Double Linked-List

(TIB11 – Struktur Data)

Pertemuan 11, 12

Sub-CPMK

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

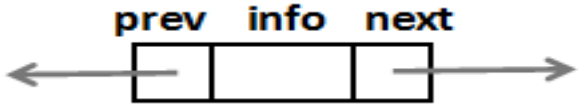
Materi:

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

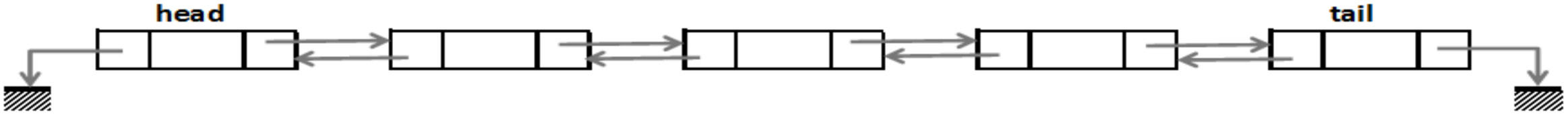
1. Konsep *Double Linked-List*

1.1. *Double Linked List*

- Setiap node memiliki dua *link*



- *Previous Link* menunjuk ke *previous* node
- *Next Link* menunjuk ke *next* node
- *Head* → Prev *Link Pointed as* NULL
- *Tail* → *Next Link Pointed as* NULL



1.2. Kelebihan *Double Linked List*

- Ketika kehilagan *Head*, asalkan *Pointer* masih menunjuk ke salah satu node/simpul, maka *Head* masih dapat dicari dengan melakukan *Previous* terus menerus sampai ditemukan simpul yang link *previous*nya menunjuk ke NULL

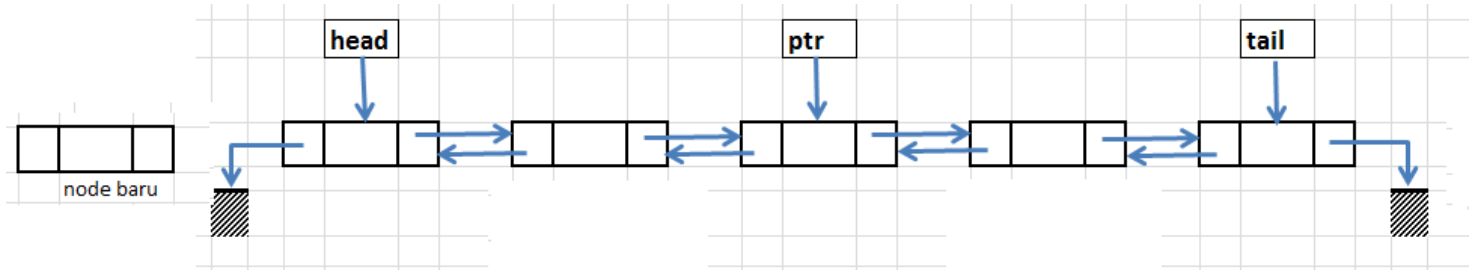
1.3. Operasi pada *Double Linked List*

- Sama seperti pada *Single Linked List*, *Double linked list* juga memiliki operasi-operasi cari, sisip/tambah dan hapus dengan mengarahkan *previous Link* ke *previous* node dari node yang akan dihapus
- Dengan adanya prev link pada tiap nodenya dapat mengurangi penambahan variabel *pointer* pada tiap operasinya

