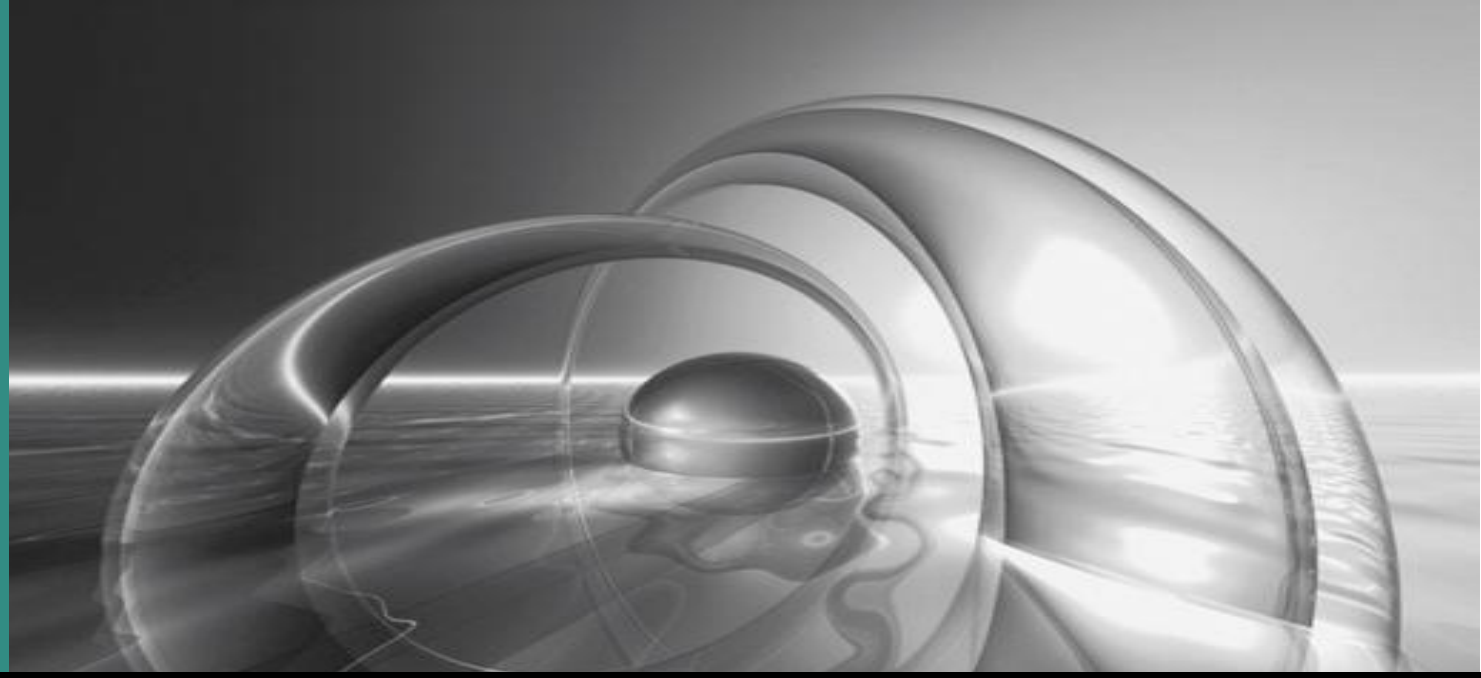


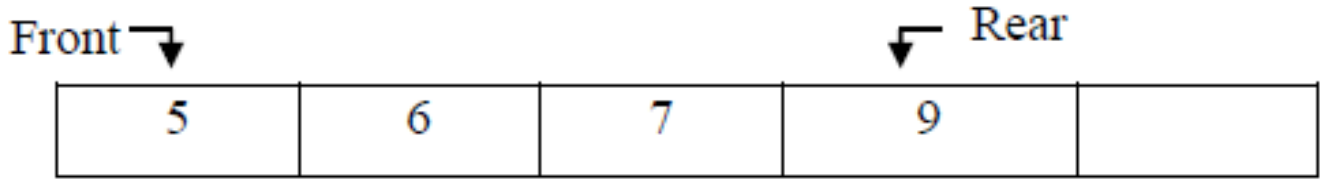
STRUKTUR DATA

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**

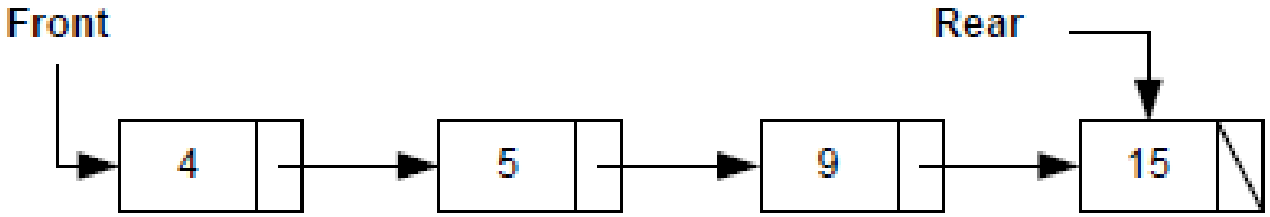
Representasi Queue (Array Statis)

```
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
```

Queue (Single Linked List)

Proses penyimpanan elemen queue pada single linked list menggunakan penyisipan di belakang dan penghapusan di depan.



