

# Double Linked List

Brigida Arie Minartiningtyas, M.Kom

# Review Linked List

---

- ▶ Linked list yang kita pelajari sebelumnya hanya mempunyai sebuah pointer pada setiap simpulnya.
- ▶ Hal ini merupakan kelemahan bahwa linked list tersebut hanya bisa dibaca dalam satu arah saja, yaitu dari kiri ke kanan.
- ▶ Hal yang seperti ini kurang cepat jika kita ingin mencari data di dalam linked list



# Double Linked List

---

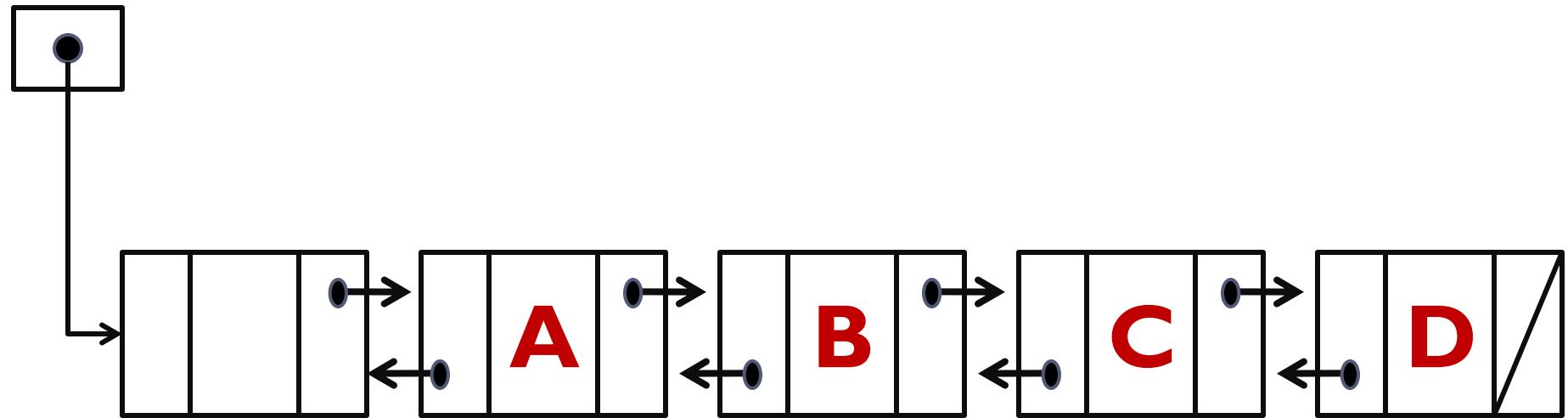
- ▶ Untuk itulah kita memerlukan linked list yang setiap simpulnya mempunyai 2 buah pointer
- ▶ Dengan pointer pertama menunjuk ke simpul sebelumnya (sebelah kiri) dan pointer kedua menunjuk ke simpul sesudahnya (disebelah kanan).
- ▶ Hal ini akan mempermudah pembacaan (bisa dilakukan dari kiri ke kanan dan dari kanan ke kiri), begitu juga untuk penambahan simpul baru dan penghapusan simpul



# Double Linked List



KEPALA



Secara garis besar Double linked list adalah linked list yang memiliki dua buah pointer yang menunjuk ke simpul sebelumnya (Prev) dan yang menunjuk ke simpul sesudahnya (Next).



# **DEKLARASI Linked List**

---

## **I. Menambah simpul**

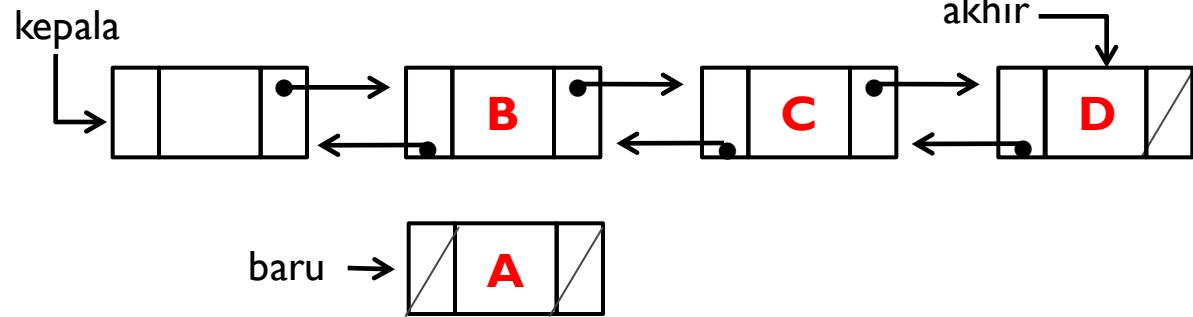
- Tambah belakang
- Tambah depan
- Tambah tengah