2. Menambahkan Node

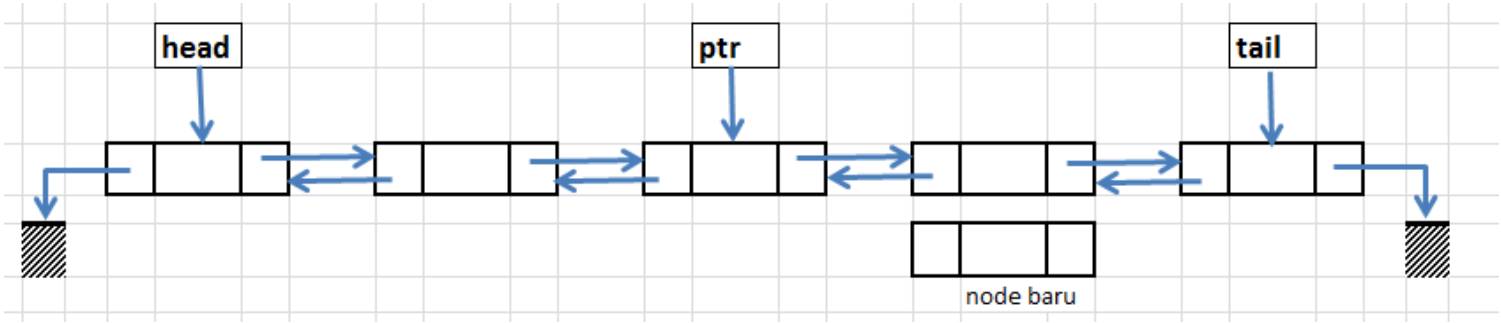
2.1. *Insert* Pada Bagian Depan *List*

- Buat sebuah simpul/node baru
  ptrBaru = malloc(……..)
- Isi informasi simpul/node baru tersebut
- Arahkan *next link* ke simpul/node kepala
  ptrBaru->*next* = *Head*
- Arahkan prev *link* dari simpul kepala ke node baru
  Head->prev = ptrBaru
- Isi prev *link* node baru dengan NULL
  ptrBaru->prev = NULL
- *Set head pointer* ke simpul/node baru tersebut
  Head = ptrBaru



2.2. Insert di Tengah – Setelah *Current Cell*

- Arahkan Ptr ke node yang akan dibuat node baru pada *next link* nya
- Buat sebuah simpul/node baru
  Baru = malloc()
- Isi informasi simpul/node baru tersebut
- *Copy next link* dari *current* node/simpul *ke next link* simpul/node baru
  Baru->*next* = Ptr->*next*
- *Copy* prev *link* dari node setelah current node/simpul ke prev *link* simpul/node baru
  Baru->prev = Ptr->*next*->prev
- *Set* prev *link* pada simpul/node setelah current simpul/node kenode baru
  Ptr->*next*->prev = Baru
- *Set next link* pada current simpul/node ke simpul/node baru
  Ptr->*next* = Baru



2.3. Insert Di Tengah – Sebelum *Current Cell*

- Arahkan Ptr ke node yang akan dibuat node baru pada prev *link* nya
- Buat sebuah simpul/node baru

  Baru = malloc()
- Isi informasi simpul/node baru tersebut
- *Copy* prev *link* dari *current* node/simpul ke prev *link* simpul/*node* baru

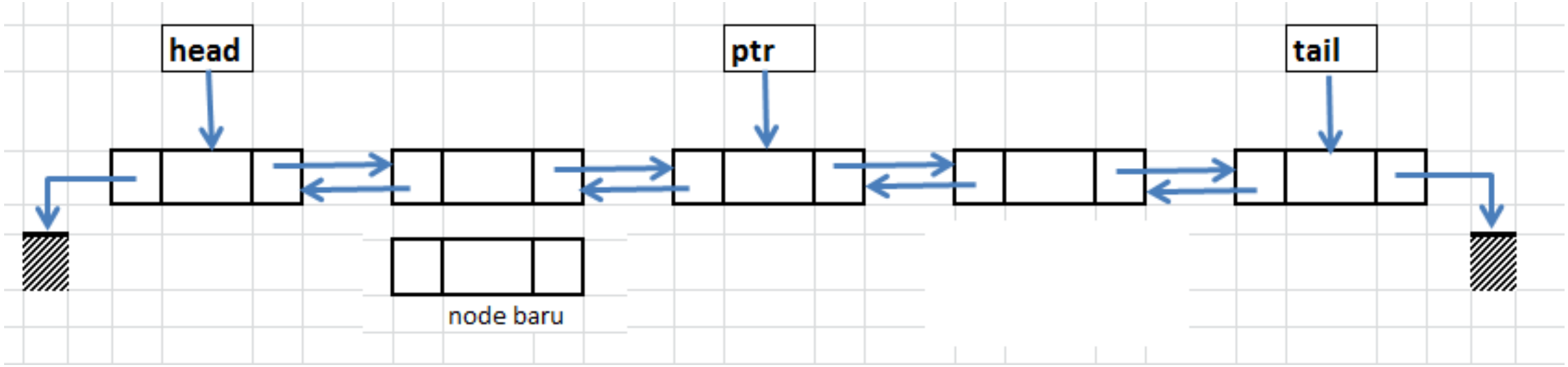
  Baru->prev = Ptr->prev
- *Copy next link* dari node sebelum *current* node/simpul ke *next link* simpul/node baru