Queue (Single Linked List)

1. deklarasi

```
type
  NamaPointer = ↑Simpul
  Simpul = Record
      info : tipedata
      next : NamaPointer
  endrecord

front, rear: NamaPointer
```

Queue (Single Linked List)

2. Penciptaan

Proses inisialisasi queue yang disimpan dalam bentuk linked list adalah memberikan nilai **nil** ke pointer front dan rear yang menandakan bahwa queue masih kosong.

```
Front ← nil
Rear ← nil
```

Queue (Single Linked List)

3. Fungsi Kosong

Fungsi ini untuk memeriksa apakah queue dalam keadaan kosong. Fungsi ini berguna ketika proses dequeue, maka harus diperiksa apakah memiliki data atau tidak.

Fungsi ini akan bernilai benar jika Front atau Rear bernilai nil.

Queue (Single Linked List)

Function Kosong(**Input** Front:NamaPointer) → **Boolean**

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

{F.S : menghasilkan fungsi kosong}

Kamus:

Algoritma :

```
Kosong←false
If (Front = Nil) Then
    Kosong←True
EndIf
EndFunction
```

Queue (Single Linked List)

4. Enqueue

- Proses enqueue dalam linked list adalah menambahkan elemen baru ke posisi belakang.
- Setelah itu, pointer penunjuk Rear harus berpindah ke posisi baru tersebut.
- Proses ini seperti penyisipan di belakang/ akhir pada single linked list

Enqueue (Single Linked List)

```
Procedure Enqueue (I/O front,rear: NamaPointer,
                           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:
    baru: NamaPointer
Algoritma:
    alloc(baru)
    baru↑.info ← elemen
    if (kosong(front)) then
        front ← baru
    else
        rear↑.next ← baru
    endif
    rear ← baru
    baru↑.next ← nil
Endprocedure
```

Queue (Single Linked List)

5. Dequeue

- Proses dequeue dalam linked list adalah mengambil data yang ditunjuk pointer Front.
- Pointer Front harus berpindah ke elemen antrian berikutnya. Proses ini dilakukan hanya jika linked list tidak kosong.
- Proses ini seperti penghapusan di depan/ awal pada single linked list

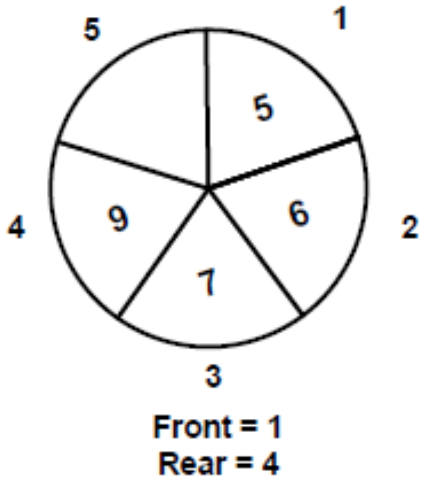
Dequeue (Single Linked List)

```
Procedure Dequeue (I/O front,rear: NamaPointer,
                           output elemen: tipedata)
{I.S: penunjuk front dan rear sudah terdefinisi}
{F.S: menghasilkan queue yang sudah berkurang satu data}
Kamus:
    phapus : NamaPointer
Algoritma:
    if (not kosong(front)) then
        phapus ← Front
        elemen ← Front↑.info
        if (Front=Rear) then
                Front ← nil
                Rear ← nil
        else
                Front← Front↑.next
        endif
        dealloc (phapus)
    endif

Endprocedure
```

Queue (Circular Array)

- Jika menggunakan array, maka ketika ada proses pengambilan data (dequeue) terjadi proses pergeseran data.
- Pergeseran data membutuhkan waktu yang lama jika data yang tersimpan banyak.
- Solusi agar proses pergeseran dihilangkan adalah dengan menggunakan **circular array.**



Aturan Queue Circular Array

- Proses penambahan elemen yaitu nilai belakang (Rear) ditambah 1

  Rear = Rear+1
- Proses penghapusan dilakukan dengan cara nilai depan (Front) ditambah 1
- Front = Front +1
- Jika Front=MaksQueue dan ada elemen yang akan dihapus, maka nilai Front=1
- Jika Rear=MaksQueue dan front tidak 1, maka jika ada elemen yang akan ditambahkan, nilai Rear=1
- Jika hanya tinggal satu elemen di queue (Front=Rear), maka Front dan Rear diisi 0 (queue kosong)