## **2. Menghapus simpul (dengan masukkan)**

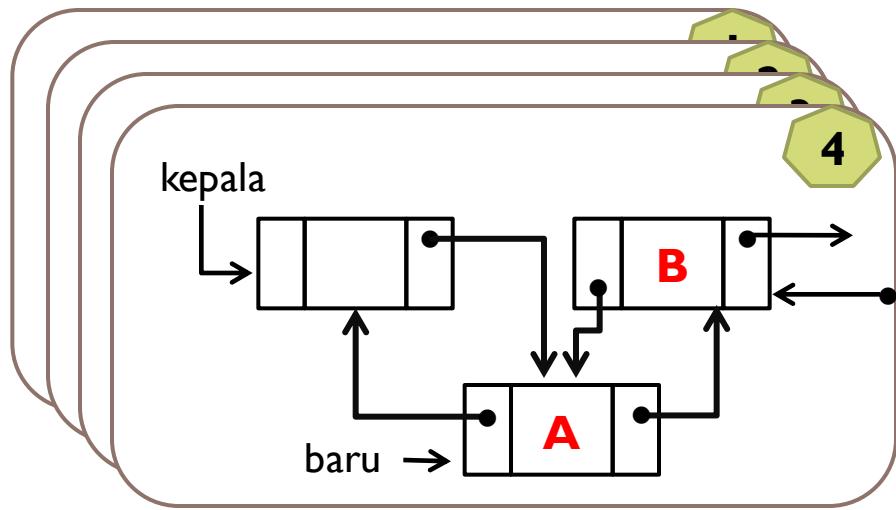


# MENAMBAH SIMPUL

## I. TAMBAH DEPAN



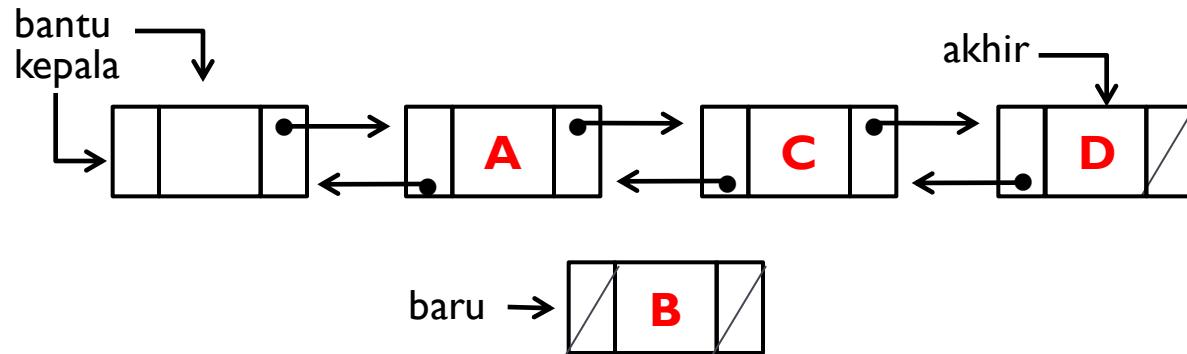
```
Type Simpul = ^Data;  
Data = record  
    Isi : char;  
    Kiri, Kanan : Simpul;  
end;  
var Elemen : char;  
Kepala : Simpul;
```



