

PERTEMUAN 10

Linked List

Hermanto, S.Kom. M.Kom



Pendahuluan

1. Dalam suatu linier list kita dapat melakukan operasi penyisipan atau penghapusan atas elemen-elemennya pada sembarang posisi.
2. Misalkan ada 1500 item yang merupakan elemen dari suatu linear list.
3. Jika elemen ke-56 akan kita keluarkan, maka elemen ke-1 s/d elemen ke-55 tidak akan berubah posisinya pada linier list tersebut. Tetapi elemen ke-57 akan menjadi elemen ke-56, element ke-58 akan menjadi elemen ke-57 dst. Selanjutnya, jika kita sisipkan satu elemen pada posisi setelah elemen ke-41, maka elemen ke-42 s/d elemen ke-1500 akan berubah posisinya.
4. Untuk menyatakan keadaan diatas diperlukan suatu konsep berbeda dengan konsep sekuensial sebelumnya.
5. Linked list merupakan suatu cara non-sekuensial yang digunakan untuk merepresentasikan suatu data.

Mengenal Struktur Data Linked List

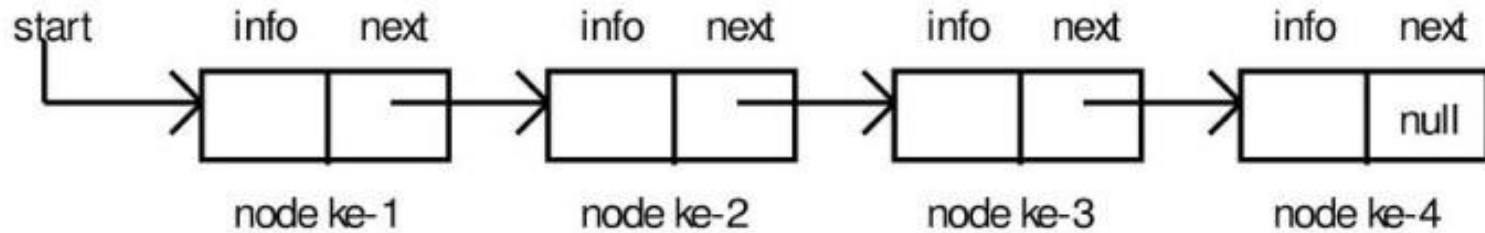
Linked List (one way list) adalah jenis struktur data yang berisi suatu kumpulan elemen data (yang disebut sebagai node) yang disusun secara linear dengan setiap data disimpan dalam sebuah simpul dimana urutan antara satu simpul dengan simpul lain dihubungkan melalui pointer.

Setiap elemen (node) dari suatu linked list terdiri atas dua bagian, yaitu:

- **INFO**, berisi informasi tentang elemen data yang bersangkutan.
- **NEXT** (link field/next pointer field), berisi alamat dari elemen node) selanjutnya yang dituju.

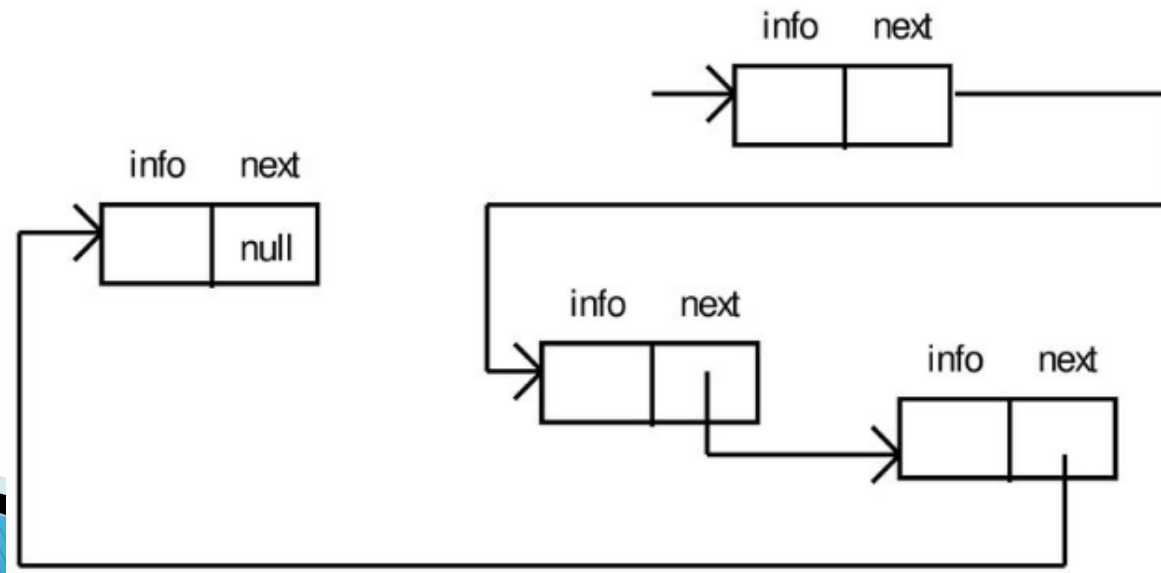
Mengenal Struktur Data Linked List

Berikut ini sebuah contoh linked list yang terdiri atas 4 node:



Pada node ke-4 field NEXT-nya berisi NULL, artinya node ke-4 tersebut adalah node terakhir.

Node-node dalam linked list tidak harus selalu digambarkan paralel seperti pada gambar diatas. Linked list pada contoh diatas dapat pula digambarkan seperti berikut ini:



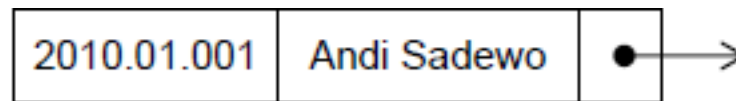
Mengenal Struktur Data Linked List

Struktur data ini mempunyai bentuk dasar dengan sifat data disisipkan ke dalam list melalui salah satu ujungnya. Gambar dibawah ini menunjukkan keadaan dalam linked list apabila secara berturut-turut nama Amir, Bakdi, Cintya dan Dhanu dimasukkan ke dalam linked list.

pertama



Dalam terminologi linked list, setiap data diletakkan dalam sebuah simpul (node). Pada gambar di atas terdapat 4 buah simpul. Setiap simpul terdiri atas 2 bagian, yaitu bagian data (info) dan bagian penunjuk ke simpul berikutnya (next). Pada contoh di atas bagian data hanya mengandung sebuah data, yaitu nama orang. Namun bagian data bisa saja mengandung beberapa data. Misalnya bagian data terdiri atas nomor mahasiswa dan nama mahasiswa seperti contoh berikut :

