Single Liked-List

(TIB11 – Struktur Data)

Pertemuan 9, 10

Sub-CPMK

- Mahasiswa mampu membuat Single Linked-List dan mengakses data nya (C3, A3)

Materi

- Konsep Linked-List
- Menambahkan Node
- Mencari Node
- Menghapus Node
- Memindahkan Node

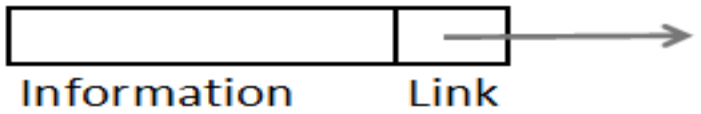
1.

Konsep Linked-List

Linked List

- Liked-List atau Senarai adalah sebuah urutan elemen yang terbatas yang diakses menggunakan pointer

  s1, s2, ..., sn
- Node/simpul : Record yang berisi informasi dan link ke node/simpul lainnya
- Linked List elements
  - Information
  - link: Penghubung ke node/simpul lain



Dua variable penting linked-list

- Head/Kepala: variabel yang berisi informasi address pointer dari node/simpul pertama
- CurrentCell / PointerCell: berisi informasi dari current node/simpul yang sedang diakses

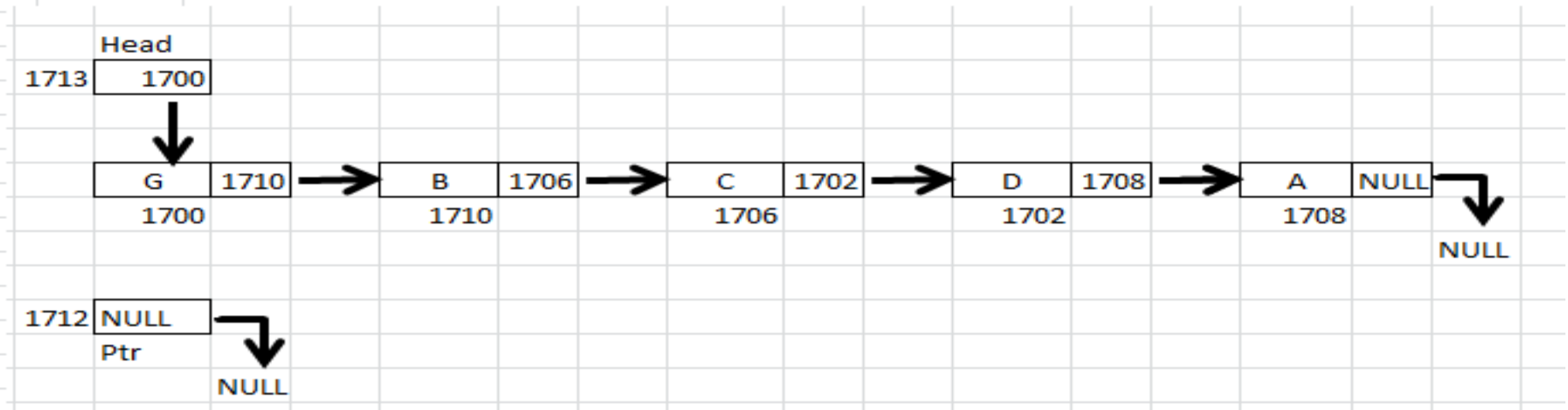
HARUS DIINGAT!!!

- Head/Kepala adalah variabel berisi informasi penting untuk mengarahkan linked-list
- Dengan ‘Head’ atau ‘Kepala’ kita dapat menuju ke node/simpul pertama dan bergerak maju ke simpul/node tujuan
- Pada saat anda kehilangan ‘Head’ berarti anda kehilangan linked list juga
- **Never ever lose your ‘HEAD’!!!**

Beberapa Variants Linked List yg Umum

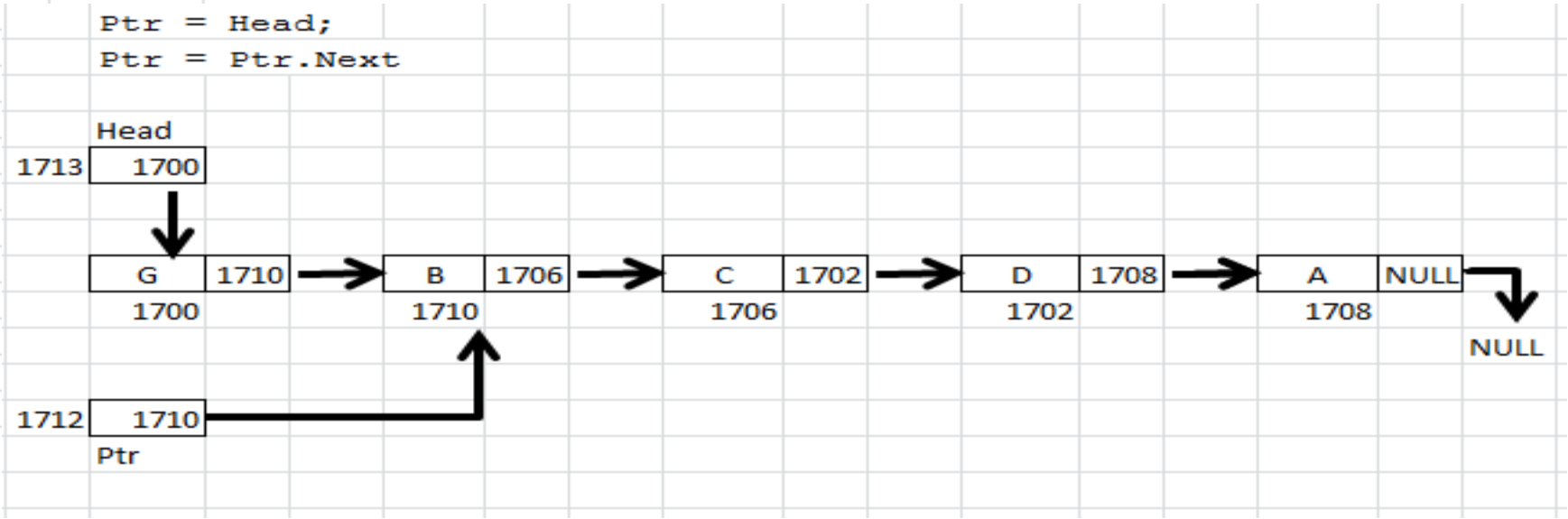
- Single Linked List
- Double Linked List
- Circular Linked List
- Multilevel List

	alamat memory	Isi Memory	
Head	1713	1700	← Kepala Linked List
Ptr	1712	NULL	← Current Node
	1711	1706	
	1710	B	
	1709	NULL	
	1708	A	
	1707	1702	
	1706	C	
	1705		
	1704		
	1703	1708	
	1702	D	
	1701	1710	
	1700	G	
	1699		
	1698		
	1697		

