Single Linked-List

(TIB11 – Struktur Data)

Pertemuan 9, 10

Sub-CPMK

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

Materi:

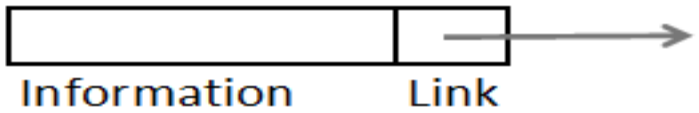
1. Konsep *Linked-List*
2. Menambahkan Node
3. Mencari Node
4. Menghapus Node
5. Memindahkan Node

1. Konsep *Linked-List*

1.1. *Linked List*

- *Linked-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



1.2. 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

1.2. Dua *Variable* Penting *Linked-list* (Lanj.)

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'!!!***

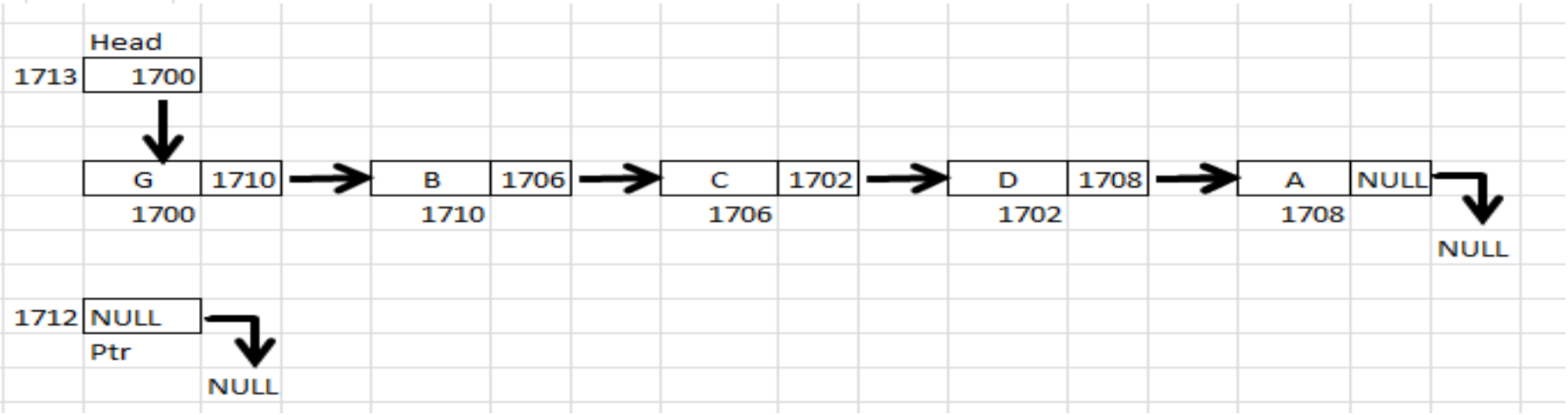
1.3. Beberapa *Variants Linked List* yang Umum

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

1.3. Beberapa *Variants Linked List* yang Umum (Lanj.)

- Tiap simpul memilik alamat memory, link next yang tersimpan berisi alamat memory dari simpul berikutnya