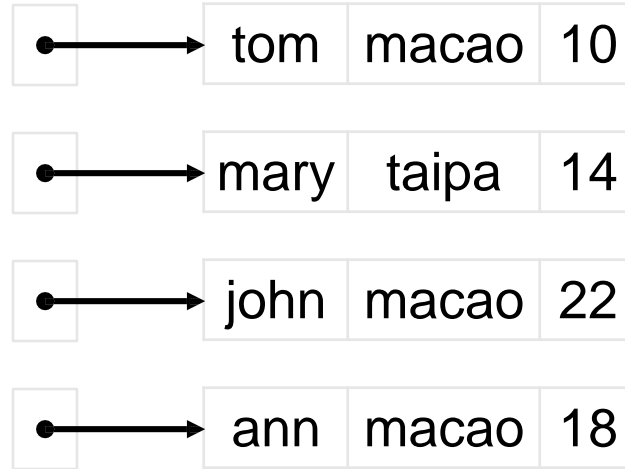


*Berisi nomor mahasiswa
Dan Nama mahasiswa*

- Pointer yang digunakan

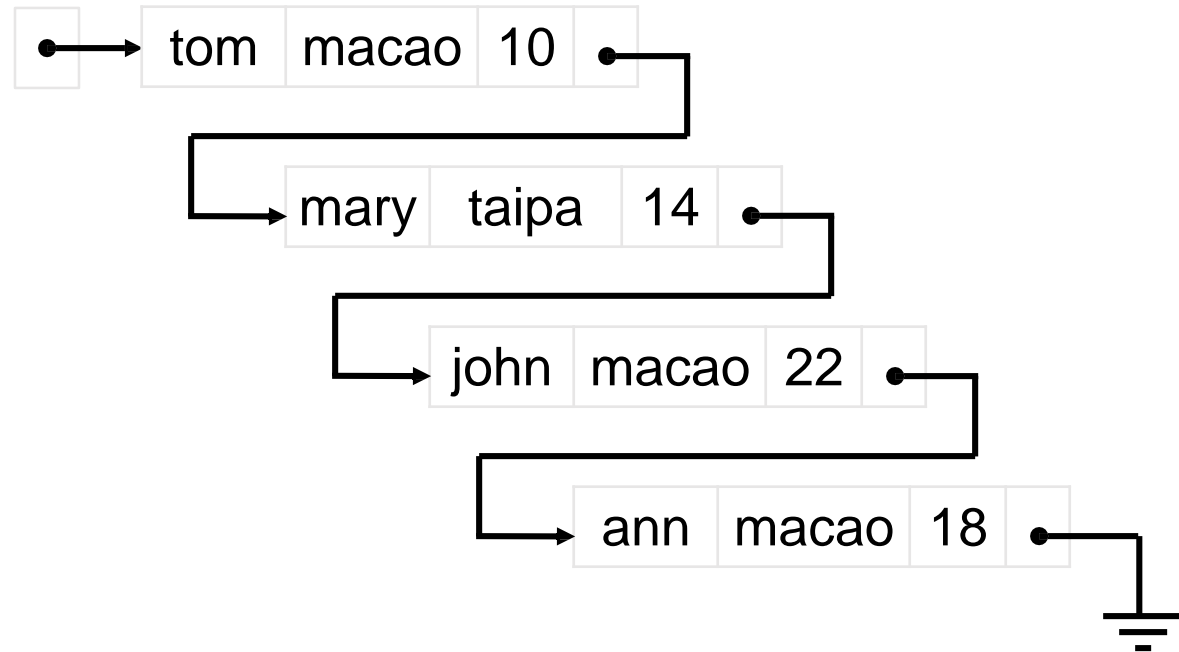
• - Berisi NULL kalau tidak menunjuk ke simpul.

Mengenal Struktur Data Linked List



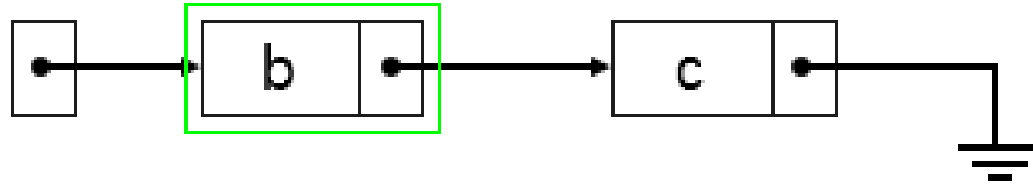
Misalnya kita buat banyak record student sebagai variabel dinamik, kita perlu mengorganisasi data kita, supaya efisien

Mengenal Struktur Data Linked List



Salah satu cara adalah menghubungkan semua record student dengan pointer

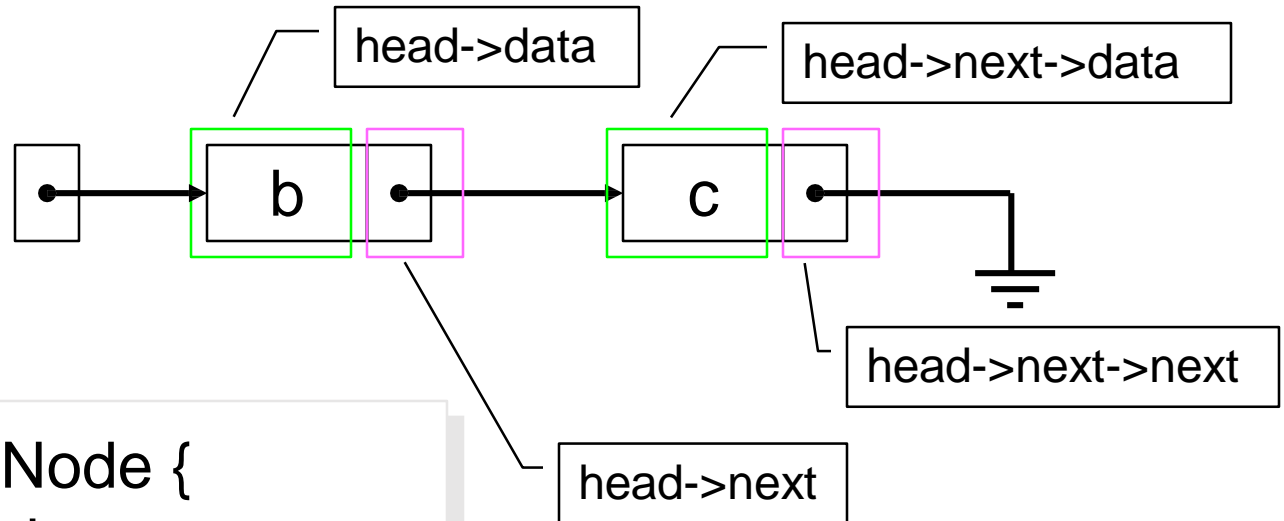
Linked List



```
struct Node {  
    char data;  
    Node *next;  
};
```

```
Node *head;
```