Contoh Queue Circular Array

No	Operasi	Status	Queue				
			1	2	3	4	5
1.	Inisialisasi	Front=0 Rear=0					
2.	Enqueue A, B, C	Front=1 Rear=3					
3.	Dequeue						
4.	Enqueue D, E						
5.	Dequeue 2 kali						
6.	Enqueue F						
7.	Enqueue G, H						
8.	Dequeue						
9.	Dequeue						
10.	Dequeue 2 kali						
11.	Dequeue						

Latihan

Diketahui maksimum circular queue = 6

Kondisi awal Front=3, Rear=6

Queue:

_____, _____, Cici, Lala, Eli, Jono

a) Tambahkan nama Rachel
b) Hapuskan satu data
c) Tambahkan nama Mimin
d) Hapuskan dua data
e) Tambahkan nama Caca
f) Hapuskan satu data
g) Hapuskan tiga data

Latihan

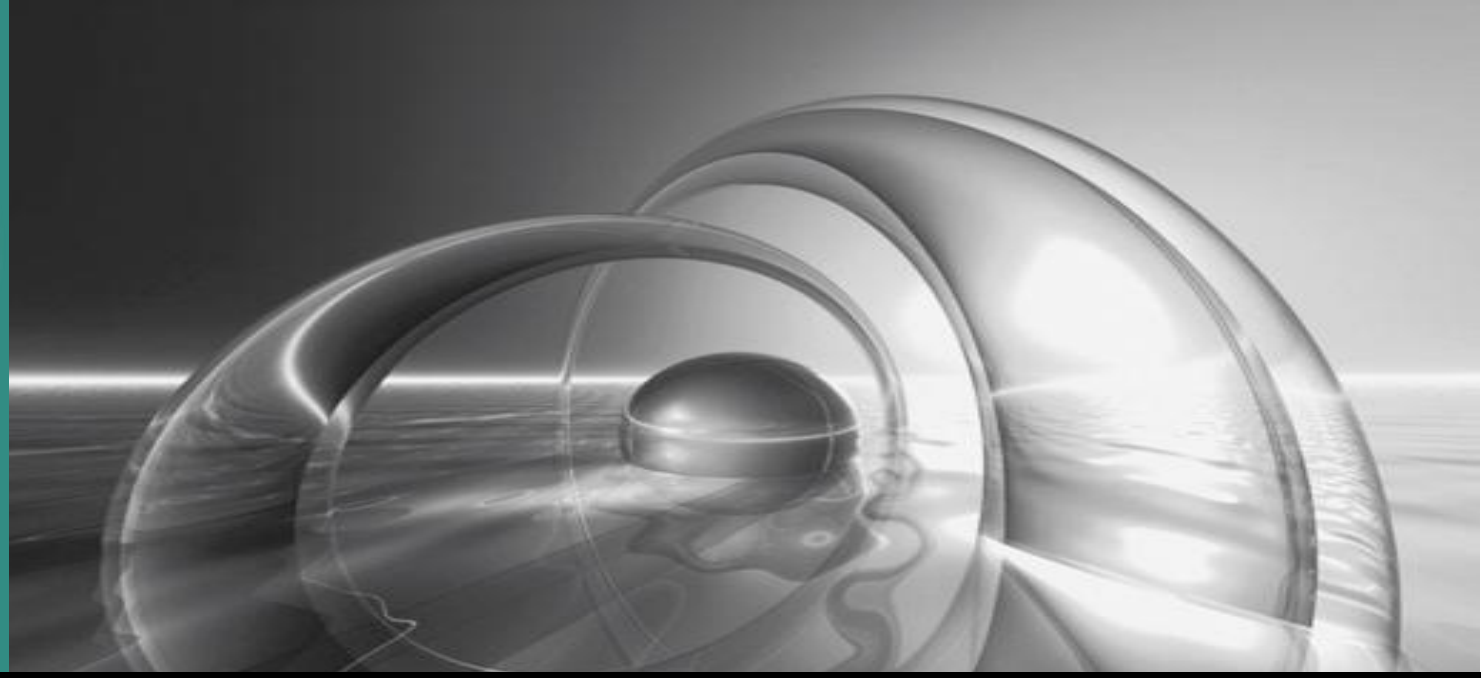
Diketahui maksimum circular queue = 6

Kondisi awal Front=2, Rear=5

Queue:

_____, London, Berlin, Rome, Paris, _____

a) Tambahkan kota Athens
b) Haspuskan dua kota
c) Tambahkan kota Madrid
d) Tambahkan kota Moscow
e) Tiga kota dihapus
f) Tambahkan kota Oslo



Terima Kasih!