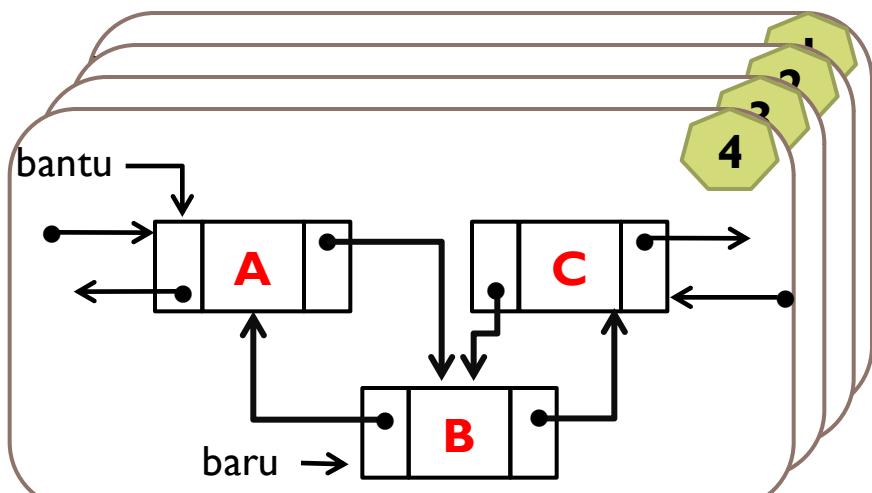
```
Procedure TAMBAH(var Kepala : Simpul;  
Elemen : char);  
  
var Baru : Simpul;  
begin  
    new(Baru);  
    Baru^.Isi := Elemen;  
    Baru^.kanan := Kepala^.kanan; { no. 1 }  
    Baru^.kiri := Kepala; { no. 2 }  
    Kepala^.kanan^.kiri := Baru; { no. 3 }  
    Kepala^.kanan := Baru; { no. 4 }  
end;
```

# MENAMBAH SIMPUL

## 2. TAMBAH TENGAH



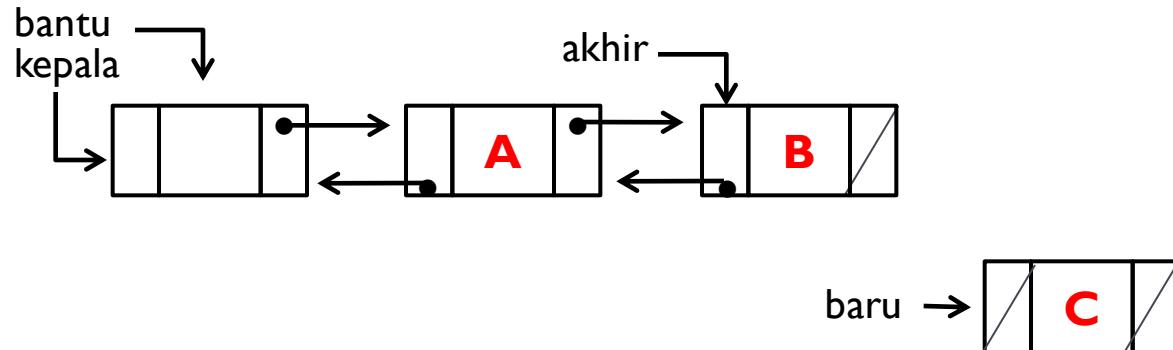
```
Type Simpul = ^Data;  
Data = record  
    Isi : char;  
    Kiri, Kanan : Simpul;  
end;  
var Elemen : char;  
Kepala : Simpul;
```



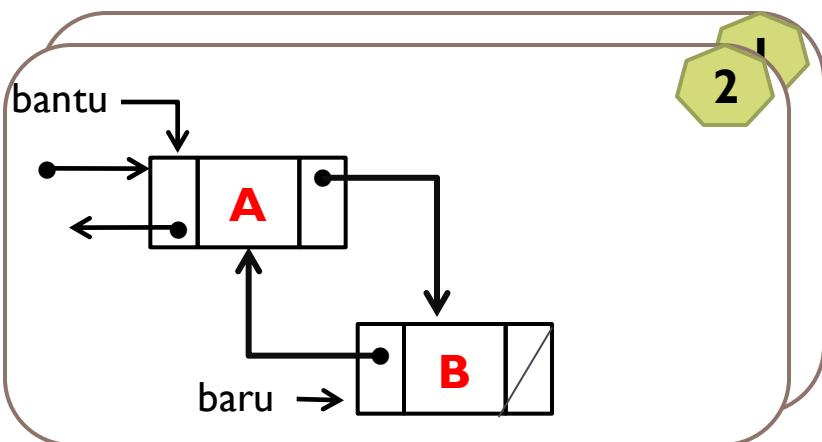
```
Procedure TAMBAH(var Kepala : Simpul;  
Elemen : char);  
  
var Baru, Bantu : Simpul;  
begin  
    new(Baru);  
    Baru^.Isi := Elemen;  
    while Bantu^.Kanan^.Isi < Elemen do  
        Bantu := Bantu^.Kanan;  
  
    Baru^.kanan := Bantu^.kanan; { no. 1 }  
    Baru^.kiri := Bantu; { no. 2 }  
    Bantu^.kanan^.kiri := Baru; { no. 3 }  
    Bantu^.kanan := Baru; { no. 4 }  
end;
```