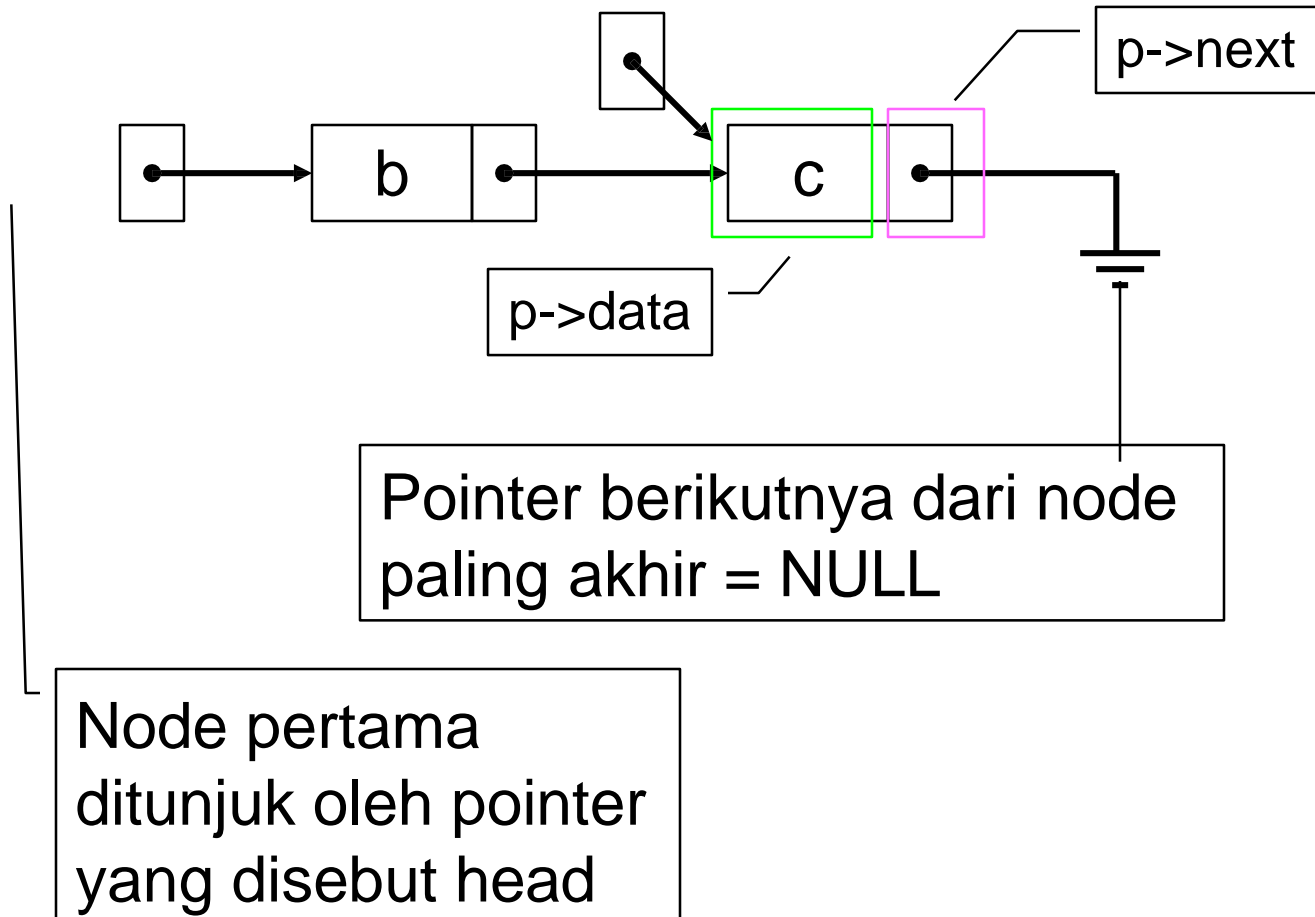

Linked List



```
struct Node {  
    char data;  
    Node *next;  
};
```

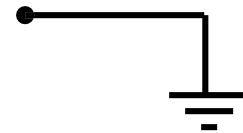
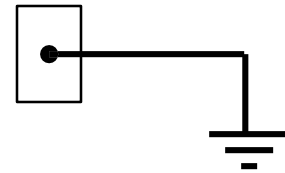
```
Node *head;
```

Linked List



Linked List Kosong

Linked List yang masih kosong dinyatakan dengan pointer head = NULL. Ingat bahwa head menunjuk ke node pertama linked list. Jika list kosong node pertama tidak ada

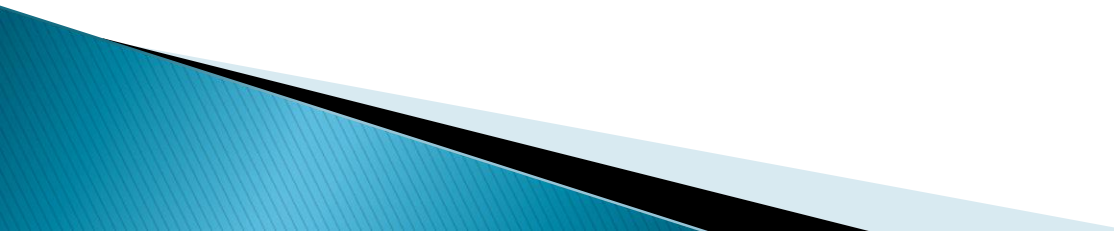


Mengenal Struktur Data Linked List

- Ada dua hal yang menjadi **kerugian** dengan representasi suatu data dengan linked list ini, yaitu:

1. Diperlukan ruang tambahan untuk tempat field pointer.
2. Diperlukan waktu yang lebih banyak untuk mencari suatu node dalam linked list.

- Sedangkan **keuntungannya** adalah:

1. Jenis data yang berbeda dapat di-link.
 2. Operasi DELETE atau INSERT hanya dilakukan dengan mengubah pointer-nya saja.
- 

Operasi Dasar pada Linked List

- ❑ **INSERT()**, menyatakan operasi untuk memasukkan data ke dalam linked list pada posisi yang ditunjuk oleh pointer pertama. Operasi ini biasa dinyatakan dengan `insert(S, d)` atau biasa disingkat `insert(d)`. `S` menyatakan linked list dan `d` menyatakan item data yang dimasukkan ke dalam linked list `S`
- ❑ **FIND()**, menyatakan operasi untuk mencari suatu data dalam linked list. Operasi ini biasa dinyatakan dengan fungsi `find(pendahulu, x)`. nilai baliknya berupa `true` kalau data yang dicari (yaitu `x`) ada; atau `false` kalau data yang dicari tidak ada. Pada saat nilai balik bernilai `true`, `pendahulu` menunjuk simpul yang berada tepat sebelum simpul yang berisi data yang dicari.
- ❑ **DELETE()**, menyatakan operasi untuk menghilangkan sebuah simpul dari linked list. Operasi ini biasa dinyatakan dengan `delete(S, x)` atau `delete(x)` saja.