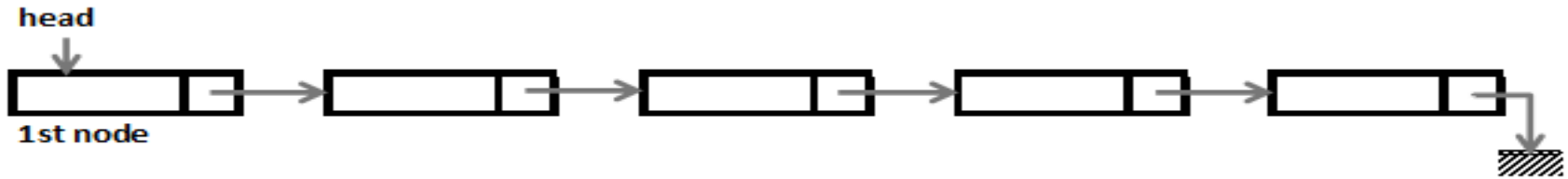


	alamat memory	Isi Memory	
Head	1713	1700	← Kepala Linked List
Ptr	1712	1700	← Current Node
	1711	1706	
	1710	B	
	1709	NULL	
	1708	A	
	1707	1702	
	1706	C	
	1705		
	1704		
	1703	1708	
	1702	D	
	1701	1710	
	1700	G	
	1699		
	1698		

```
Ptr = Head;
Ptr = Ptr.Next
```



Single Linked List



2.

Menambahkan Node

Linked List Operation

- Search / Locate
- Insert
  - Setelah current cell/simpul/node
  - Sebelum current cell/simpul/node
- Delete

Kemungkinan Operasi

- Pada bagian depan dari list
- Di tengah list
- Pada bagian akhir dari list

Insert Operation

- Pada bagian depan list

  Hanya dapat terjadi pada operasi insert sebelum current cell

- Pada bagian akhir list

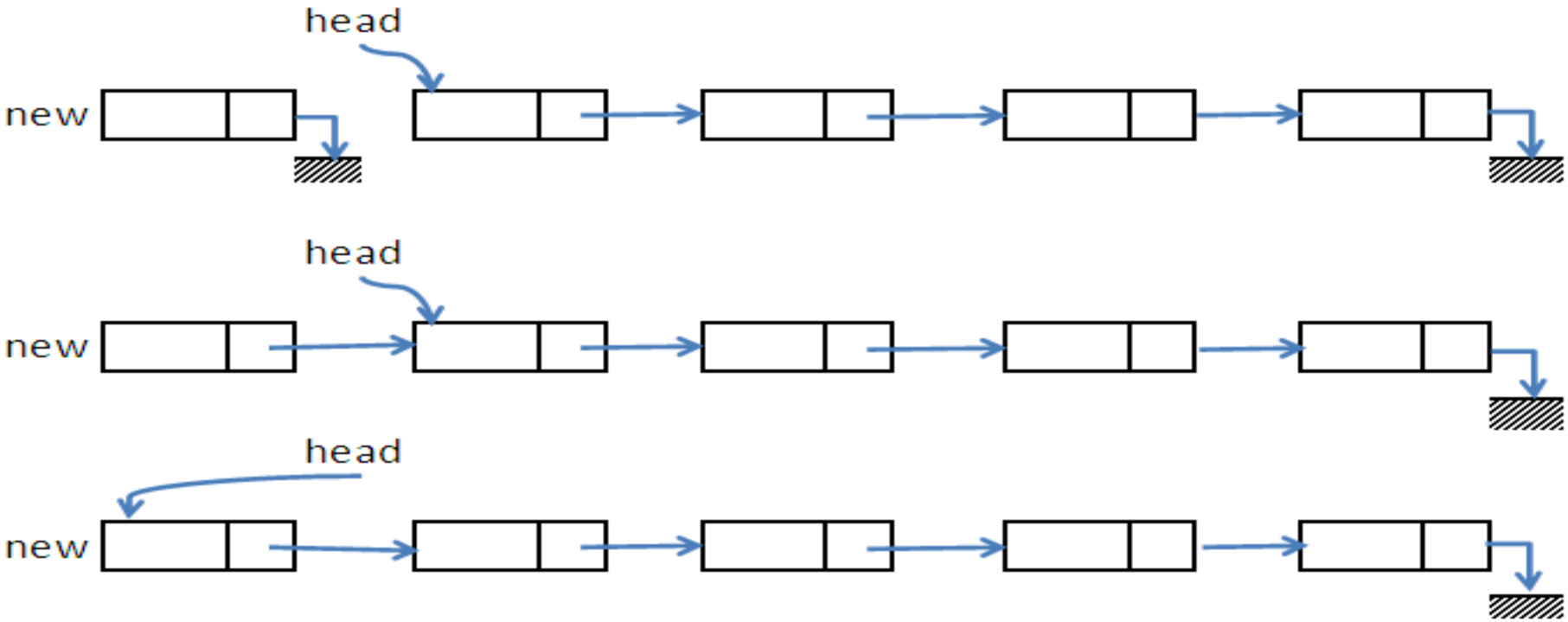
  Hanya dapat terjadi pada insert setelah current cell