  Baru->*next* = Ptr->prev->*next*
- *Set next link* pada simpul/node sebelum *current* simpul/node kenode baru

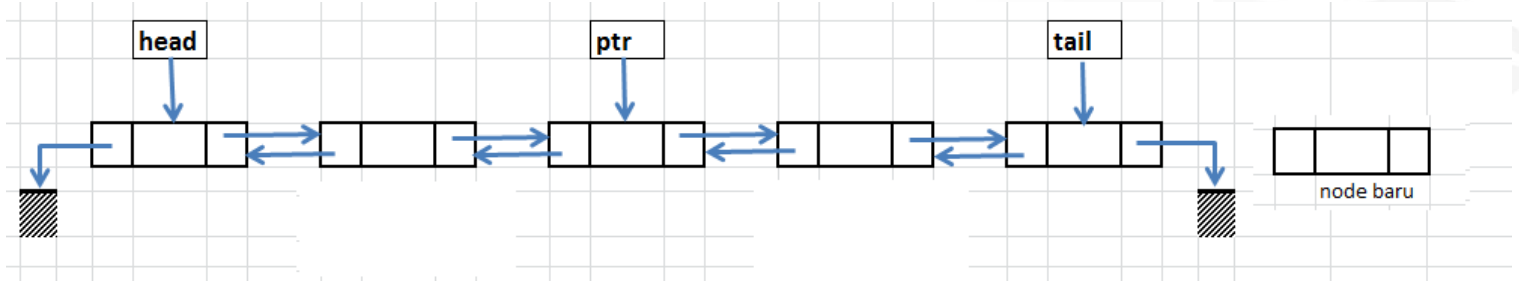
  Ptr->prev->*next* = Baru
- *Set* prev *link* pada *current* simpul/node ke simpul/node baru

  Ptr->prev = Baru

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



2.4. *Insert* Pada Bagian Akhir *List*



- Arahkan Ptr ke node terakhir
- Buat sebuah node baru

  Baru = malloc()
- Isi informasi node baru tersebut
- *Set next link* dari baru node sebagai NULL

  Baru->*next* = NULL
- Isi prev *link* dari node baru ke *Tail*

  Baru->prev = *TAIL*
- Arahkan *next link* pada node terakhir atau *tail* ke node baru

  Ptr->*next* = Baru
- Jika memiliki variabel *Tail*, Jangan lupa memindahkan *Tail* ke node baru

  Tail = Baru

3. Mencari Node

3.1. Mencari Node Pada *Double* LL

- Mencari node pada *Double* LL dapat dilakukan dengan cara maju menggunakan *next link* atau mundur menggunakan prev *link*
- Jika maju maka Ptr diarahkan ke *Head*
- Jika mundur maka Ptr diarahkan ke *Tail*