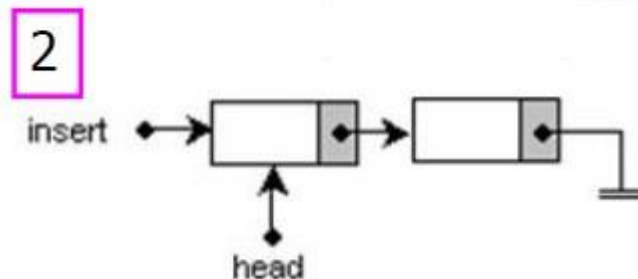
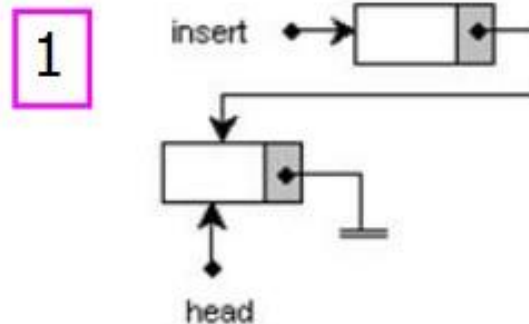
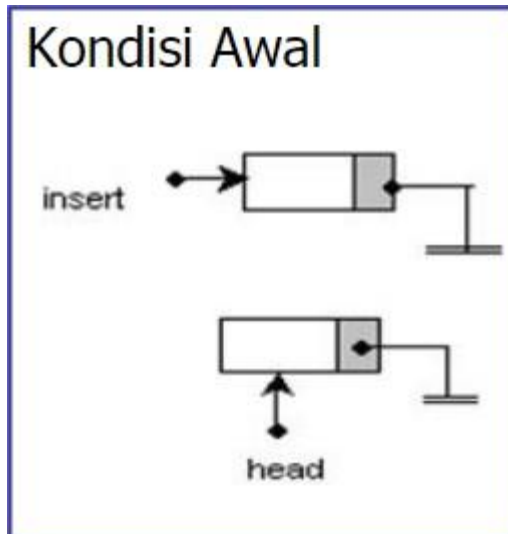
Operasi Insert

- ▶ Operasi yang terpenting pada linked list, yaitu menambahkan node (insert) dan menghapus node (delete).
- ▶ Fungsi insert pada linked list meliputi :
 - Insert sebagai node awal (head) dari linked list
 - Insert setelah node tertentu
 - Insert sebelum node tertentu
 - Insert sebagai node akhir (tail) dari linked list

Insert sebagai node awal (head) dari linked list

```
void insertashead(NODEPTR insert)
{
    insert->next=head;
    head = insert;
}
```

Ilustrasi dari fungsi diatas adalah sebagai berikut:



Insert setelah node tertentu

Statement untuk insert setelah node tertentu dari linked list adalah sebagai berikut :

```
void insertafternode(int x, NODEPTR insert)
{
    NODEPTR after ;
    after = head;
    while (after->info != x)
        after = after->next;
    insert->next = after->next;
    after->next = insert;
}
```

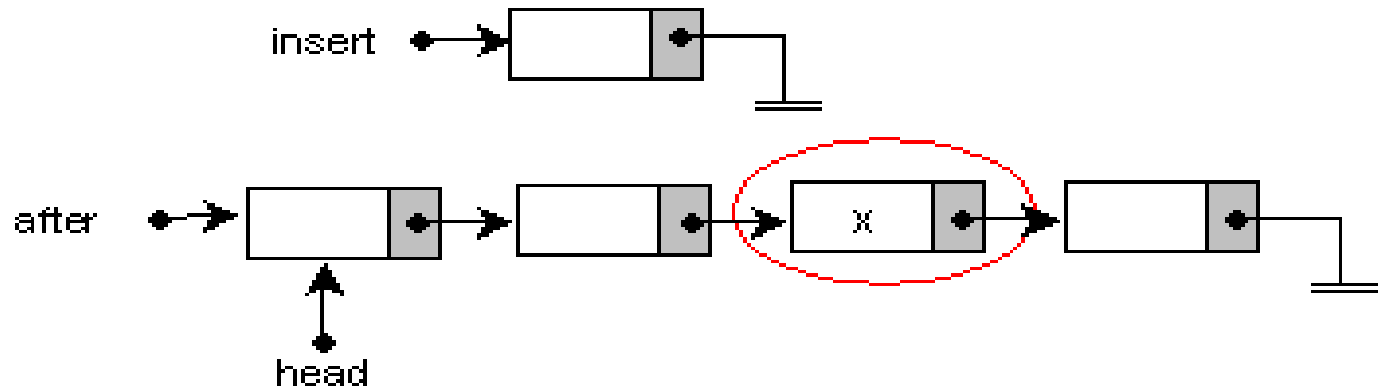
Langkah-langkah untuk proses di atas adalah sebagai berikut:

- Pointer next dari elemen baru menunjuk elemen setelah elemen tertentu
- Pointer next elemen sebelumnya menunjuk ke elemen baru

Insert setelah node tertentu

```
after = head;
```

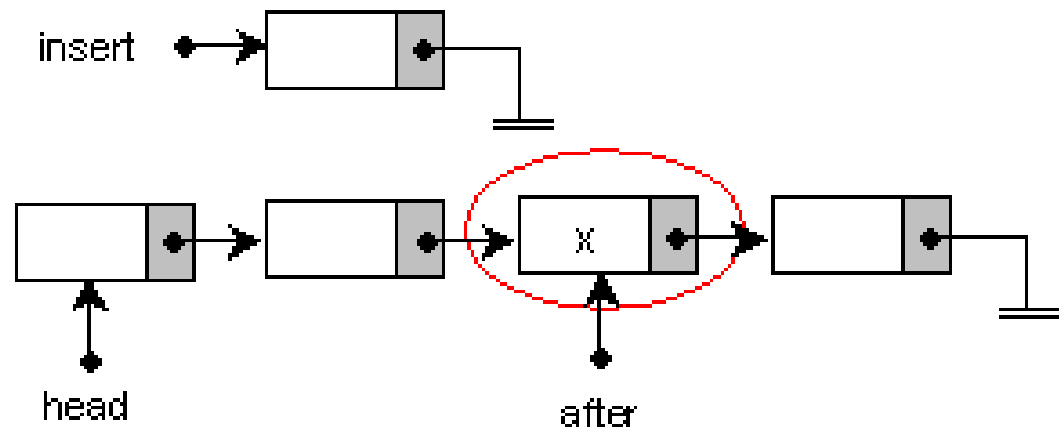
1



```
while (after->info != x)
```

```
after = after->next;
```