- Pad abagian tengah middle list

Insert pada bagian depan list

- Buat sebuah simpul/node baru
- Isi informasi simpul/node baru tersebut
- Arahkan next link ke simpul/node kepala
- Set head pointer ke simpul/node baru tersebut

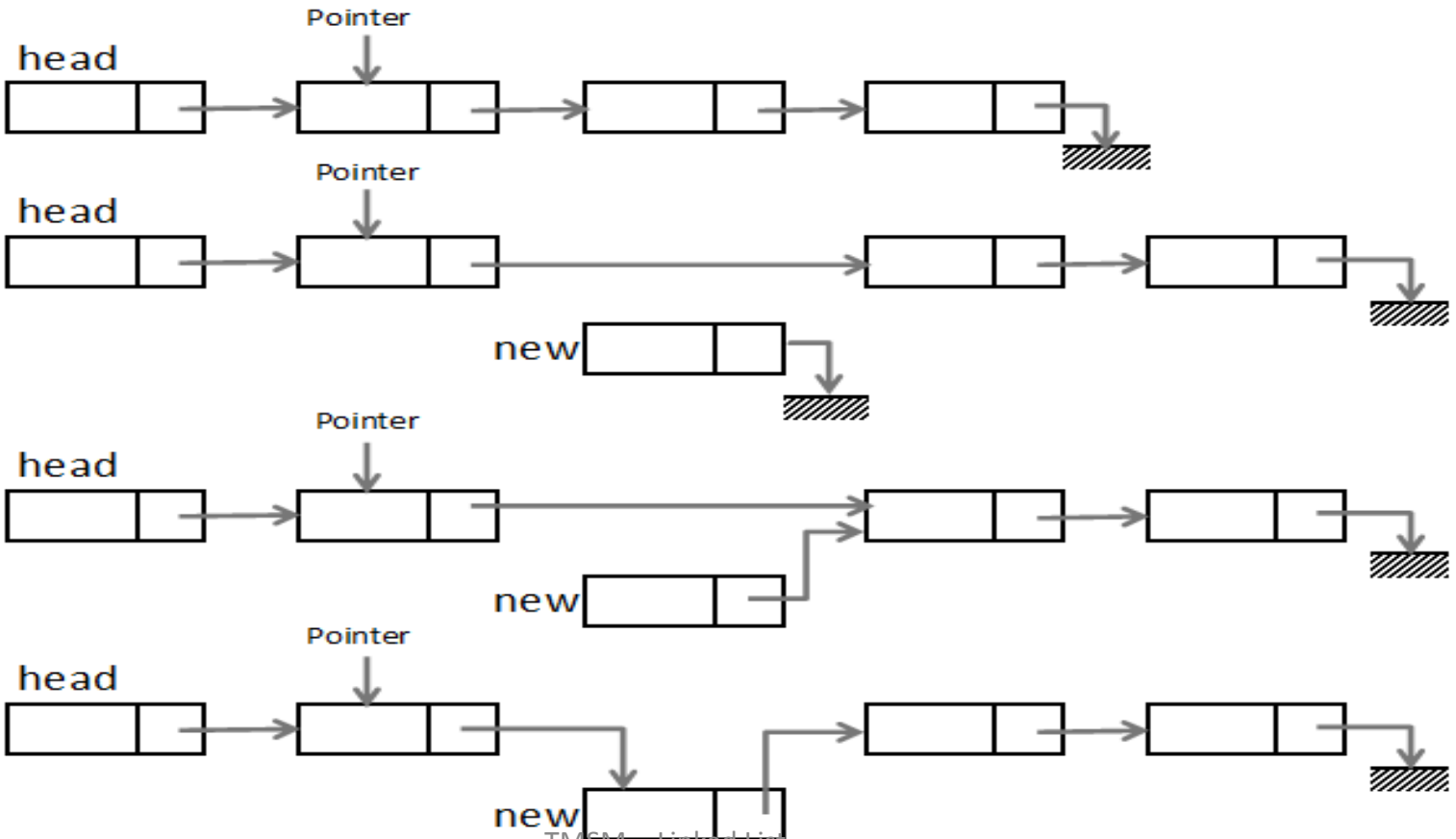
Insert pada bagian depan list (cont.)



Insert di tengah – setelah current cell

- Buat sebuah simpul/node baru
- Isi informasi simpul/node baru tersebut
- Copy next link dari current node/simpul ke next link simpul/node baru
- Set next link pada current simpul/node ke simpul/node baru