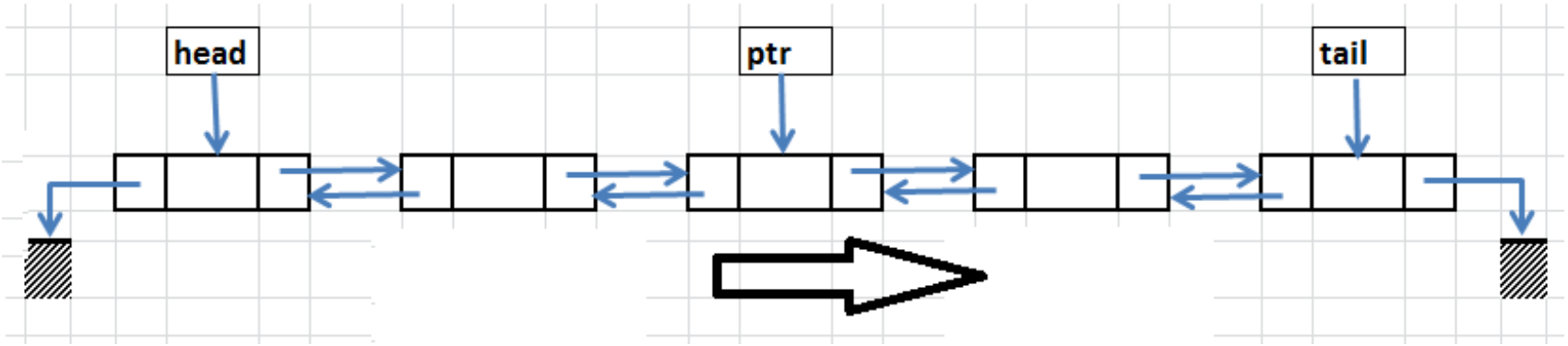
3.2. Pencarian Maju

- Proses sama seperti pada *single* LL
- Assign *PointerCell* sebagai *Head*

```
PointerCell = Head;
```

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

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



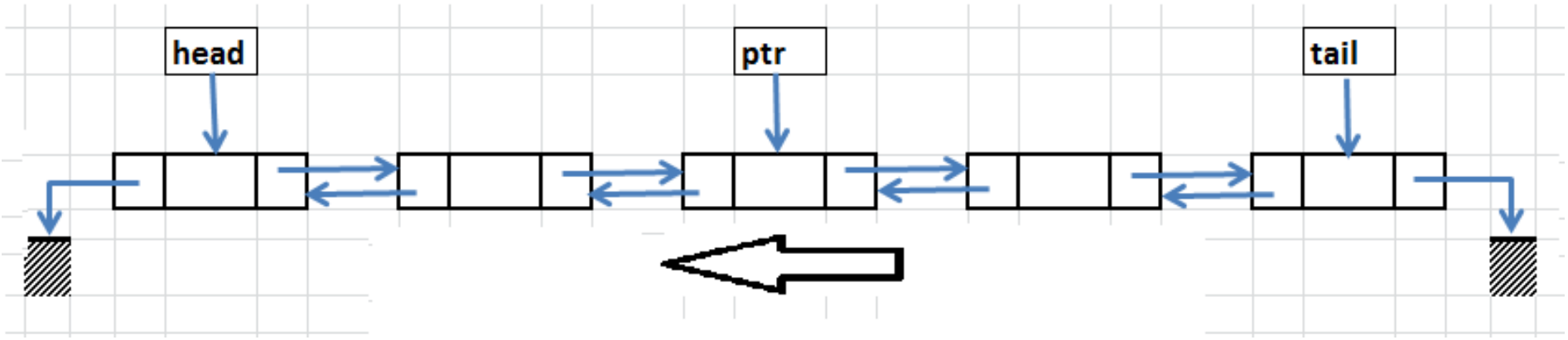
3.3. Pencarian Mundur

- *Assign PointerCell* sebagai *Head*

```
PointerCell = Tail;
```

- Bergerak mundur dengan mengarahkan *PointerCell* ke *prev PointerCell* sampai ditemukan node/simpul yang sesuai.

```
PointerCell = PointerCell->prev;
```