	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		

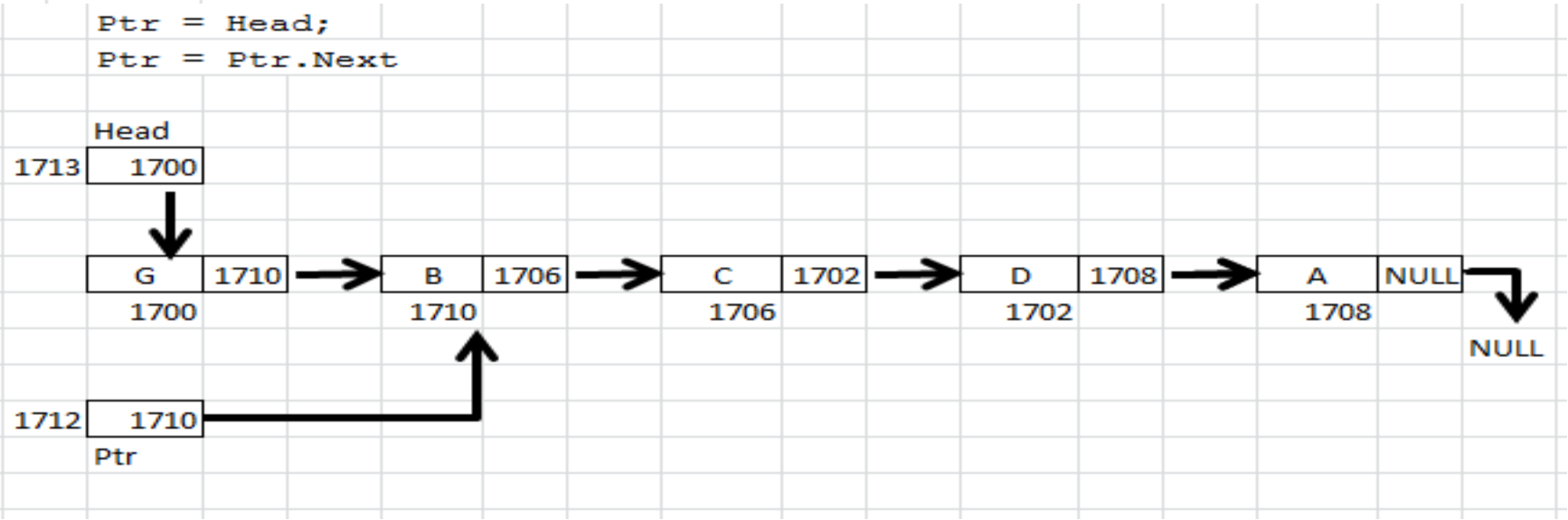


1.3. Beberapa *Variants Linked List* yang Umum (Lanj.)

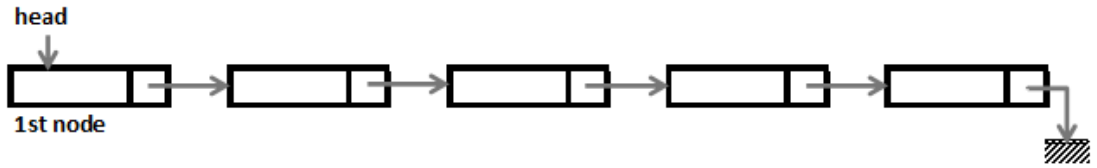
- Untuk melakukan penulusuran linked-list, pointer ptr dapat diarahkan ke ke Head dahulu
- Kemudian untuk berpindah simpul, dapat dilakukan dengan mengisi ptr dengan isi dari next link simpul tsb, sehingga ptr dapat menunjuk ke simpul berikutnya (ptr=ptr->Next