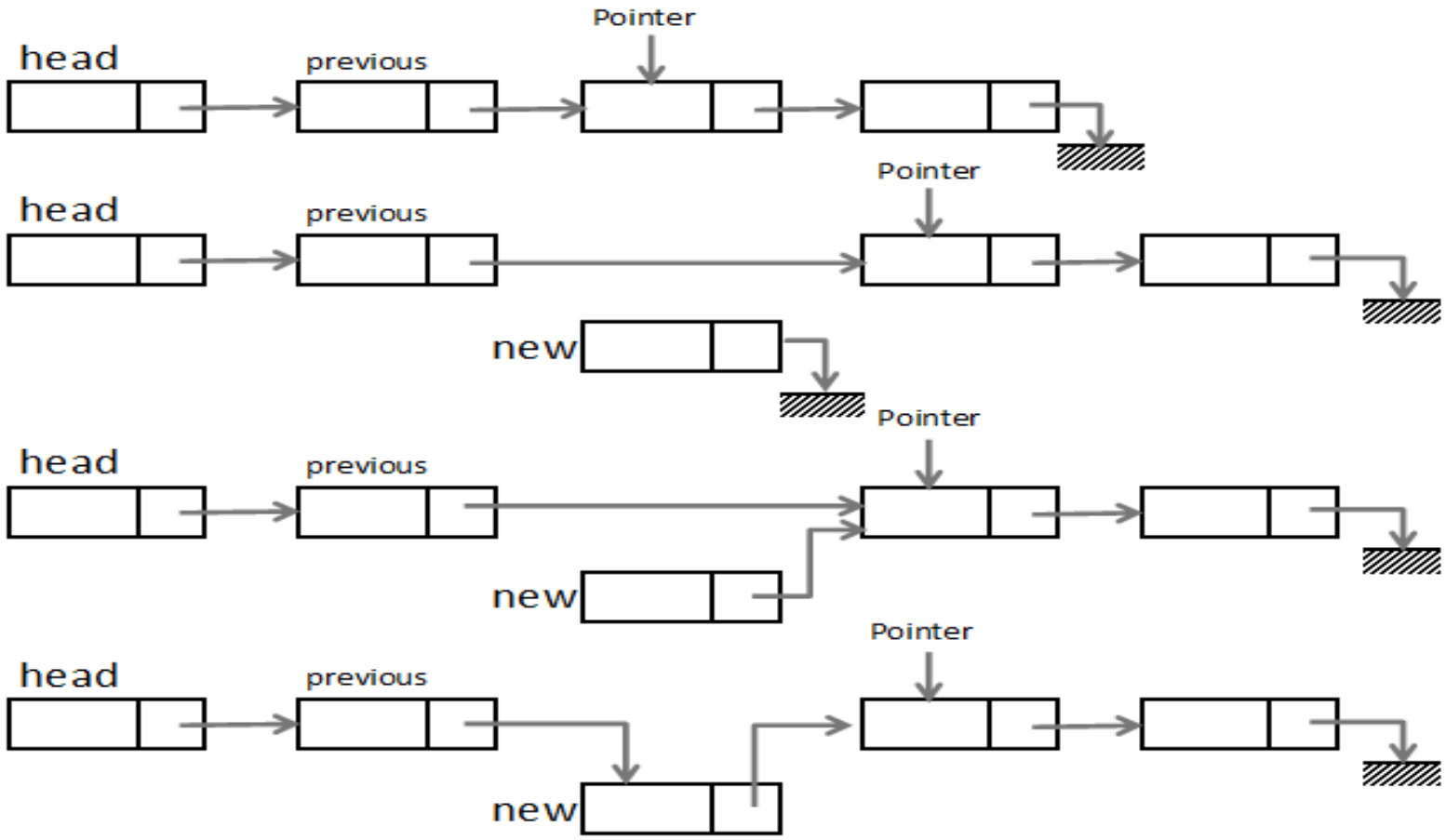
Insert di tengah – setelah current cell (cont.)



Insert di tengah – sebelum current cell

- Buat sebuah node baru
- Isi informasi node baru tersebut
- Copy next link dari previous node ke next link dari node baru
- Set next link pada previous node ke node baru

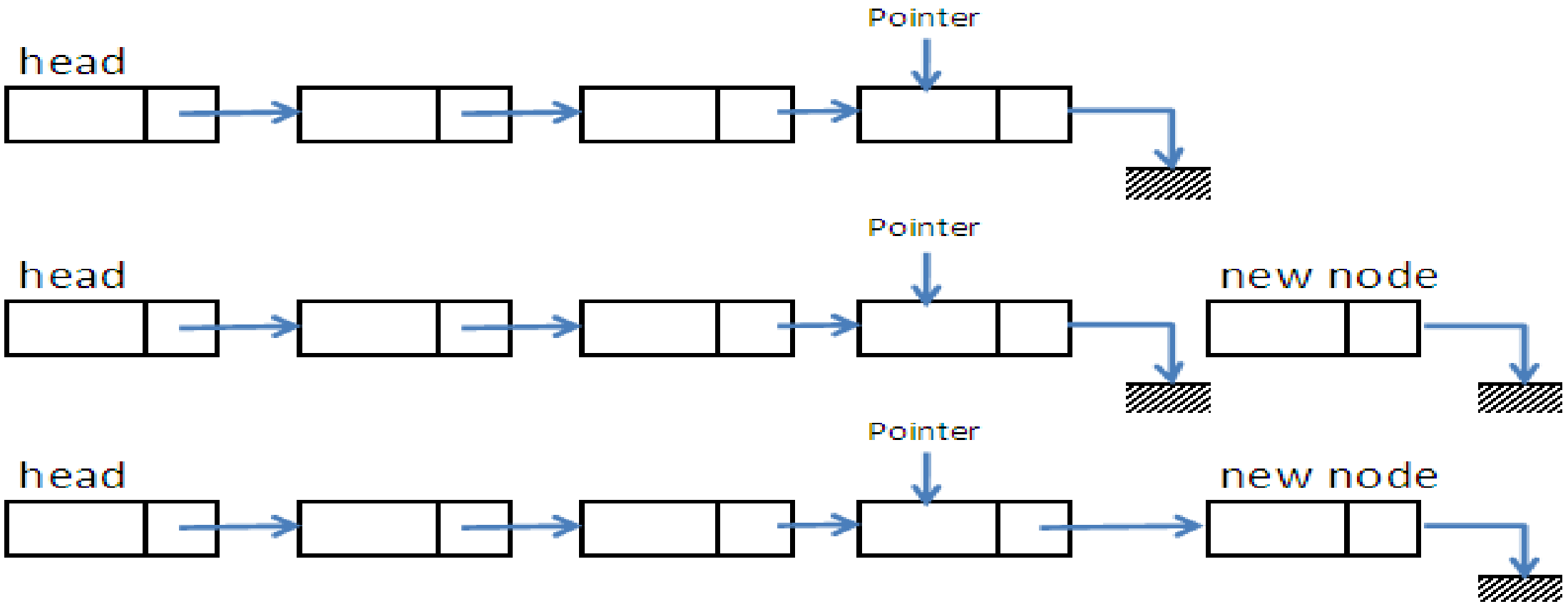
Insert di tengah – sebelum current cell (cont.)



Insert pada bagian akhir list

- Buat sebuah node baru
- Isi informasi node baru tersebut
- Set next link dari new node sebagai NULL
- Arahkan next link pada last node atau tail ke node baru

Insert at the end (cont.)



3.