4. Menghapus Node

4.1. *Delete Operation*

Penghapusan dapat dilakukan:

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

- Karena ini adalah *double linked-list*, maka jangan lupa mengarahkan prev *link* dari node setelah node yang akan dihapus ke node pada prev *link* dari node yang akan dihapus

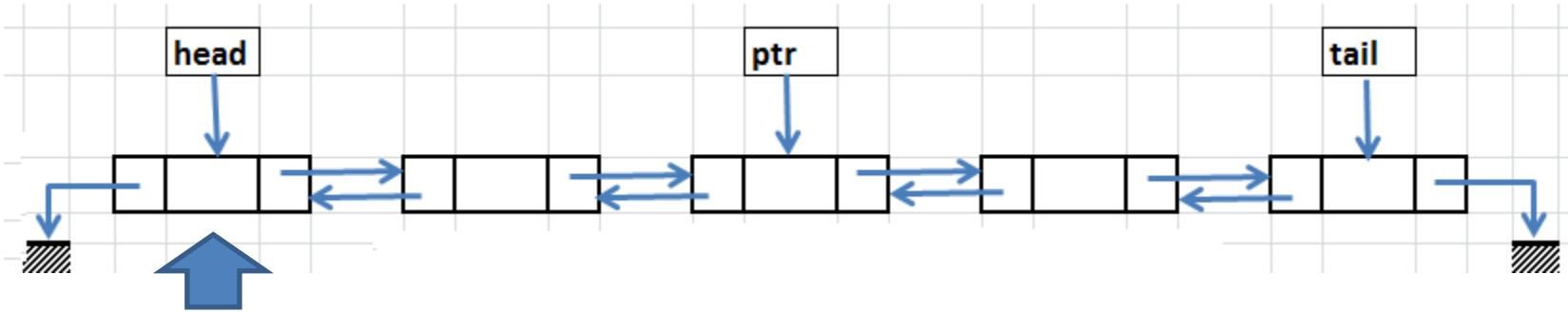
4.2. Hapus Pada Bagian Depan / *Delete Head*

- (*WARNING!!!: don't lose the head!*)
- Arahkan *pointer* Ptr ke *Head* node sebagai *current* node
- Set variabel *Head* ke *next* dari *current* node

  Head = Head->Next
- *Set* prev *link* dari *Head* dengan NULL

  Head->prev = NULL
- Hapus *current* Node

  Free(Ptr)



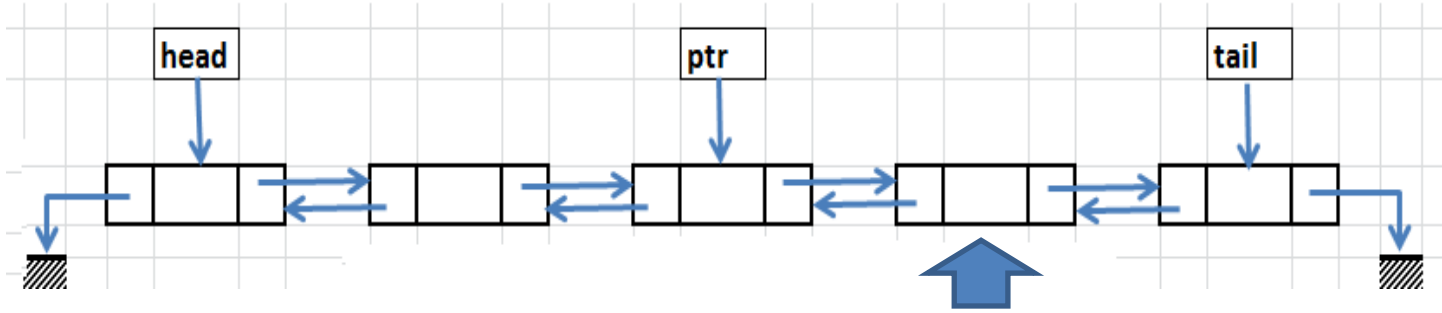
4.3. Hapus Pada Bagian Tengah

- Arahkan *Pointer* Ptr ke node yang akan dihapus
- *Copy link* prev node dari node yangakan dihapus ke prev link node setelah node yang akan dihapus

