# PERTEMUAN 14
# QUEUE (LANJUT)

## A. TUJUAN PEMBELAJARAN

Mampu menjelaskan pengertian *Queue*, dapat melakukan operasi penyisipan maupun penghapusan elemen dalam *Queue*, dan dapat mengimplementasikan *Queue* dengan *Linked List* pada bahasa pemrograman *C++.* Dimodul ini anda harus mampu :

Ketepatan dan penguasaan dalam penggunaan konsep Queue dengan Linked List dalam membangun aplikasi pada bahasa pemrograman C++, Latihan dan Tugas**.**

## B. URAIAN MATERI

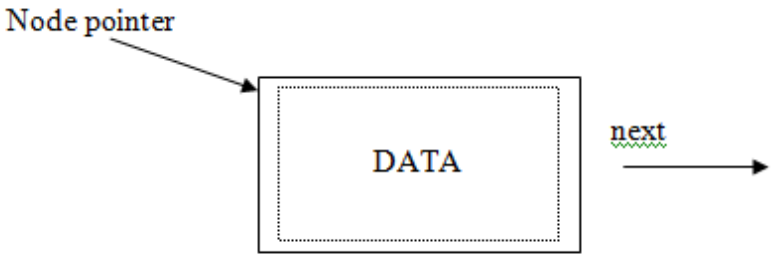
### 1. Implementasi Queue sebagai Linked List

*Queue* merupakan structure data antrean yang terbagi menjadi dua bagian, di sebelumnya telah dijelaskan structure data yang di implementasikan dengan *Array*. Disini akan melanjutkan *Queue* dimplementasikan dengan *Linked List.* Queue suatu gambaran special linear list mengunkan operasi sisip. Sisip hanya di izinkan hanya pada salah satu bagian *Rear (belakang)*, dan untuk operasi hapus pada bagian sisi sebaliknya yaitu bagian *Front (depan)* List.

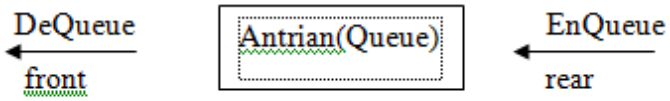
Operasi dari Queue mengunakan cara FIFO atau (First In First Out).adalah yang pertama masuk adalah yang pertama kali keluar.

### 2. CREATE

Ciptakan node baru , lalu suatu node akan memiliki data atau elemen yang akan di tugaskan padanya lalu suatu pointer ke penerusnya. Node merupakan nama baru yang akan dimasukan.




Contoh gambar 14.1. struktur node




Gambar 14.41 EnQueue dan Dequeque.

### 3. ENQUEUE DAN DEQUEUE

a. EnQueue adalah suatu proses menambah data di dalam Queue
b. DeQueue adalah suatu proses pengambilan data di dalam Queue.

```
Struct Queue
        {
            ElemenType data;
            Struct node*next :
        };
            Struct Queue
                {
                    Struct node*front;
                    Struct node*rear;
```

```
                    };
        Struct Queue Q;


         // next adalah pointer ke node selanjutnya
        } ;
        Typedef struct queue Q; // kita sebut saja Q.
```

Butuh dua pointer yang disebut front & rear untuk dapat digunakan.

Q*front=NULL

Q*rear=NULL

1) front adalah pointer bantu yang dipergunakan bertujuan menunjuk data yang paling depan.
2) Rear adalah pointer bantu yang dipergunakan bertujuan menunjuk data yang paling akhir atau belakang.

Buat variable head, tail, front, lalu rear bertipe Tnode.