Mencari Node

Locate Operation

- Assign PointerCell sebagai Head

```
PointerCell = Head;
```

- Bergerak maju dengan mengarahkan PointerCell ke next PointerCell sampai ditemukan node/simpul yang sesuai.

```
PointerCell = PointerCell->Next;
```

Locating Ke Previous Node

Dapat dilakukan dengan berbagai cara

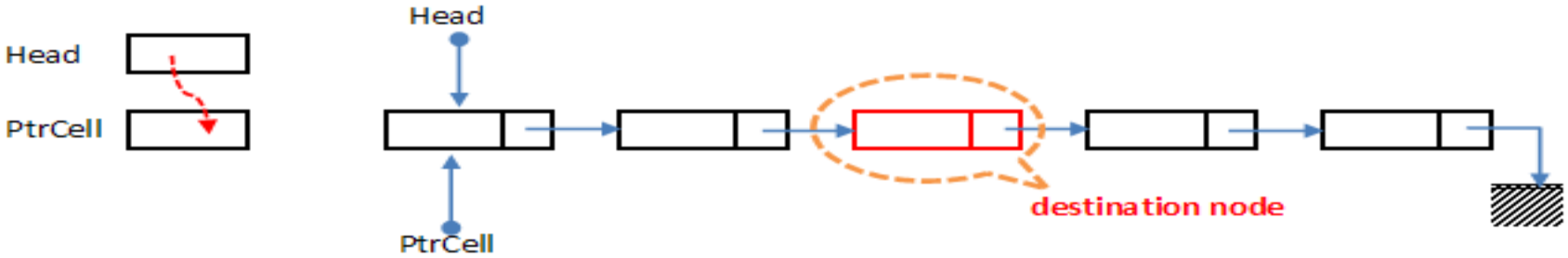
- Simpan previous node ketika locating the current node

```
PreviousNode = CurrentNode;
CurrentNode = CurrentNode->Next;
```

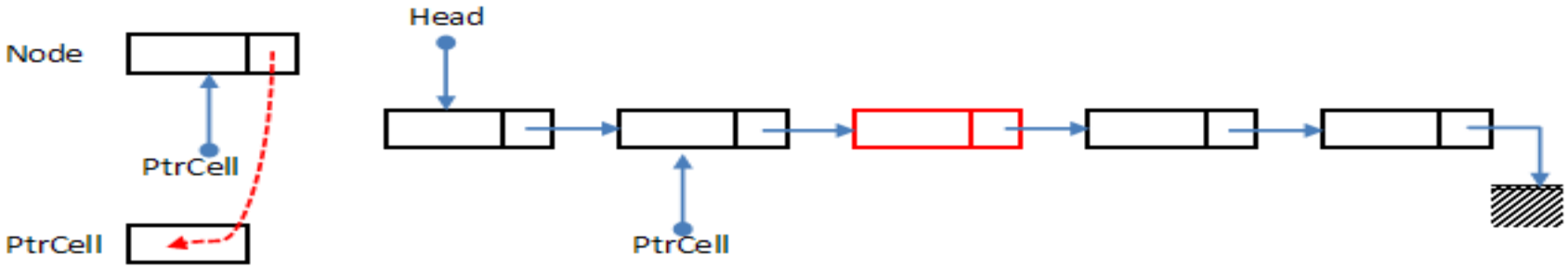
- Retrieve jika diperlukan

```
RetrieveNode = HeadNode;
While (RetrieveNode-> != CurrentNode)
{
   RetrieveNode = RetrieveNode->Next;
}
PreviousNode = RetrieveNode;
```

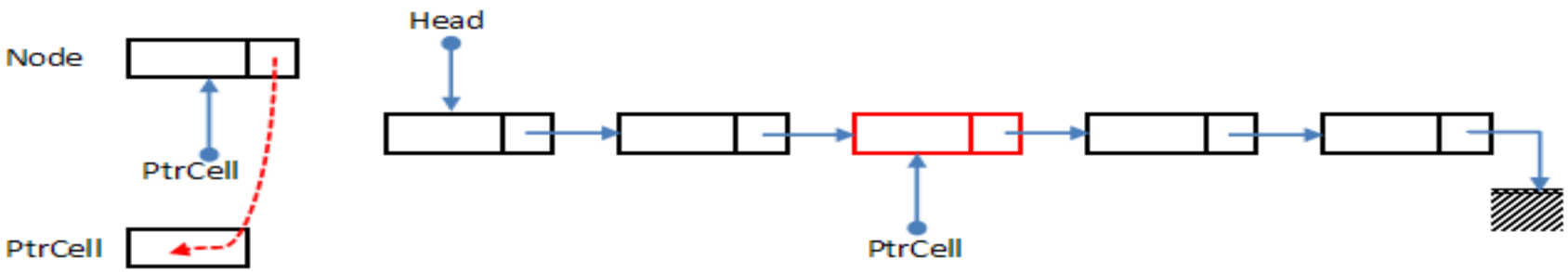
PtrCell = Head



PtrCell = PtrCell->Next



PtrCell = PtrCell->Next



4.