  Ptr->*next*->prev = Ptr->Prev
- *Copy link next* node dari node yang akan dihapus ke *next link* node sebelum node yang akan dihapus

  Ptr->prev->*next* = Ptr->prev
- Hapus *Current* Node

  Free(Ptr)

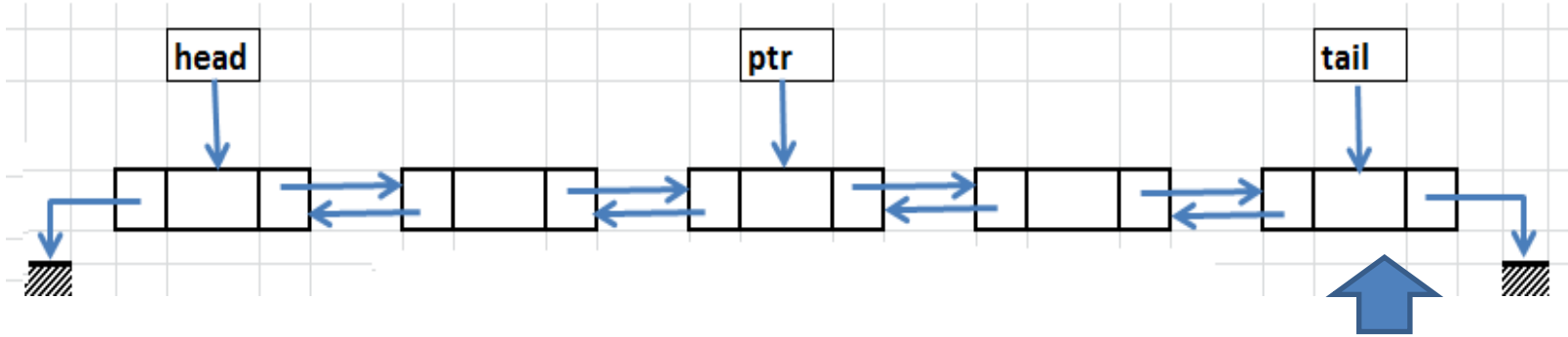


4.4. Hapus Pada Bagian Akhir / *Delete Tail*

- Arahkan *Pointer* Ptr ke node yang akan dihapus Atau dalam hal ini adalah node terakhir
- Ubahlah *next link* dari node sebelum node yang akan dihapus dengan NULL

  Ptr->prev->*next* = NULL
- Hapus *current* node

  Free(Ptr)



5. Memindahkan Node

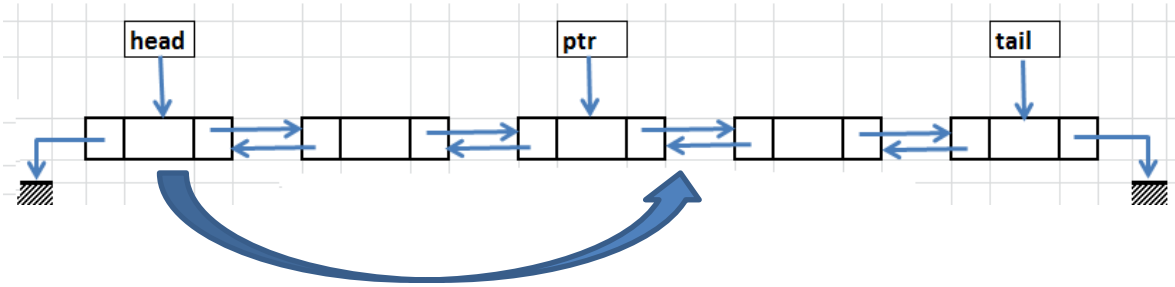
5.1. Memindahkan Node pada *Head* ke Tengah