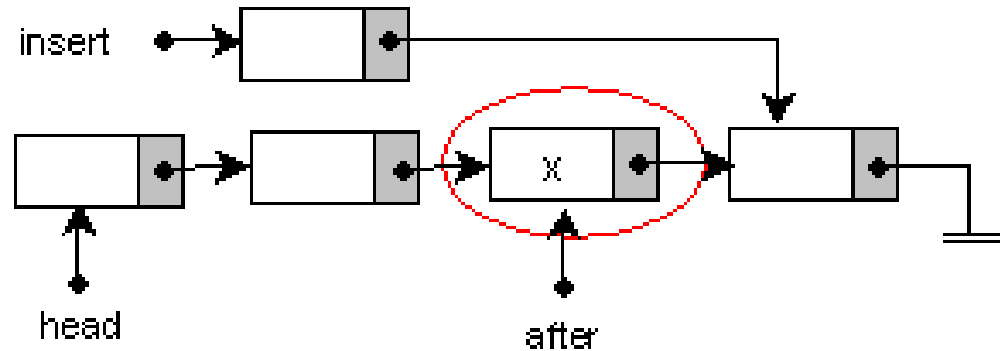
2



Insert setelah node tertentu

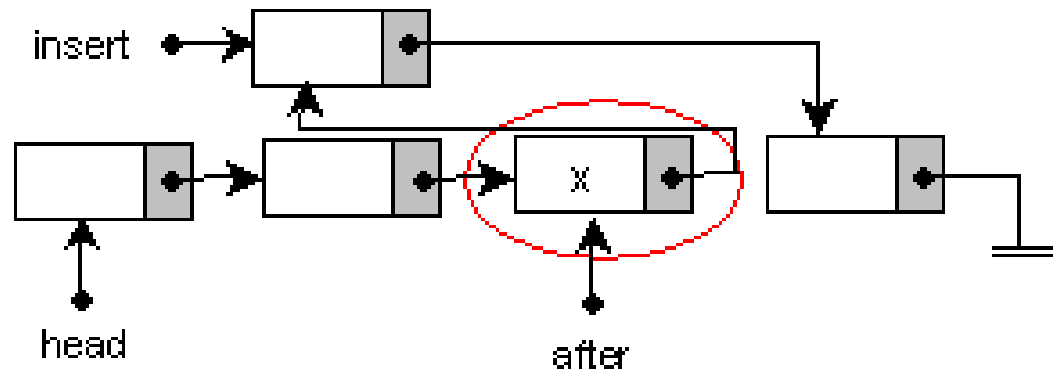
```
insert->next = after->next;
```

3



```
after->next = insert;
```

4



Insert sebelum node tertentu

```
void insertbeforenode(int x, NODEPTR insert)
```

```
{    NODEPTR before, prevbefore;
```

```
    if (head->info == x)
```

```
        insertashead(insert);
```

```
    else
```

```
    {    before = head;
```

```
        do
```

```
        {    prevbefore = before;
```

```
            before = before->next;
```

```
        }while (before->info != x);
```

```
        insert->next = before;
```

```
        prevbefore->next = insert;
```

```
    }
```

```
}
```

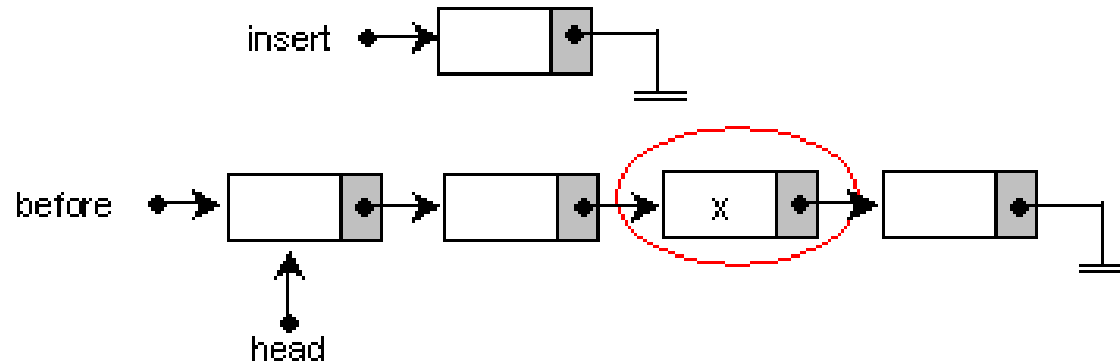
Jika data yang dicari
berada pada awal Linked
List

Jika data yang dicari tidak
berada pada awal Linked
List

- Telusuri list sampai elemen tertentu, catat juga elemen sebelumnya
- Lakukan penyisipan sebelum elemen tertentu tersebut

Insert sebelum node tertentu

```
before = head;
```



1

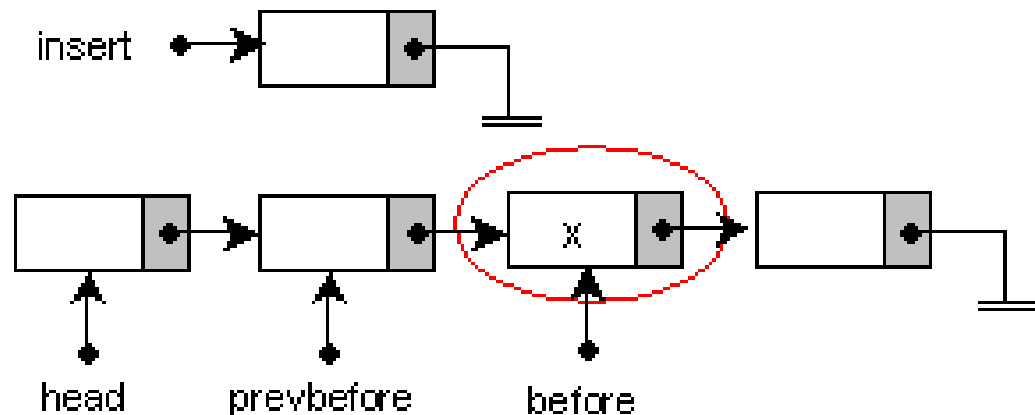
2

```
do
```

```
prevbefore = before;
```

```
before = before->next;
```