## 4. INISIALISASI

*TNode head, tail, front, rear ;*

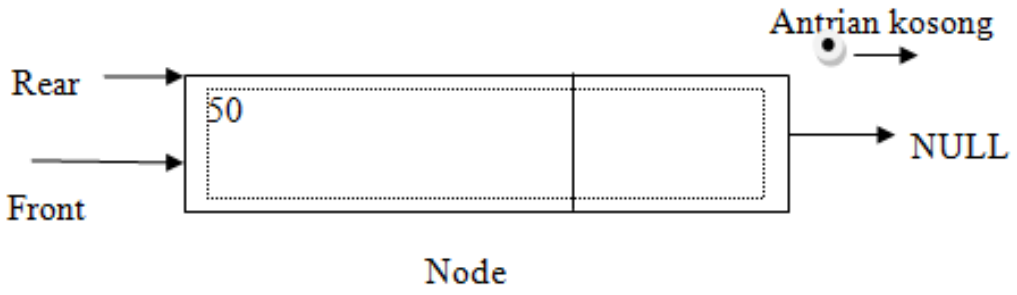
Pada proses operasi inisialisai variable head, tail, front lalu rear dengan cara memberikan nilai awal yaitu NULL.

```
Void inisialisai ()
        {
        Head=tail=front=rear=NULL;
        }
Tnode;
```

### 5. Tambah Simpul Jika Queue Kosong

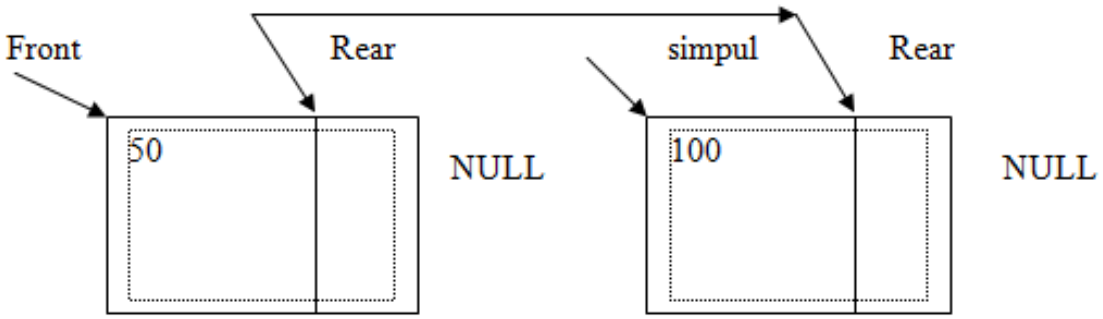
Untuk memastikan kondisi antrian kosong dapat mengunakan langkah-langkah berikut ini:

cek *front* atau *rear* apabila datanya null berarti antrian kosong.




Gambar 14.42Tambah elemet pertama dalam antrian.

Antrian kosong dan simpul yang akan di sisipkan adalah simpul pertama, oleh karena itu menetapkan front dan rear ke simpul bantuan untuk queue.




Gambar 14.43Tambah elemet dalam antrian.

*1. Rear->Next = Simpul*

*2. Rear = Simpul*

Catatan : Perpindahan pointer Rear ditunjukan oleh garis panah menuju akhir.

**6. isEmpty**

Cara cek antrian bila keadaan dalam kondisi kosong menggunakan metode isEmpty.

```
Int isEmpty()
        {
            If (Q.head ==NULL)
            Return (1);
Else
        Return (0);

        }
```

**7. RemoveFirst**

Contoh deklarasi RemoveFirst.

```
Void RemoveFirst ()
        {
            Node temp=head;
            If (head==tail)
                Head=tail=rear=NULL;
        Else
                {
                    Temp=temp.next;
                    Head=temp;
                    Temp=NULL;
                }
        }
```