	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
```



1.4. *Single Linked List*



- Contoh Single Linked List,
- tiap simpul memilik alamat memory, link next yang tersimpan berisi alamat memory dari simpul berikutnya
- Head selalu menunjuk ke simpul pertama

2. Menambahkan Node

2.1. *Linked List Operation*

Berikut ini operasi-operasi linked List

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

Catatan: Setiap operasi tersebut akan dijelaskan pada slide-slide berikutnya

2.2. Kemungkinan Operasi

Kemungkinan operasi penyisipan, penghapusan maupun pemindahan dapat terjadi pada:

- bagian depan dari *list*
- tengah *list*
- bagian akhir dari *list*

2.3. *Insert Operation*

- Pada bagian depan *list*

  Hanya dapat terjadi pada operasi *insert* sebelum *current cell*
- Pada bagian akhir *list*

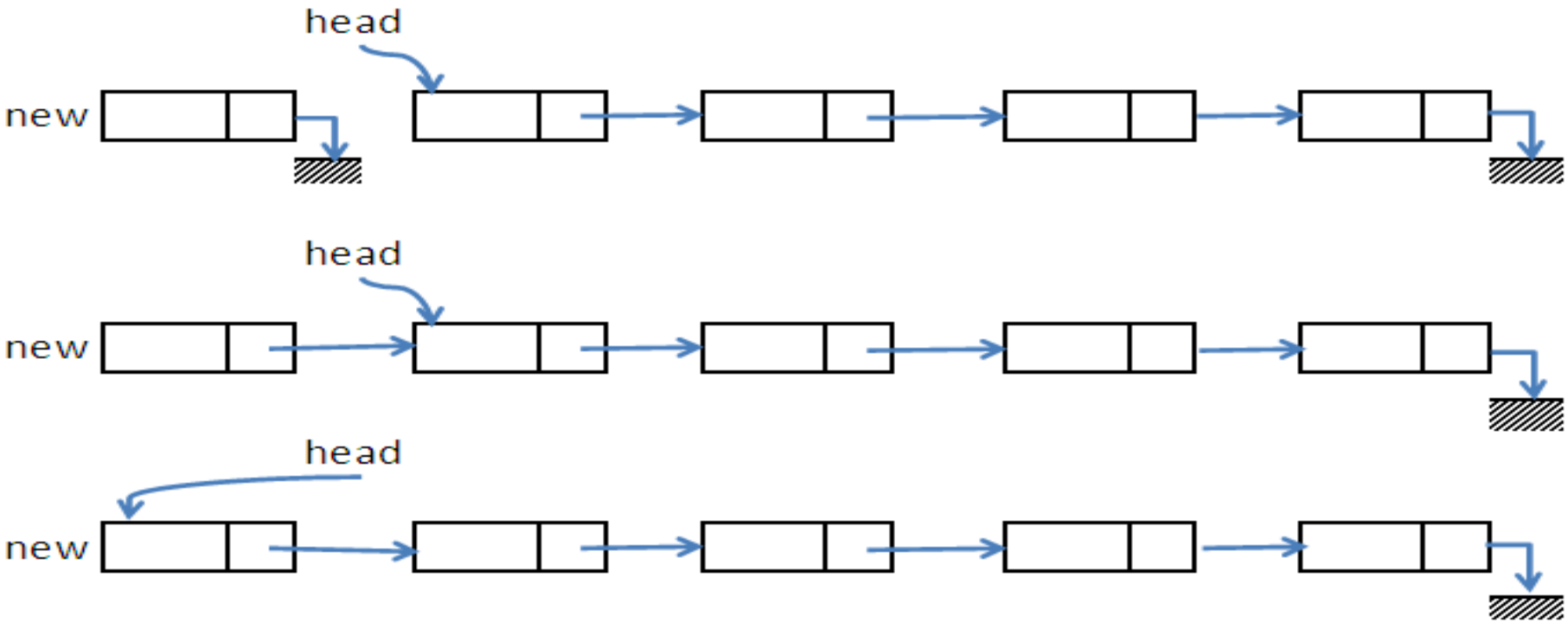