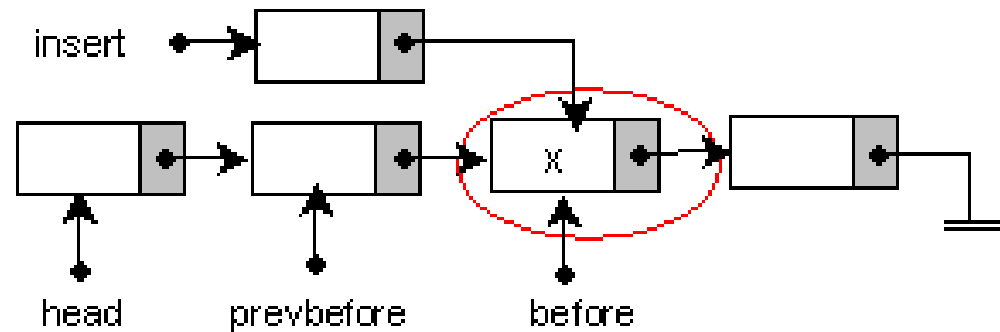
```
while (before->info != x);
```



Insert sebelum node tertentu

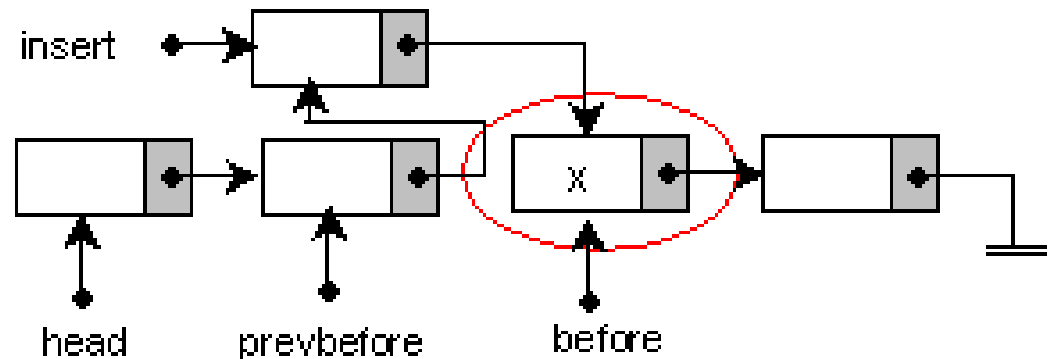
```
insert->next = before;
```

3



4

```
prevbefore->next = insert;
```



Insert sebagai node akhir (tail) dari linked list

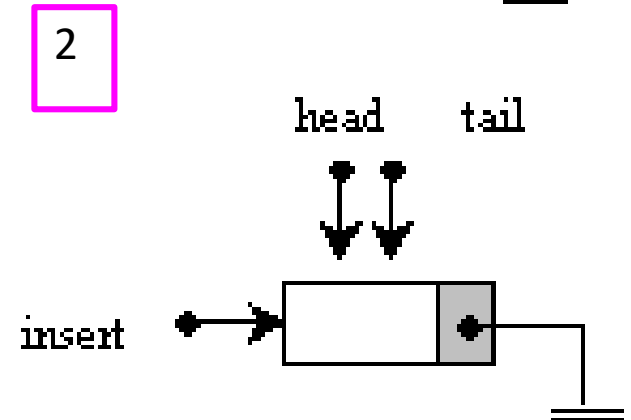
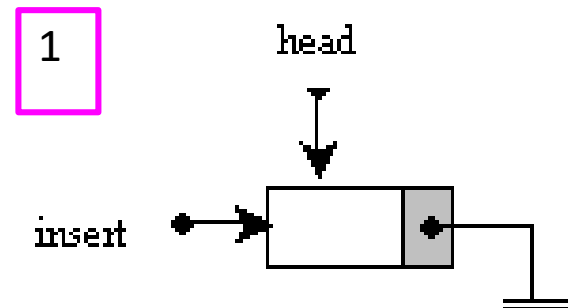
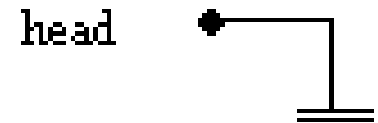
Langkah-langkah untuk proses ini adalah sebagai berikut :

- ❑ Telusuri list sampai elemen terakhir (tail->next=NULL)
- ❑ Lakukan pentisipan setelah elemen terakhir

```
void insertat tail(NODEPTR insert)
```

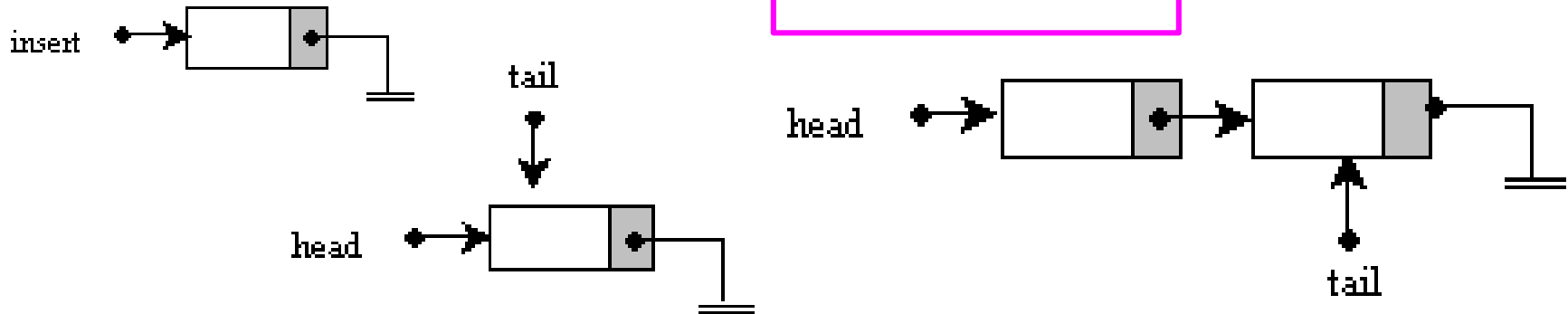
```
{  
    if (head==NULL){  
        1 head = insert ;  
        2 tail = head ;  
    }  
    else  
    {  
        tail->next = insert ;  
        tail = tail->next ;  
    }  
}
```

Kondisi Awal



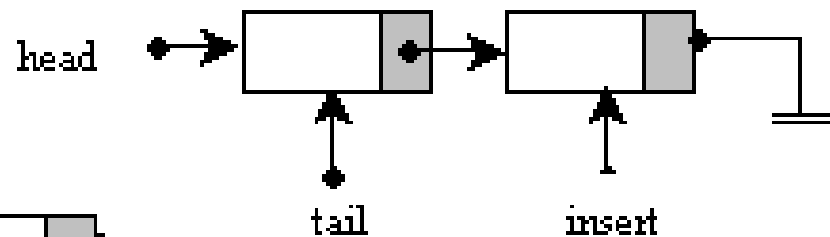
Insert sebagai node akhir (tail) dari linked list