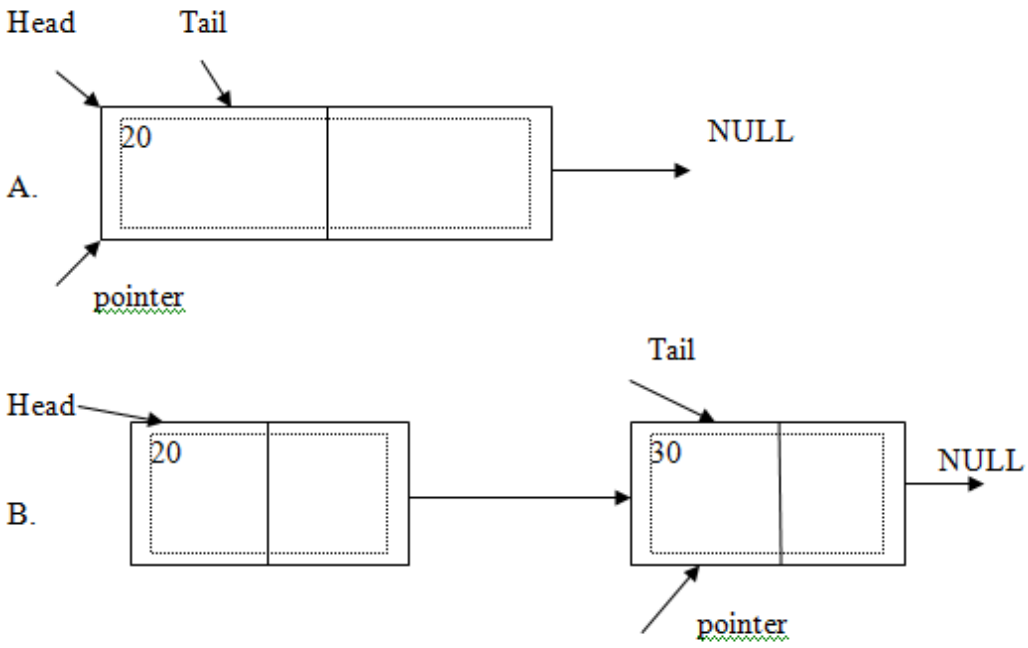
## 8. Enqueue

Adalah suatu proses menambah elemen di dalam EnQueue, ketika EnQueu berjalan, elemen di dalam Queue akan bertambah menjadi satu. Lalu pointer rear akan bergerak mengarah ke elemen baru yang sudah ditambahkan. Linked List rear akan mengarah pada simpul baru yang telah ditunjuk oleh tail.

Langkah-langkah pada EnQueue :

a. Menambah elemen baru didalam struktur bagian paling akhir atau belakang.
b. Perubahan pada posisi rear.

Contoh gambaran EnQueue.




Gambar 14.44 EnQueue.

Contoh deklarasi EnQueue.

```
Void EnQueque (ElemenType e)
        {
                Struct node *P;
                P=(Struct node*)malloc(sizeof(struct node)) ;
                P->data = e;
                P->next = NULL;
                If(empty())
                {
                        Q.Head=Q.Tail=P;
                }
        Else
                {
                        Q.Tail->next =P;
                        Q.Tail =P;
                }
        }
```

**9. AddLast**

Contoh deklarasi program AddLast

```
Void addLast (node input)
        {
                If (IsEmpty ())
        }
                Head=input;
```

```
                Tail=input;
        }
Else
        {
                Tail.next=input;
                Tail=input;
        }
}
```

**10. Operasi Peek**

Adalah suatu proses akses elemen yang diarahkan oleh front (elemen pertama ditambahkan), pada operasi ini berbeda dengan Enquue, karena proses ini tidak diiringi penghapusan data yang sudah ada tetapi pengaksesan atau pengembalian data saja.

Contoh deklarasi program Peek.

```
Int peekQueue()
        {
                Return front.data;
        }
```

**11. Antrian Prioritas**

Adalah sebuah elemen himpunan yang pada disetiap elemen akan diberikan level prioritas tertentu.lalu sebuah pemrosesan elemen berdasarkan pada prioritas sebagai berikut:

Elemen yang memiliki prioritas yang lebih tinggi diproses terlebih dahulu.