Menghapus Node

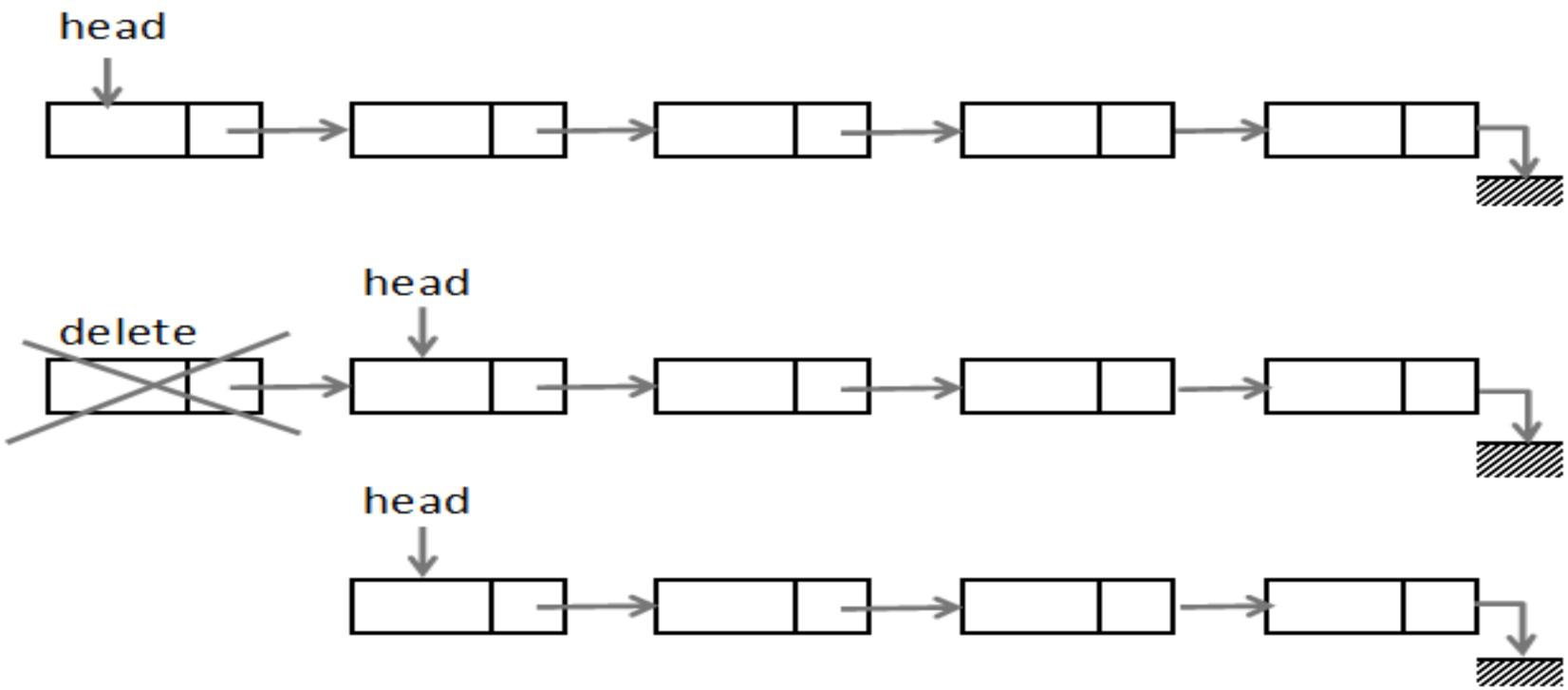
Delete Operation

- Pada bagian depan / delete head (WARNING!!!: don’t lose the head!)
- Pada bagian tengah
- Pada bagian akhir / delete tail

Delete Head

- Arahkan pointer ke Head node sebagai current node
- Set variabel Head ke next dari current node
- Hapus current Node

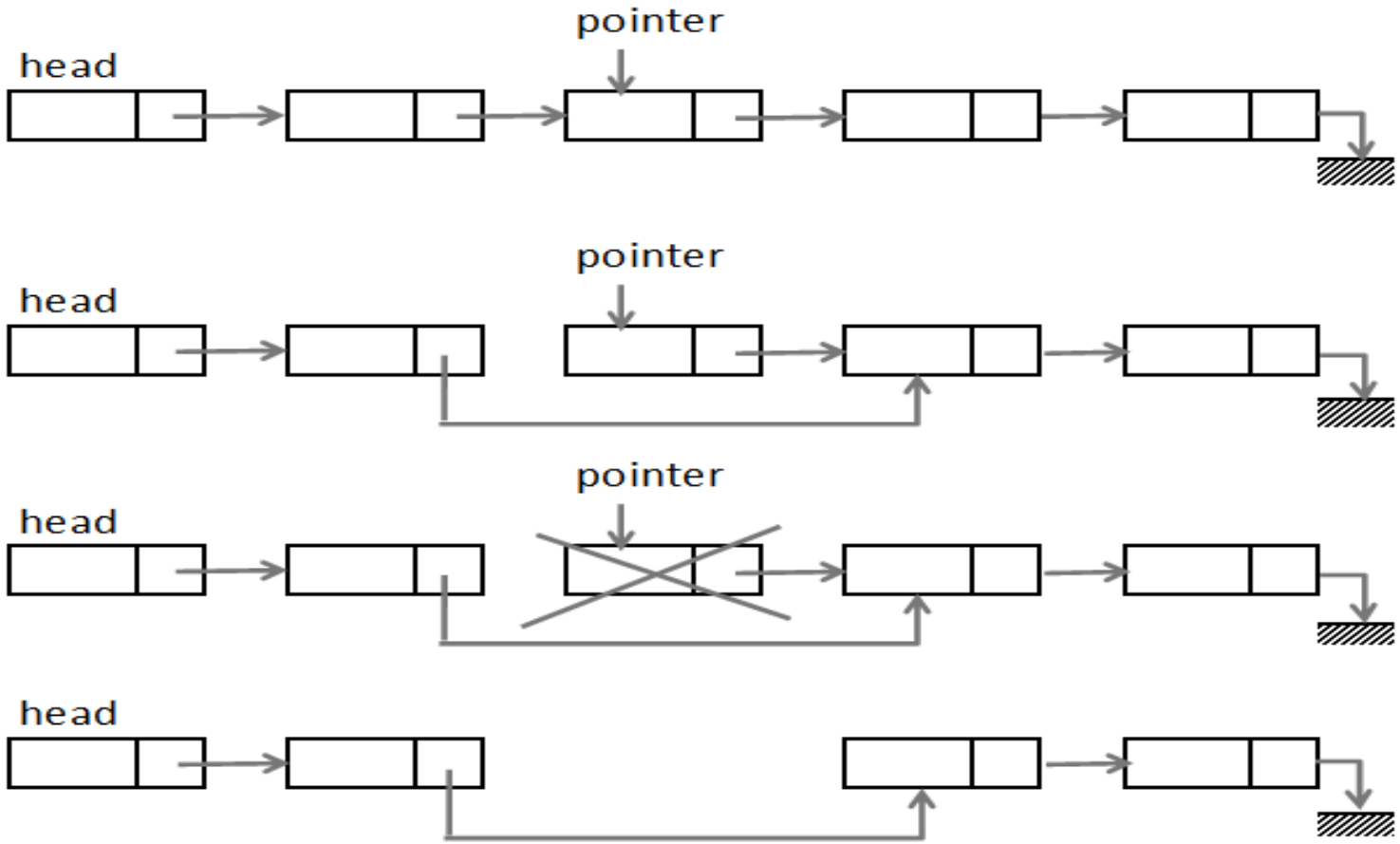
Delete Head



Delete Di tengah

- Lokasikan cursor ke node yang akan dihapus
- Set PreviousNode.next dengan CurrentNode.next
- Hapus Current Node