Kondisi Awal

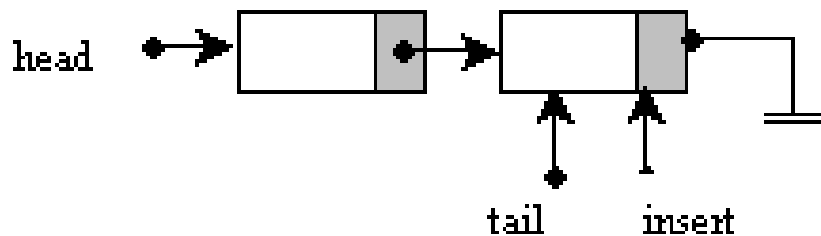


1. `tail->next = insert ;`
2. `tail = tail->next ;`

1



2



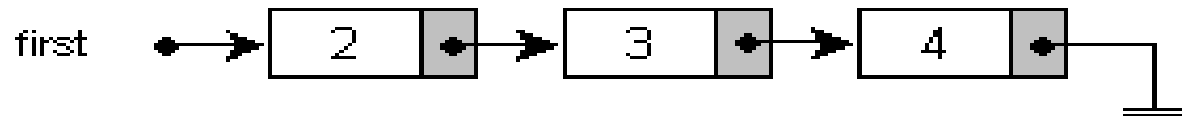
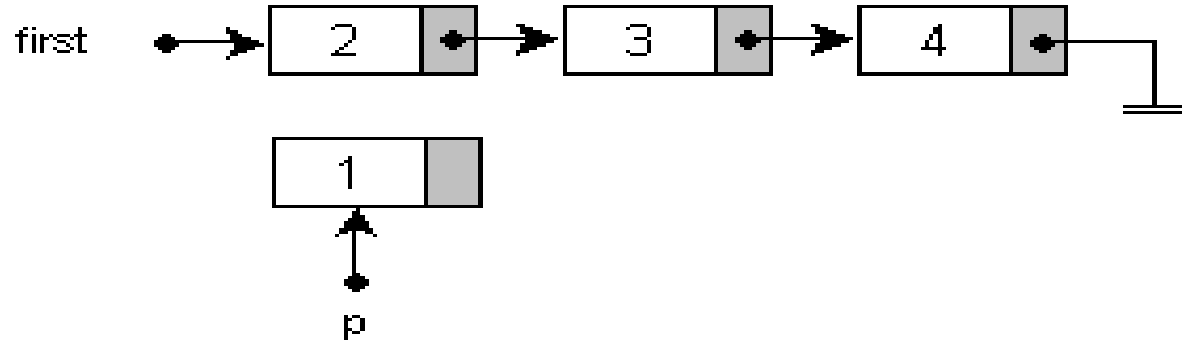
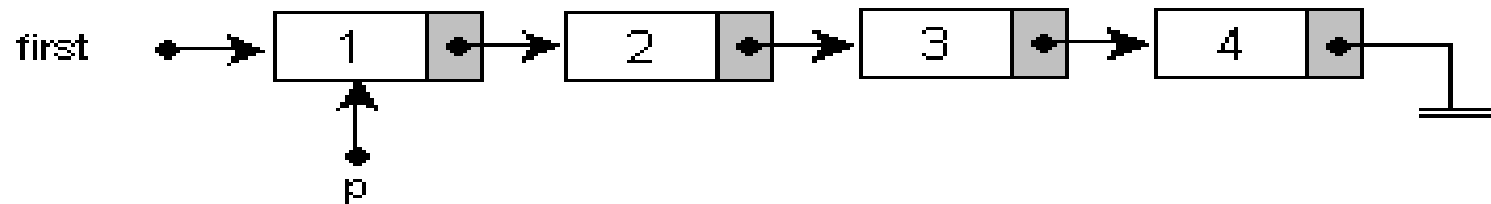
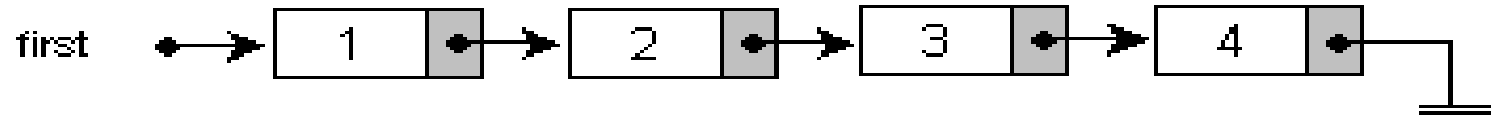
Operasi Delete

- Fungsi delete pada linked list meliputi :
 - Delete sebagai simpul pertama(head) dari linked list
 - Delete setelah simpul tertentu
 - Delete simpul terakhir

Penghapusan Simpul Pertama

- Langkah-langkah untuk proses di atas adalah sebagai berikut:
 - Pointer first diarahkan pada data ke-2
 - Pointer p diarahkan pada data ke-1
 - Bebaskan pointer p (secara otomatis data ke-1 terhapus)

Penghapusan Simpul Pertama

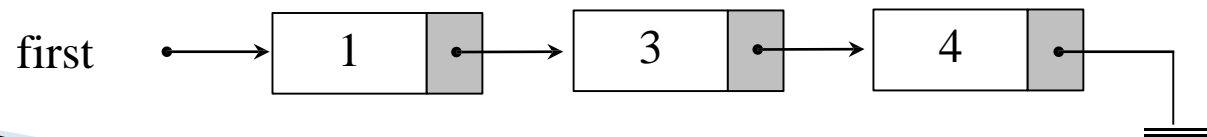
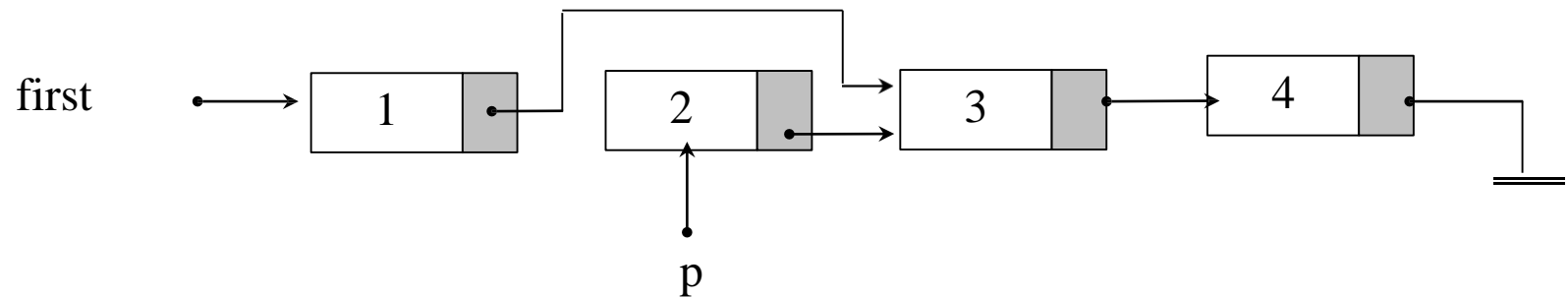
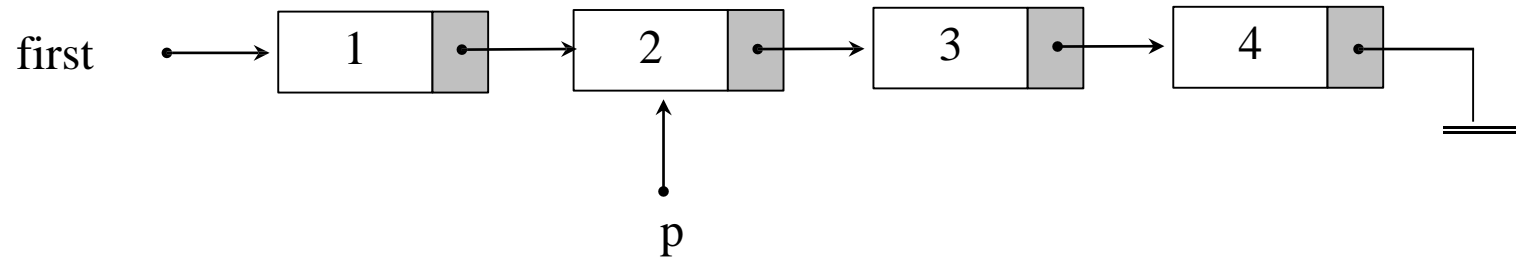
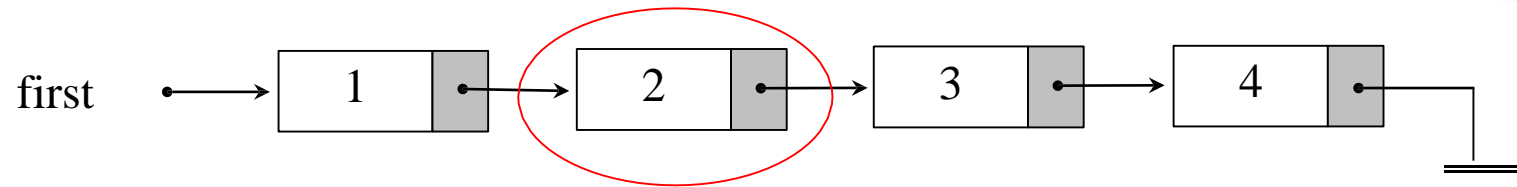