Lalu dua elemen yang memiliki sebuah prioritas level yang sama akan diproses sesuai urut kedatanganya, contohnya seperti yang datang pertama

## 12. Prioritas Antrian Dengan Metode One-Way List

Pngenalan Queue berprioritas dijalankan dengan one way list, berikut ini:

Pada tiap node mempunyai dua field:

a. Informasi (INF)
b. Nomor prioritas (PRN)

Dalam list, node X mendahului simpul Y , contohnya :

1) Apabila prioritas X memang lebih tinggi levelnya dibandingkan dengan prioritas Y.
2) Apabila kedua prioritas tersebut memiliki level yang sama, namun X terlebih dahulu telah masuk ke dalam antrian.

Kesimpulan : pada disetiap kali operasi penghapusan, nilai Front bertambah. Lalu untuk disetiapkali penambahan pada nilai Rear akan bertambah.

Contoh program untuk mengimplementasikan queue dengan Linked List.

```
//program implementasi Queue menggunakan Linked List
#include<stdio.h>
#include<stdlib.h>

    Struct queue
    {
        Int data;
        Struct queue *next;
        //struct queue *front;
            //struct queue*rear;
    };
    Typedef struct queue Q;
    Q *front=NULL;
    Q *rear=NULL;

    Void enqueue (int val);
```

```
Int dequeue ();
        // int isEmpty ();
        Void display ();
        //void exit ();

Void main ()
{
        Int ch,val,x;
        Do
{
        Printf("ANTRIAN MENGGUNAKAN LINKED LIST");
        Printf("\1.ENQUEUE");
        Printf("\2.DEQUEUE");
        Printf("\3.DISPLAY");
        Printf("\4.EXIT");
        Printf("masukan pilihan");
        Scanf("%d",& pilih);
        Switch(pilih)
                {
                        Case 1:
                                        Printf("\masukan nilai;");
                                        Scanf("%d",&val);
                                        Enqueue(val)
                                        Tampilkan();
                                        Break;
                        Case 2:
                                        X=Dequeue():
```

```
                    If(x!=-1)
                    Printf("nilai dequeue=%d\n",x);
                    Tampilakan():
                    Break;
            Case 3:
                    Display();
                    Break;
            Case 4:
                    Exit(1);
            Default:print("[PILIHAN SALAH");
            Break;
    }
            While(pilih=4);
    }
    Void enqueue(int val)
    {
            Q*node;
            Node=(Q*malloc(sizeof(Q)://buat simpul baru
            Node->data=val;
            Node->next=NULL;
            If(front==NULL)//periksa inisial kosong
            Front=node//apabila kosong, front ke simpul baru.
    Else
            Rear->next=node;
    Rear=node;
    }
    Inr dequeue()
```

```
{ int val;
Q*p;
If(front==NULL//periksa inisilalisai kosong.
{
Printf("LIST KOSONG");
Val=-1//jika queue kosong -1 adalah bantuan
Else
{
P=front;
Front=p->next//buat front point ke elemen selanjutnya
Val=p->data;
If(front==NULL)//jika fornt point null queue kosong
Rear=NULL//jadi buat rear point juga.
Free(p);
}
        Return(val);
{
Void display()
{
        Q*temp:
        Temp=front;
        Print("\hasil elemen Queue");
If(temp==NULL)
Print("queue kosong");
Else
{
While (temp->next!=NULL)
```

```
Print("\n%d",temp->data);
Temp=temp->next;
}//akhir while
//elemen akhir masih tersisa untuk ditampilkan
Printf("\n%d",temp->data;
}//selesai
```

## C. SOAL LATIHAN/TUGAS

| Latihan | Petunjuk Pengerjaan Tugas |
|---|---|
| **Latihan Dan Tugas 14** | 1. Buat contoh simulasi antrian dibank mengunakan linked list?<br>2. Buat 2 menu berbeda untuk nasabah dan teller dibank?<br>3. Buat program menghitung waktu tunggunya pada saat mengantri?<br>4. Buat flowchart penambahan pada antrian mengunakan linked list?<br>5. Buat flowchart penghapusan pada antrian mengunakan linked list? |

## D. REFERENSI

C And Data Structure by practice by Ramesh Vasappanvara.

Implementasi queue oleh Dinar Heriana.

Struktur data oleh Antonius Rachmat.