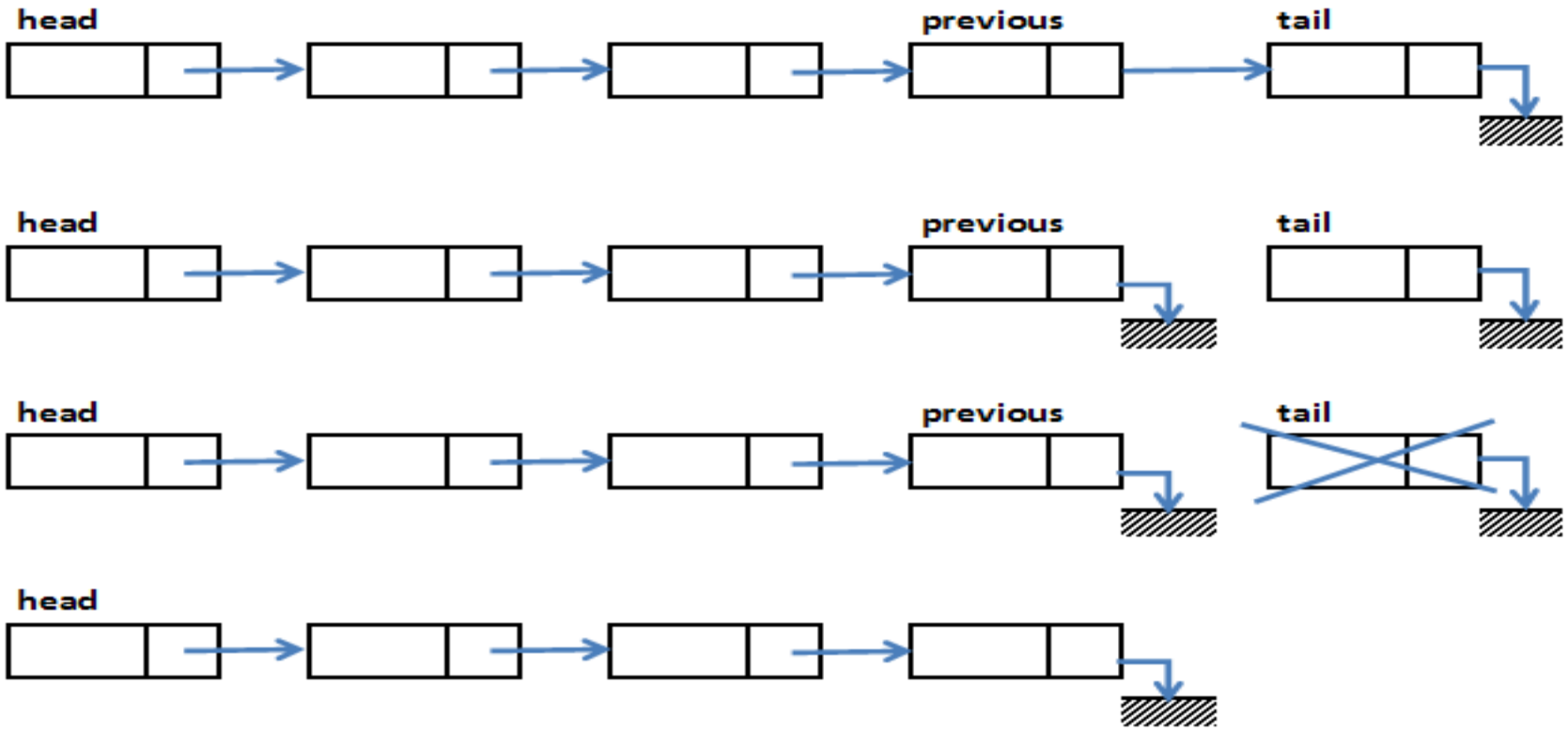
Delete Di tengah



Delete Pada Akhir List

- Lokasikan cursor ke tail / end of list.
- Set previous node.next sebagai Null.
- Hapus current node

Delete Pada Akhir List



5.

Memindahkan Node

Memindahkan Node

Kemungkinan pemindahkan Node:

- Node pada Head dipindahkan ke Tengah
- Node pada Head dipindahkan ke Tail
- Node di tengah dipindahkan ke Head
- Node di tengah dipindahkan ke Tail
- Node di tengah dipindahkan ke tengah yang lain
- Node pada Tail dipindahkan ke Head
- Node pada Tail dipindahkan ke Tengah

Memindahkan Node pada Head ke Tengah

- Memindahkan Node dapat dilakukan dengan bantuan dua buah variabel pointer yaitu ptrTujuan serta variabel pointer Ptr.
- simpan alamat node di depan tujuan pemindahan ke variabel pointer ptrTujuan (gunakan operasi pencarian node)
- Simpan alamat node yang akan dipindahkan ke variabel pointer Ptr,

  Ptr = Head
- Pindahkan Head ke node berikutnya

  Head = Head->next
- copykan next link dari node yang ditunjuk oleh ptrTujuan ke next link dari node yang ditunjuk oleh Ptr