- 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*
- Isi prev *Head* yagn sekarang dengan NULL

  Head->prev = NULL



5.1. Memindahkan Node pada *Head* ke Tengah (Lanj.)

- copykan *next link* dari node yang ditunjuk oleh ptrTujuan ke *next link* dari node yang ditunjuk oleh Ptr

  Ptr->*next* = ptrTujuan->*next*
- copykan prev *link* dari node setelah Node Tujuan ke node yang ditunjuk oleh Ptr

  Ptr->prev = ptrTujuan->*next*->prev
- Arahkan prev *link* dari node yang ditunjuk oleh node setelah node Tujuan ke Ptr

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

  ptrTujuan->*next*=Ptr

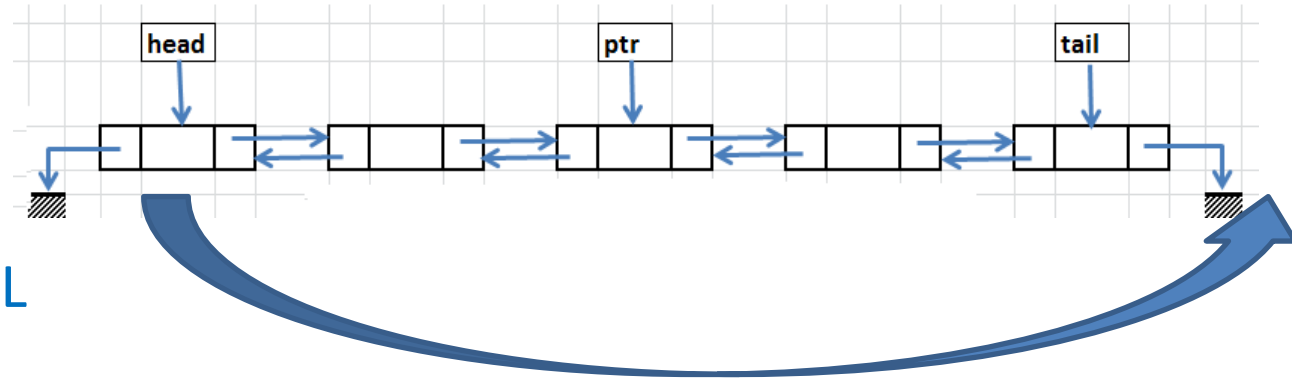
5.2. Memindahkan Node pada *Head* ke *Tail*

- 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
- Isi prev *Head* yang sekarang dengan NULL

  Head ->prev = NULL



5.2. Memindahkan Node pada *Head* ke *Tail* (Lanj.)

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

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

  Ptr->prev = ptrTujuan
- 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 sebuah variabel *pointer* Ptr.
- Simpan alamat node yang akan dipindahkan ke variabel *pointer* Ptr, (gunakan operasi pencarian node)
- copykan *next link* dari node yang ditunjuk oleh Ptr ke *next link* dari node sebelum dari node tersebut

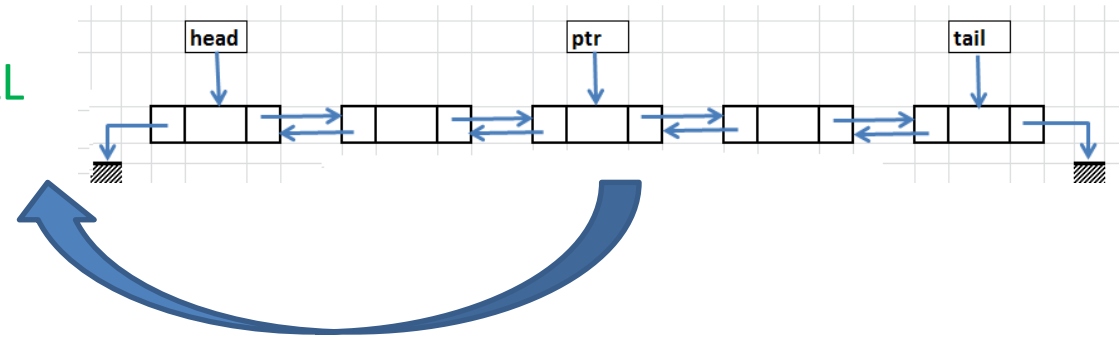
  Ptr->prev->*next* = Ptr->*next*
- copykan prev *link* dari node yang ditunjuk oleh Ptr ke prev *link* dari node berikut dari node tersebut