  Hanya dapat terjadi pada *insert* setelah *current cell*
- Pada bagian tengah *middle list*

Catatan: Gambar ilustrasi masing-masing operasi dapat dilihat pada penjelasan slide berikutnya

2.4. *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

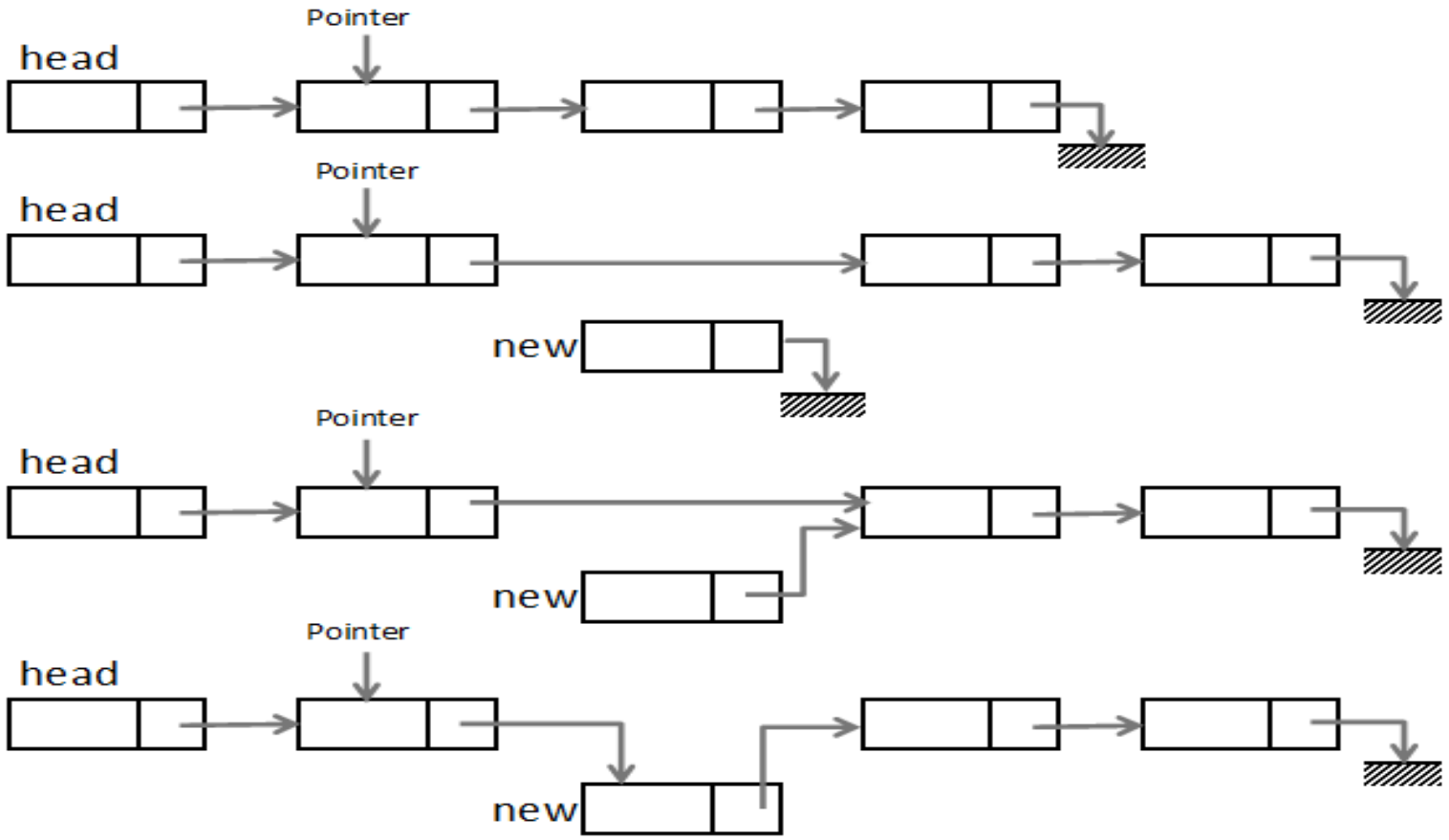
2.4. *Insert* Pada Bagian Depan *List* (Lanj.)



2.5. 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

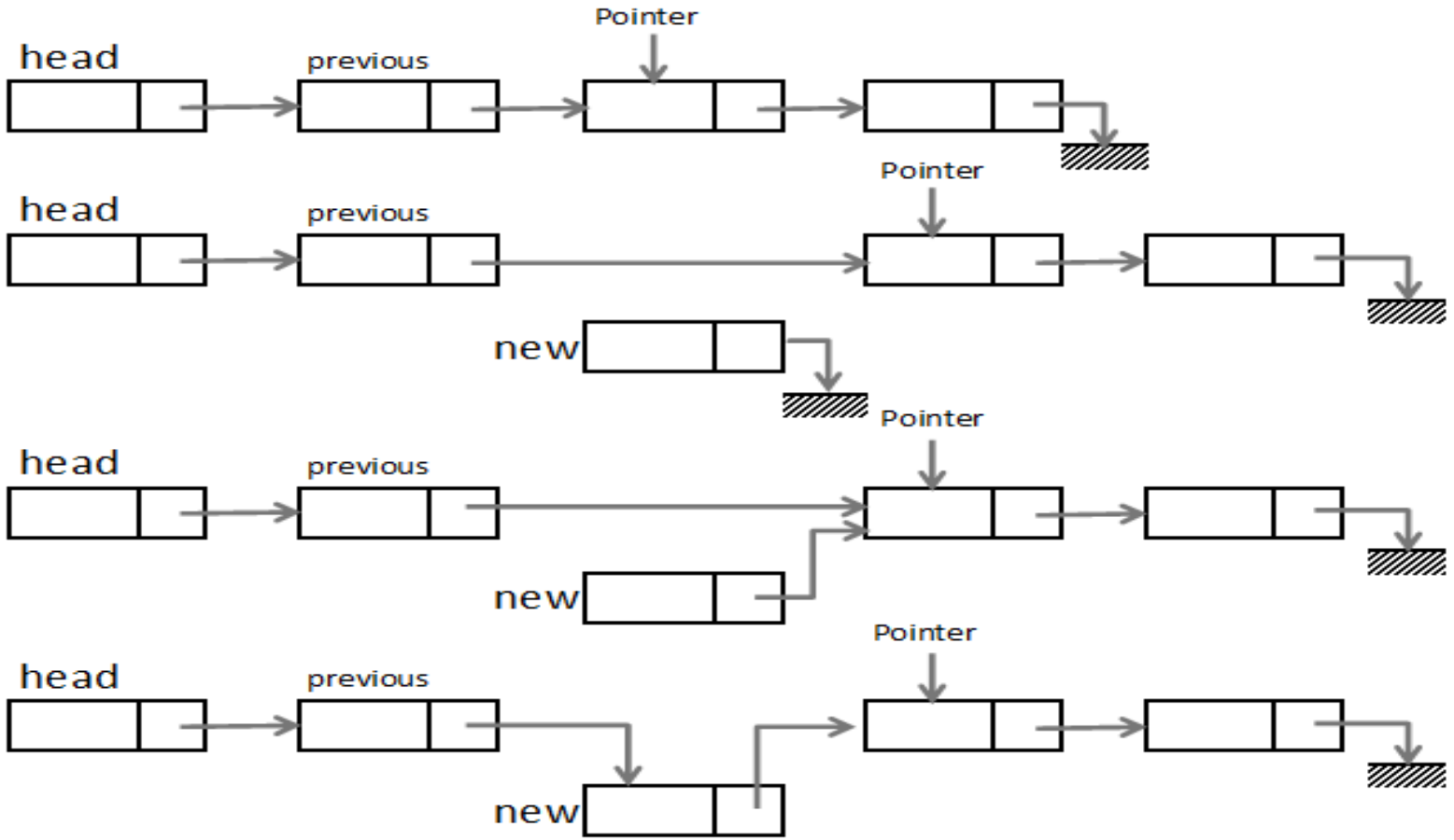
2.5. Insert di Tengah – Setelah *Current Cell* (Lanj.)



2.6. *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

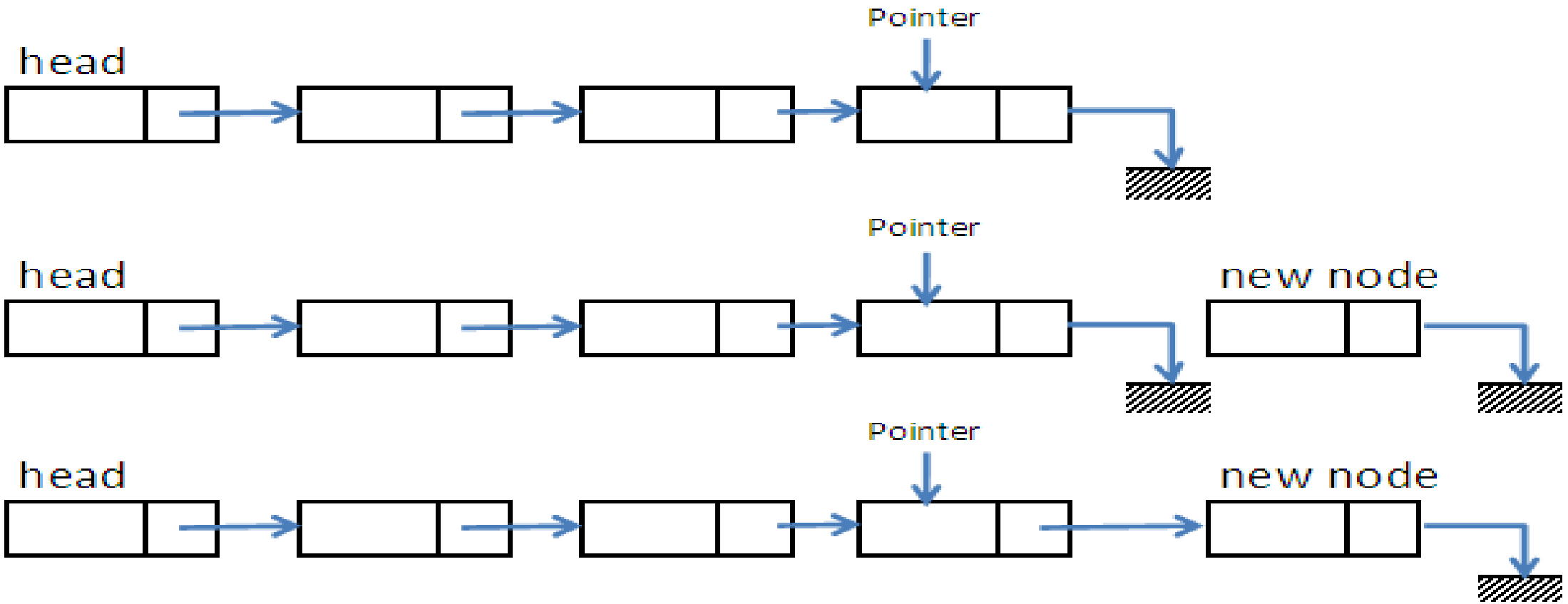
2.6. *Insert* Di Tengah – Sebelum *Current Cell* (Lanj.)



2.7. *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

2.7. *Insert* Pada Bagian Akhir *List* (Lanj.)



3. Mencari Node

3.1. *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;
```