# MENAMBAH SIMPUL

## 3. TAMBAH BELAKANG



```
Type Simpul = ^Data;  
Data = record  
    Isi : char;  
    Kiri, Kanan : Simpul;  
end;  
var Elemen : char;  
Kepala : Simpul;
```

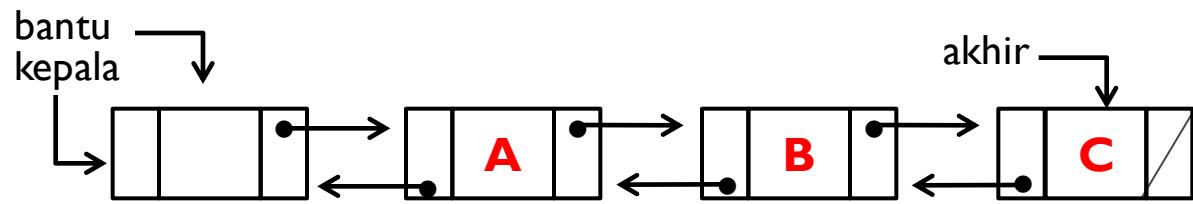


```
Procedure TAMBAH(var Kepala : Simpul;  
Elemen : char);  
  
var Baru, Bantu : Simpul;  
begin  
    new(Baru);  
    Baru^.Isi := Elemen;  
  
    while Bantu^.Kanan^.Isi < Elemen do  
        Bantu := Bantu^.Kanan;  
  
    Baru^.Kiri := Bantu; { no. 1 }  
    Bantu^.Kanan := Baru; { no. 2 }  
end;
```

# MENAMBAH SIMPUL

```
Procedure TAMBAH(var Kepala :Simpul;Elemen : char) ;  
  
var Baru,Bantu : Simpul;  
  
begin  
    new(Baru);                      {*Alokasi simpul baru *}  
    with Baru^ do                   {* Pengisian data dan inisialisasi *}  
        begin  
            Isi := Elemen;  
            Kiri := nil;  
            Kanan:= nil  
        end;  
  
    bantu := Kepala;  
  
    while Bantu^.Kanan^.Isi < elemen do      {* Penempatan posisi var Bantu *}  
        Bantu := Bantu^.Kanan;  
  
    if Bantu^.kanan <> nil then           {* Penyisipan di tengah *}  
        Begin  
            Baru^.kanan := Bantu^.Kanan;  
            Bantu^.Kanan^.kiri := Baru;  
        end;  
  
    Bantu^.kanan := Baru;                  {* Menambah di belakang *}  
    Baru^.kiri := Bantu  
end;
```

# MENGHAPUS SIMPUL



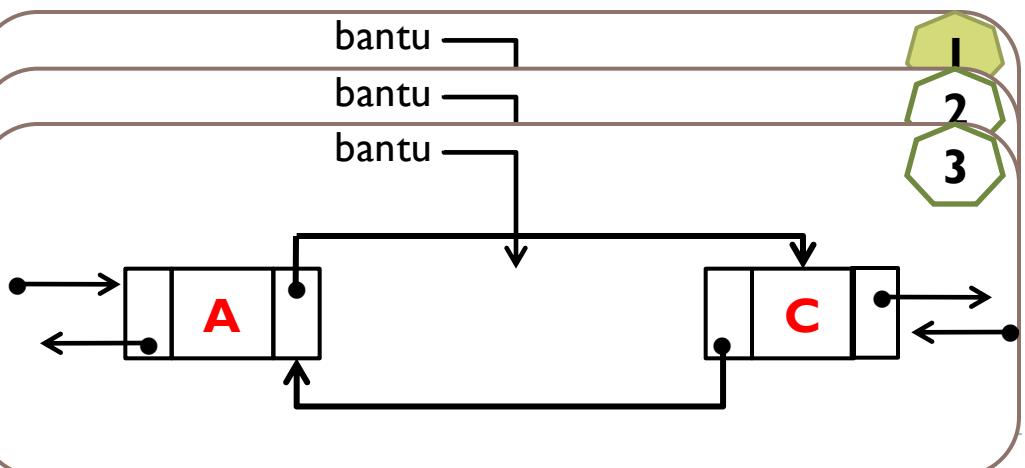
```
Type Simpul = ^Data;
Data = record
  Isi : char;
  Kiri, Kanan : Simpul;
end;
var Elemen : char;
Kepala : Simpul;
```