Penghapusan pada Node Tertentu – Data di Tengah

Langkah-langkah untuk proses ini adalah sebagai berikut:

- ☐ Arahkan pointer first s/d data yang ditunjuk
- ☐ Pointer p diarahkan pada first->next
- ☐ Arahkan pointer first->next pada p->next
- ☐ Bebaskan pointer p (secara otomatis data setelah simpul tertentu terhapus)

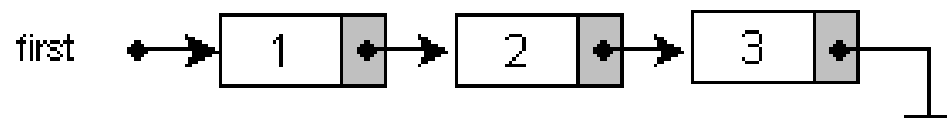
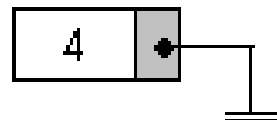
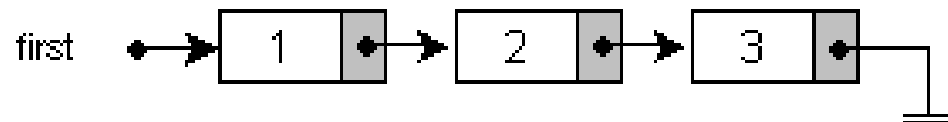
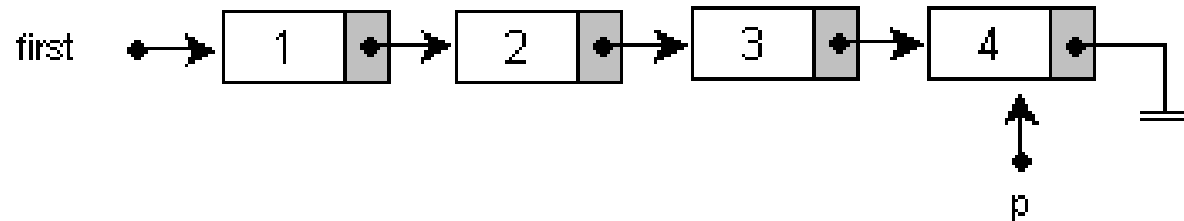
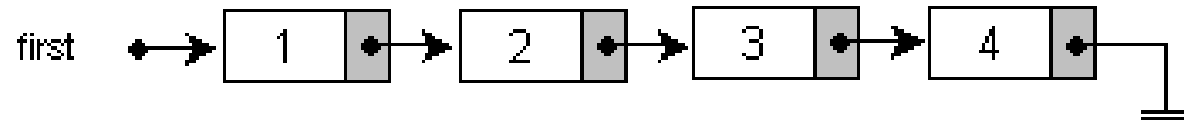
Penghapusan pada Node Tertentu – Data di Tengah



Penghapusan Simpul Terakhir

1. Telusuri simpul s/d $\text{first} \rightarrow \text{next} = \text{NULL}$
2. Arahkan pointer p ke first
3. Bebaskan pointer $p \rightarrow \text{next}$ (Simpul Terakhir)
4. Arahkan $p \rightarrow \text{next}$ ke NULL

Penghapusan Simpul Terakhir



```

void del_node(int data)
{
    NODEPTR del, beforedel ;
    del = head ;

    beforedel = del ;
    if (del->info == data )
    {
        if (del->next == NULL)
        {
            head = NULL ;
            free_node(del);
        }
        else
        {
            head = head -> next ;
            free_node(del);
        }
    }
}

```

jika data yang akan dihapus berada pada awal Linked List

jika data yang akan dihapus berada pada awal Linked List dan hanya ada satu data

jika data yang akan dihapus berada pada awal Linked List dan Linked List lebih dari satu data

else

{

while(del->info != data)

{

beforedel = del ;

del=del->next;

}

if (del->next == NULL)

{

beforedel->next=NULL;

free_node(del);

}

else

{

beforedel->next = del->next;

free_node(del);

}

}

jika data yang akan dihapus bukan pada awal Linked List

untuk mencari data yang akan dihapus

Jika data yg akan dihapus pada akhir Linked List

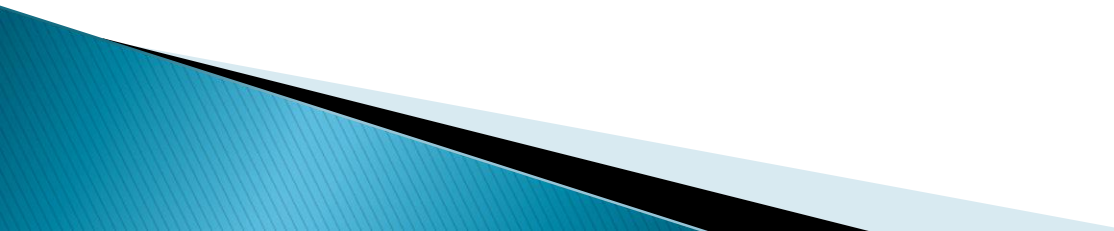
Jika data yg akan dihapus bukan pada akhir Linked List



Terima
kasih

Latihan

Soal

1. Untuk data yang bagaimanakah yang dapat direpresentasikan dengan menggunakan struktur data list linier?
 2. Sebutkan dan jelaskan perbedaan array dan linked list?
 3. Jelaskan operasi-operasi pada linked list!
- 

Latihan

Soal

4. Sebuah keluarga terdiri atas Ayah, Ibu, dan 2 orang anak. Identitas Ayah dan Ibu terdiri atas:

- Nama
- Umur
- Alamat
- Tanggal Lahir
- Pointer

Sedangkan identitas 2 anak tersebut terdiri atas:

- Nama
- Umur
- Alamat
- Tanggal Lahir
- Pointer ke Ayah
- Pointer ke Ibu

Buatlah struktur record yang merepresentasikan keluarga tersebut, lengkapi dengan ilustrasi gambar