3.2. *Locating* Ke *Previous* Node

Dapat dilakukan dengan berbagai cara

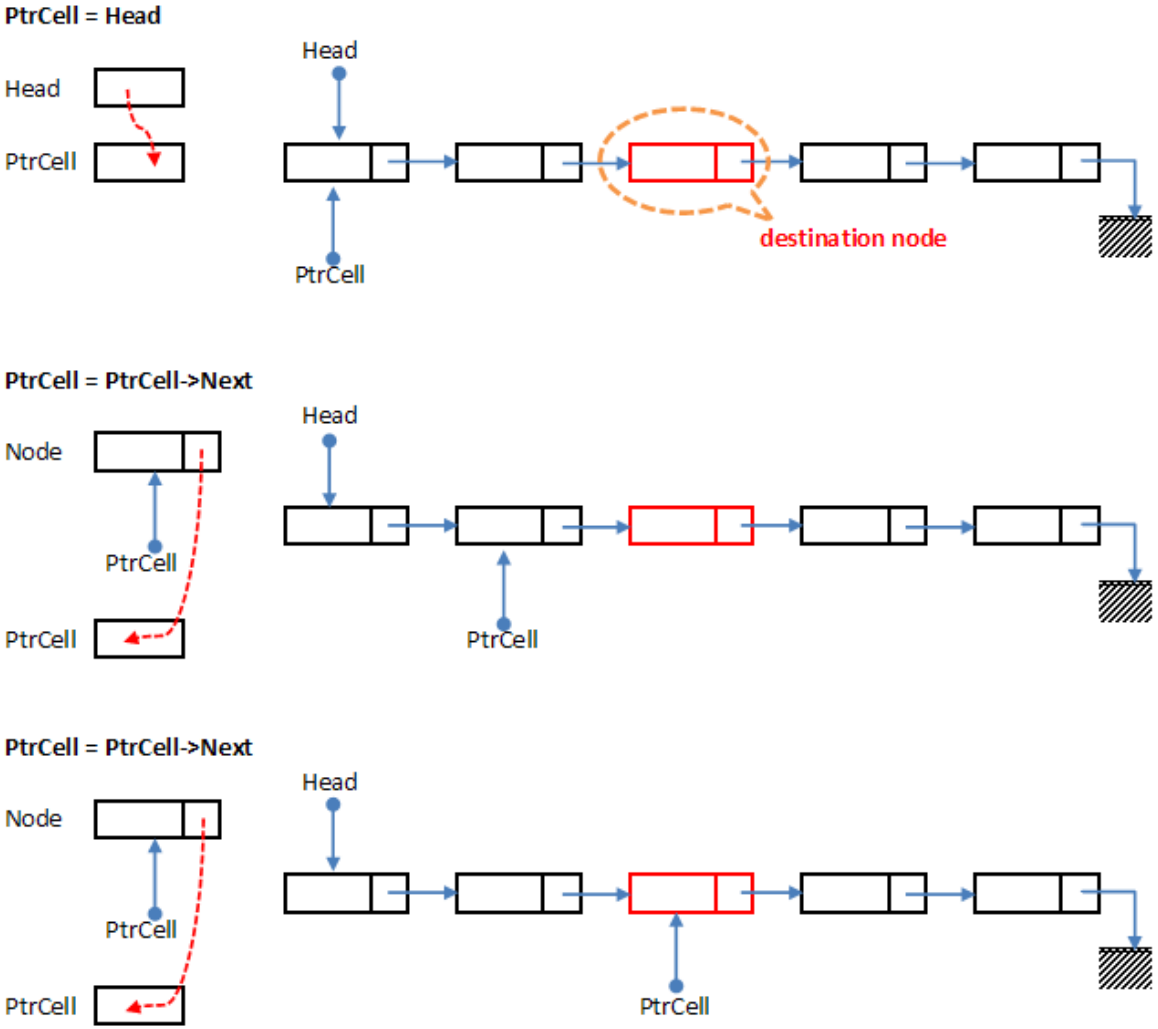
- Simpan *previous* node ketika *locating the current* node