HAPUS\_SIMPUL(Kepala,'B')

```
Procedure HAPUS_SIMPUL(var Kepala : Simpul; Elemen : char);
```

```
var Bantu : Simpul;
Begin
  repeat
    Bantu := Bantu^.Kanan
  until (Bantu^.Isi = Elemen) or (Bantu = Kepala);

  if Bantu^.Isi = Elemen then
    begin
      { no. 1} Bantu^.Kiri^.Kanan := Bantu^.Kanan;
      { no. 2} Bantu^.Kanan^.Kiri := Bantu^.Kiri;
      { no. 3} dispose(Bantu); Bantu := Kepala
    end;
  end;
```



# MENGHAPUS SIMPUL

```
Procedure HAPUS_SIMPUL(var Awal,Ahir :Simpul; Elemen :char) ;  
  
var Bantu : Simpul;  
  
Begin  
    if Kepala^.Kanan = Kepala then  
        writeln('Senarai KOSONG') {* Senarai Kosong *}  
    else  
        begin  
            Bantu := Kepala; {* Jika Senarai isi *}  
  
            repeat (* Mencari simpul yang akan dihapus *)  
                Bantu := Bantu^.Kanan  
            until (Bantu^.Isi = Elemen) or (Bantu = Kepala);  
  
            if Bantu^.Isi = Elemen then  
                begin  
                    Bantu^.Kirj^.kanan := Bantu^.Kanan;  
                    Bantu^.Kanan^.Kiri := Bantu^.Kirj;  
                    dispose(Bantu);  
                    Bantu := Kepala;  
                end  
            else  
                writeln('Karakter yang akan dihapus TIDAK ADA')  
        end  
    end;  
end;
```

# Linked List

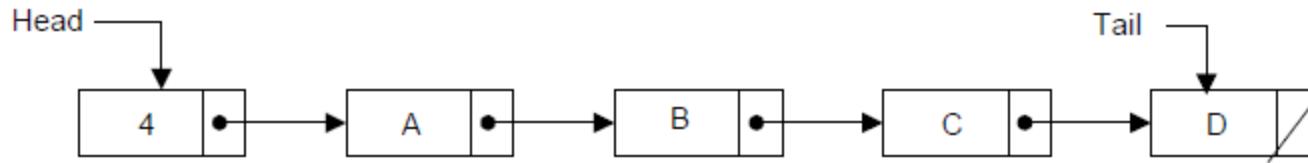
---

- ▶ **Single Linked List**
  - ▶ Disebut demikian karena pada setiap simpul hanya memiliki satu buah field yang berhubungan dengan simpul berikutnya
- ▶ **Double Linked List**
  - ▶ Linked List ini memiliki dua buah field yang digunakan untuk menunjuk ke simpul sebelumnya dan ke simpul sesudahnya.
  - ▶ Banyak digunakan untuk mempermudah proses pencarian simpul dalam suatu senarai berantai.

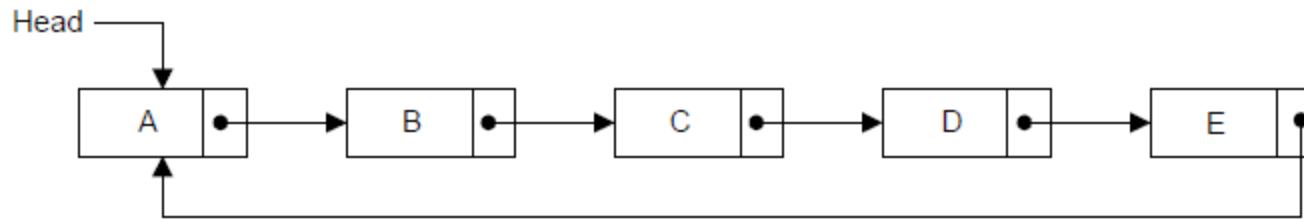


# Variasi Linked List

## ▶ Header Single Linked List



## ▶ Circular Single Linked List



## ▶ Header Circular Single Linked List