  Ptr->next = ptrTujuan->next
- Arahkan next link dari node yang ditunjuk oleh ptrTujuan ke Ptr

  ptrTujuan->next=Ptr

Memindahkan Node pada Head ke Tail

- Memindahkan Node dapat dilakukan dengan bantuan dua buah variabel pointer yaitu ptrTujuan serta variabel pointer Ptr.
- simpan alamat node di depan tujuan pemindahan ke variabel pointer ptrTujuan yaitu node terakhir (gunakan operasi pencarian node atau jika mempunyai variabel pointer Tail dapat juga dengan mengcopykan Tail ke ptrTujuan : ptrTujuan = Tail)
- Simpan alamat node yang akan dipindahkan ke variabel pointer Ptr,
  Ptr = Head
- Pindahkan Head ke node berikutnya
  Head = Head->next
- copykan next link dari node yang ditunjuk oleh ptrTujuan ke next link dari node yang ditunjuk oleh Ptr
  Ptr->next = ptrTujuan->next
- Arahkan next link dari node yang ditunjuk oleh ptrTujuan ke Ptr
  ptrTujuan->next=Ptr
- Karena dipindahkan ke Tail, jangan lupa mengubah isi next link dari node yang dipindahkan menjadi NULL
  Ptr->next = NULL

Catatan: Prinsipnya sama seperti memindahkan node dari Head ke Tengah

Memindahkan Node ditengah ke Head

- Memindahkan Node dapat dilakukan dengan bantuan variabel pointer ptrAsal serta sebuah variabel pointer Ptr.
- Simpan alamat node yang akan dipindahkan ke variabel pointer Ptr, dan simpan alamat node didepan node yang akan dipindahkan ke variabel pointer ptrAsal. (gunakan operasi pencarian node untuk mengarahkan masing-masing variabel pointer tersebut)
- copykan next link dari node yang ditunjuk oleh Ptr ke next link dari node yang ditunjuk oleh ptrAsal

  ptrAsal->next = Ptr->next
- Isi next link dari node yang ditunjuk oleh Ptr ke dengan isi dari variabel Head

  Ptr->next = Head
- Arahkan Head ke node yang ditunjuk oleh Ptr

  Head = Ptr

Memindahkan Node ditengah ke Tengah

- Memindahkan Node dapat dilakukan dengan bantuan variabel pointer ptrAsal dan ptrTujuan serta sebuah variabel pointer Ptr.
- Simpan alamat node yang akan dipindahkan ke variabel pointer Ptr, dan simpan alamat node didepan node yang akan dipindahkan ke variabel pointer ptrAsal serta alamat node didepan node tujuan ke variabel pointer ptrTujuan. (gunakan operasi pencarian node untuk mengarahkan masing-masing variabel pointer tersebut)
- copykan next link dari node yang ditunjuk oleh Ptr ke next link dari node yang ditunjuk oleh ptrAsal

  ptrAsal->next = Ptr->next
- Isi next link dari node yang ditunjuk oleh Ptr ke dengan isi dari next link node yang ditunjuk oleh variabel ptrTujuan

  Ptr->next = ptrTujuan->next
- Isi next link dari node yang ditunjuk oleh ptrTujuan dengan isi dari next link node yang ditunjuk oleh variabel Ptr

  ptrTujuan->next = ptr

Memindahkan Node ditengah ke Tail

- Memindahkan Node dapat dilakukan dengan bantuan variabel pointer ptrAsal dan ptrTujuan serta sebuah variabel pointer Ptr.
- Simpan alamat node yang akan dipindahkan ke variabel pointer Ptr, dan simpan alamat node didepan node yang akan dipindahkan ke variabel pointer ptrAsal serta alamat node didepan node tujuan ke variabel pointer ptrTujuan. (gunakan operasi pencarian node untuk mengarahkan masing-masing variabel pointer tersebut)
- copykan next link dari node yang ditunjuk oleh Ptr ke next link dari node yang ditunjuk oleh ptrAsal

  ptrAsal->next = Ptr->next
- Isi next link dari node yang ditunjuk oleh Ptr ke dengan isi dari next link node yang ditunjuk oleh variabel ptrTujuan

  Ptr->next = ptrTujuan->next
- Isi next link dari node yang ditunjuk oleh ptrTujuan dengan alamat node yang ditunjuk oleh variabel Ptr

  ptrTujuan->next = Ptr
- Karena dipindahkan ke Tail, jangan lupa mengubah isi next link dari node yang dipindahkan menjadi NULL

  Ptr->next = NULL

Memindahkan Node pada Tail ke Head

- Memindahkan Node dapat dilakukan dengan bantuan dua buah variabel pointer yaitu ptrAsal serta variabel pointer Ptr.
- Simpan alamat node terakhir (Tail) ke Ptr
- Simpan alamat node di depan yang akan dipindahkan (di depan Tail) ke variabel pointer ptrAsal,
- Arahkan next link dari node yang ditunjuk oleh Ptr ke Head

  Ptr->next = Head
- Pindahkan Head ke node yang ditunjuk oleh Ptr

  Head = Ptr
- NULL kan next link dari node yang ditunjuk oleh ptrAsal

  ptrAsal>next=NULL
- Jangan lupa jika mempunyai variabel Tail, copykan ptrTujuan ke Tail (Tail=ptrAsal)

Memindahkan Node pada Tail ke tengah

- Sama seperti memindahkan Node dari Tengah ke Tengah yang lain
- Karena yang dipindahkan ada pada Tail, Jangan lupa mengubah ptrAsal->next yang semula menunjuk ke node yang akan dipindahkan dengan NULL

Ringkasan

- Single Linked List sangat tergantung pada Head, Head jangan sampai hilang
- Penambahan, penghapusan, pemindahan node tidak boleh sampai membuat sebagian link hilang karena link terputus
- Penggunaan variabel pointer Tail dapat mempermudah proses-prosespada single linked list
- Detail proses penghapusan, penambahan, pencarian dan pemindahan node dapat dilihat pada masing-masing slide, proses tersebut hanya salah satu contoh proses saja, banyak variasi proses yagn lain

Terimakasih

TUHAN Memberkati Anda

Teady Matius Surya Mulyana (tmulyana@bundamulia.ac.id)