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

- *Retrieve* jika diperlukan

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

3.2. *Locating* Ke *Previous* Node (Lanj.)



- Arahkan Ptr ke Head (isi variabel Ptr dengan Head)

  PtrCell = Head

- Copykan isi next dari simpul yang di tunjuk oleh variabel Ptr ke variabel Ptr

  PtrCell = PtrCell->Next

- Ulangi sampai menemukan simpul yang dituju

4. Menghapus Node

4.1. *Delete Operation*

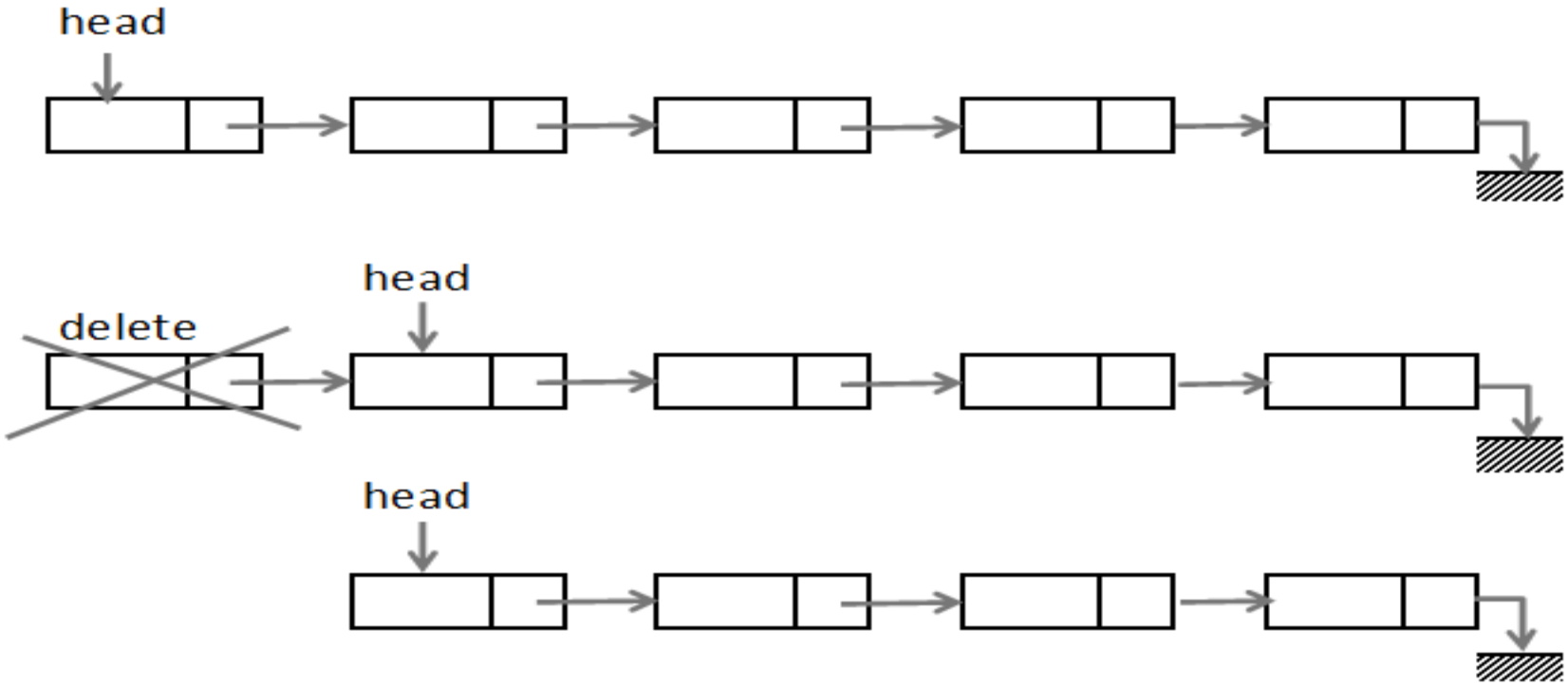
- Pada bagian depan / *delete head* (*WARNING!!!: don't lose the head!*)
- Pada bagian tengah
- Pada bagian akhir / *delete tail*

4.2. *Delete Head*

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

Catatan: Gambar ada di slide berikutnya

4.2. *Delete Head* (Lanj.)

