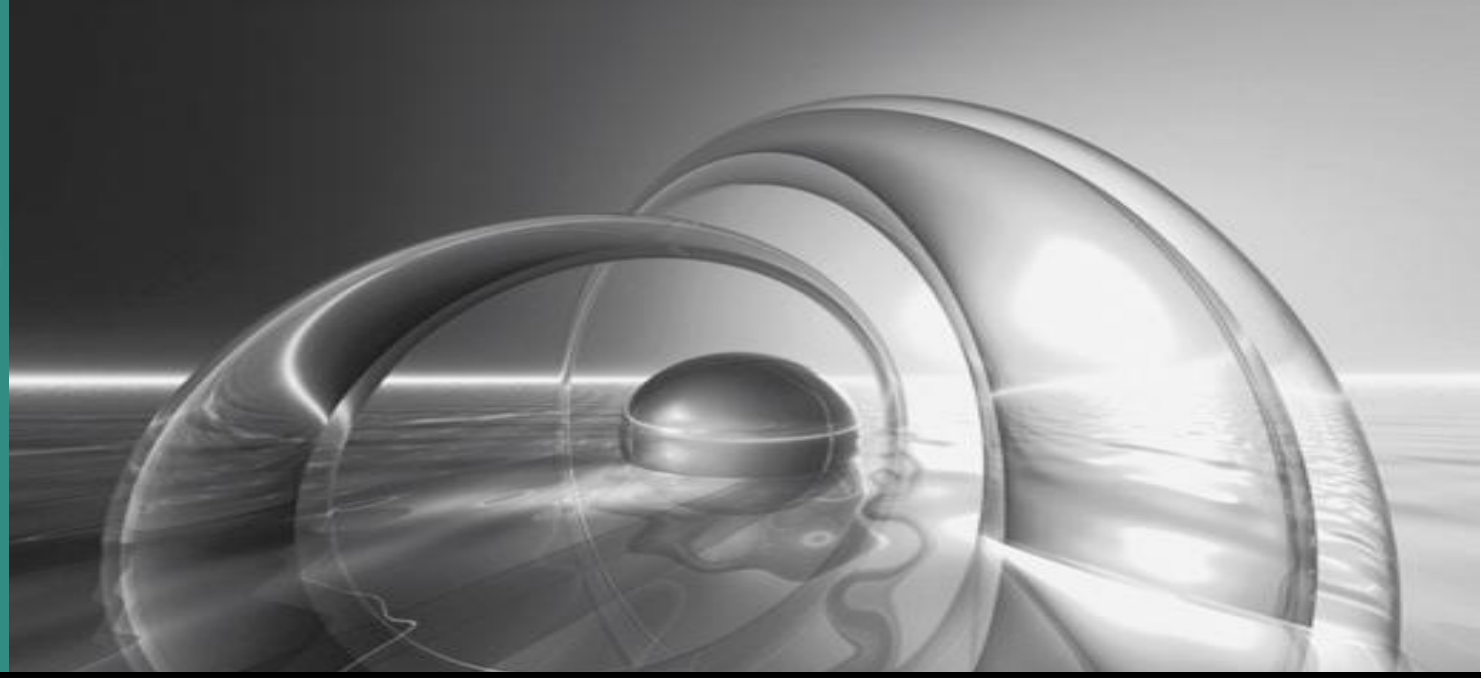
5. Proses Dequeue

- Proses pengambilan satu elemen dari queue
- Elemen yang diambil selalu dari elemen pertama
- Setelah elemen pertama diambil, maka akan terjadi proses pergeseran elemen data setelah elemen data yang diambil (dari posisi ke-2 sampai posisi paling belakang

Representasi Queue (Array Statis)

```
Procedure Dequeue (I/O front,rear: integer,
                                output elemen: tipedata)
{I.S: penunjuk front dan rear sudah terdefinisi}
{F.S: menghasilkan queue yang sudah berkurang satu data}
Kamus:
    i : integer
Algoritma:
    if (not kosong(rear)) then
        elemen ← queue(front)
        for i←1 to (rear-1) do
           queue(i)← queue(i+1)
        endfor
        rear ← rear-1
    endif

Endprocedure
```



Terima Kasih!