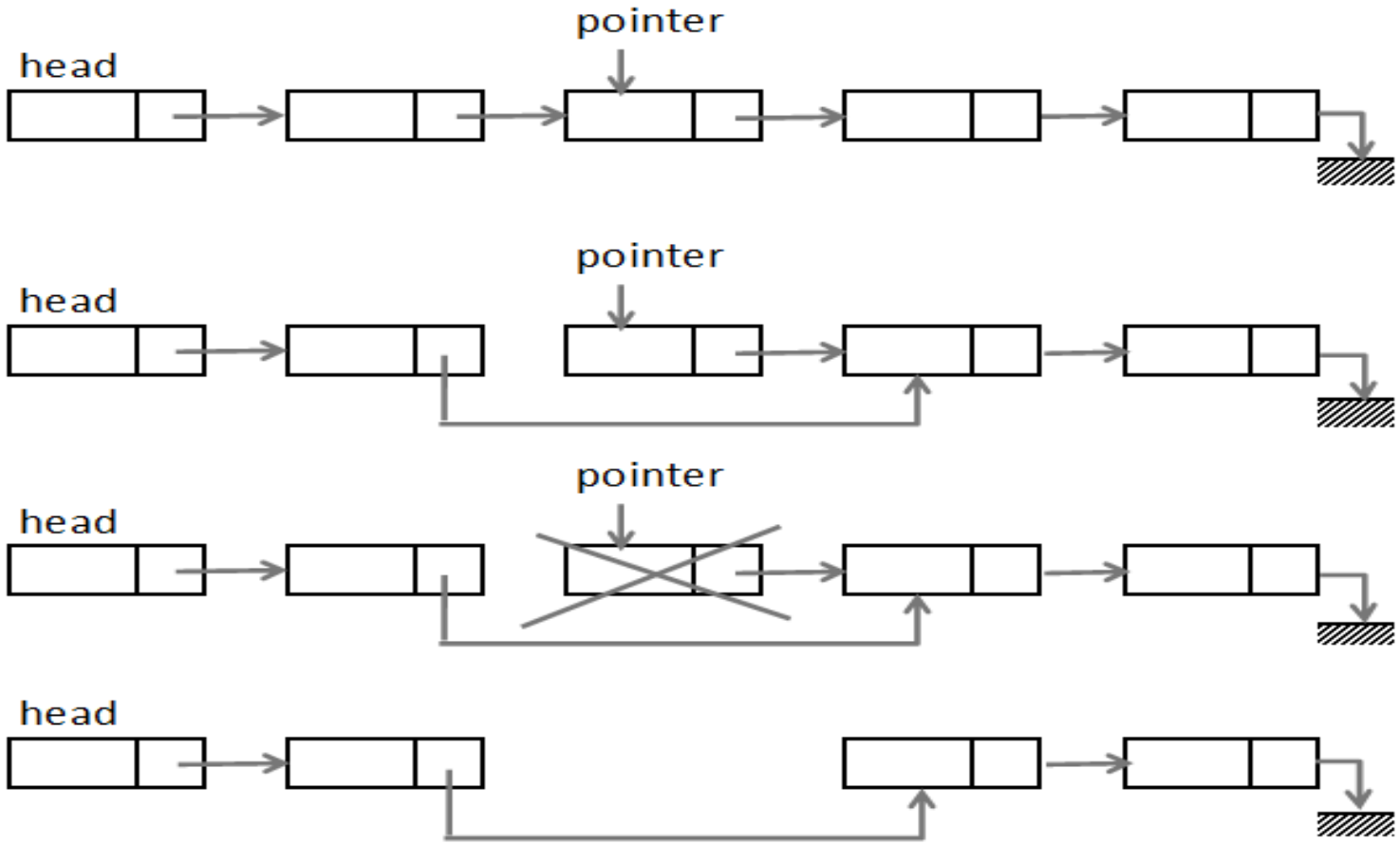


4.3. *Delete* di Tengah

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

Catatan: Gambar ada di slide berikutnya

4.3. *Delete* di Tengah (Lanj.)

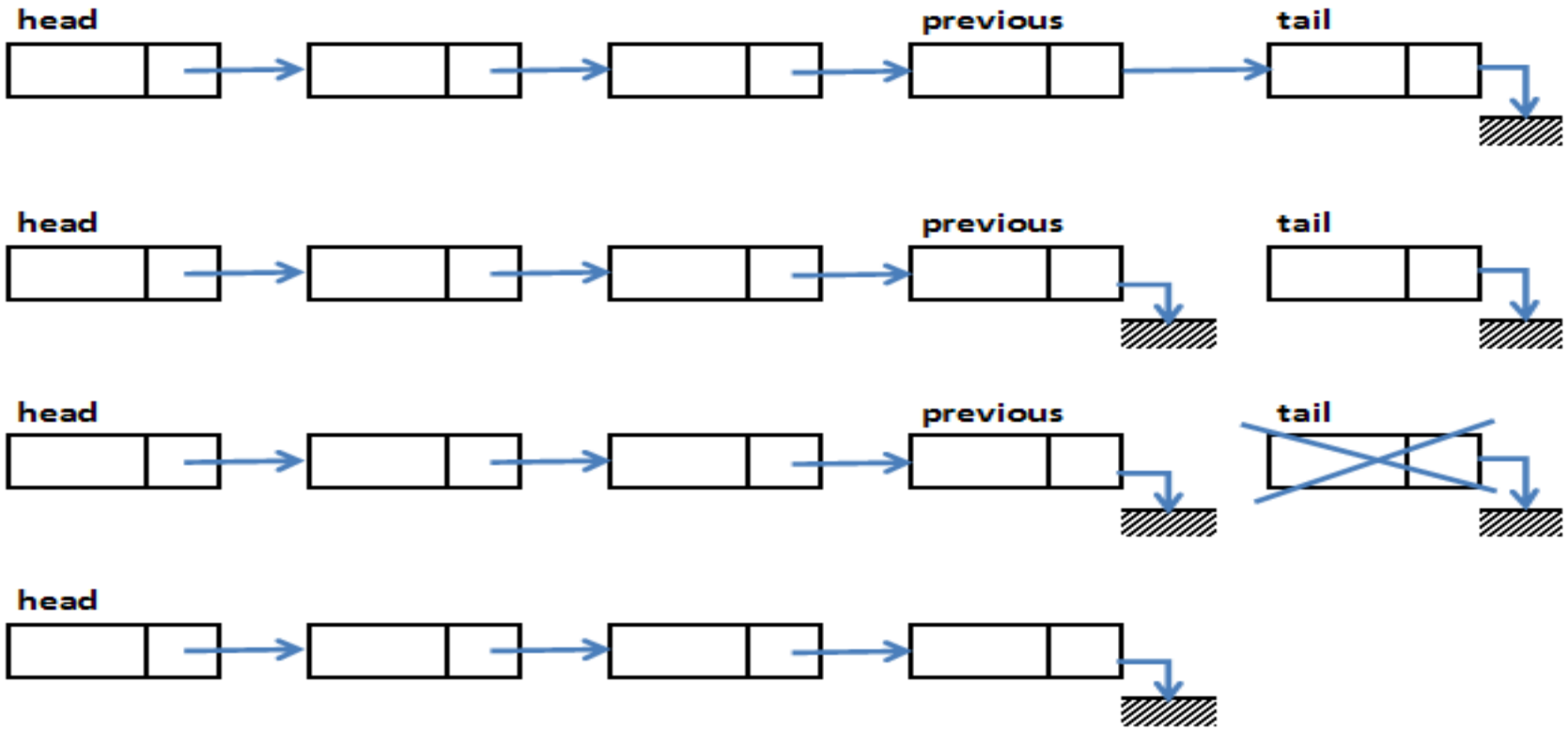


4.4. *Delete* Pada Akhir *List*

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

Catatan: Gambar ada di slide berikutnya

4.4. *Delete* Pada Akhir *List (Lanj.)*



5. 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

5.1. 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

5.2. 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

5.3. 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

5.3. Memindahkan Node ditengah ke *Head* (Lanj.)

- 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

5.4. 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

5.5. 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)

5.6. 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-proses pada *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 yang lain

Contoh membuat single Linked-List

```
1  #include <stdio.h>
2  #include <conio.h>
3  #include <iostream.h>
4  struct Simpul
5  {
6    char dat[10];
7    struct Simpul *next;
8  };
9  struct Simpul *temp, *head=NULL, *ptr=NULL;
10
11 /* program utama */
12 void main()
13 {
14   char masukan[10];
15   do
16   {
17     cout<<"Input Data: ";gets(masukan);
18     if(strlen(masukan)>0)
19     {
20       if(strlen(masukan)>0)
21       {
```

Contoh membuat single Linked-List (lanjutan)

```
22              for(int i=strlen(masukan);i<10;i++)
23                strcat(masukan," ");
24              //Tentukan Alokasi Memori Baru
25              temp = (struct Simpul *) malloc(sizeof(struct Simpul));
26              //Isi Data
27              strcpy(temp->dat,masukan);
28              temp->next = NULL;
29              if(head==NULL)
30              {
31                //Tentukan Kepala
32                head=temp;
33                ptr=head;
34              } else
35              {
36                //Sambungkan Link Ptr Terakhir ke Link Baru
37                ptr->next = temp;
38                ptr=ptr->next;
39              }
40            }
41          }
42        }while (strlen(masukan)>0);
```

Contoh membuat single Linked-List (lanjutan)

```
43      cout<<endl;
44      //Tampilkan Semua Node
45      if (head != NULL)
46      {
47        ptr = head;
48        cout<<"cell: {Alamat} [ Data | Cell berikut]"<<endl;
49        do
50        {
51          cout<<"cell: {"<<ptr<<"} ["<<ptr->dat<<" | "<<ptr->next<<"]"<<endl;
52          ptr = ptr->next;
53        } while(ptr != NULL);
54      }
55      getch();
56  }
```

Terimakasih

TUHAN Memberkati Anda

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