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

  Ptr->*next* = *Head*
- Isi prev *link* dari node yang ditunjuk oleh Ptr dengan NULL

  Ptr->prev = NULL
- Arahkan *Head* ke node yang ditunjuk oleh Ptr

  Head = Ptr



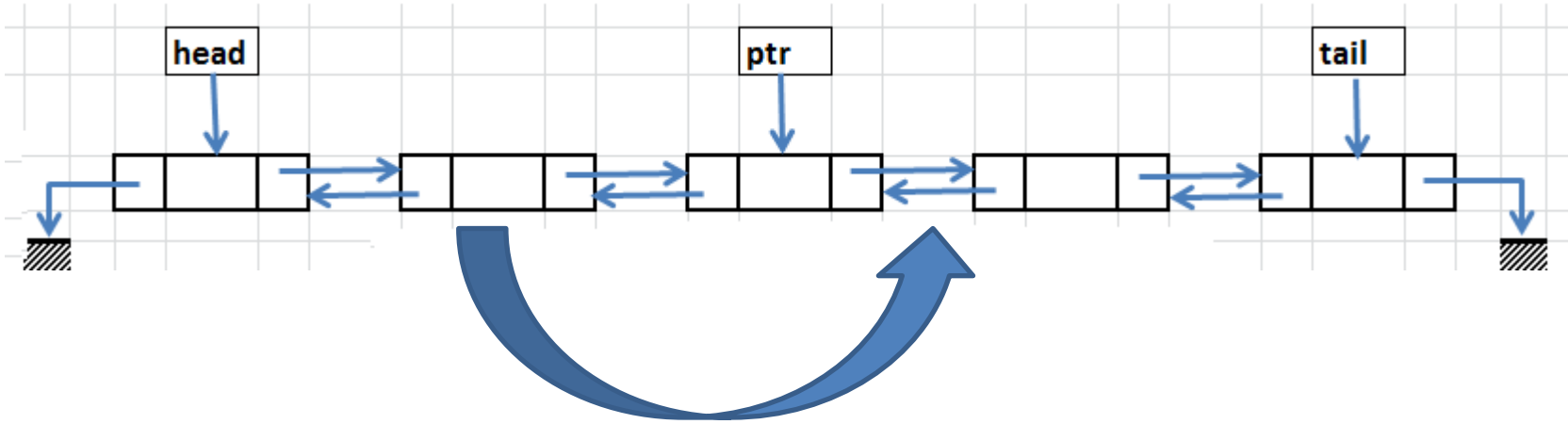
5.4. Memindahkan Node ditengah ke Tengah (setelah *current*)

- Memindahkan Node dapat dilakukan dengan bantuan variabel *pointer* ptrTujuan serta sebuah variabel *pointer* Ptr.
- Simpan alamat node yang akan dipindahkan ke variabel *pointer* Ptr serta simpan 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 sebelum dari node tersebut

  Ptr->prev->*next* = Ptr->*next*
- Copykan prev *link* dari node yang ditunjuk oleh Ptr ke prev *link* dari node berikut dari node tersebut

  Ptr->*next*->prev = Ptr->prev

5.4. Memindahkan Node ditengah ke Tengah (setelah *current*) (Lanj.)



5.4. Memindahkan Node ditengah ke Tengah (setelah *current*) (Lanj.)

- Isi prev *link* dari node yang ditunjuk oleh Ptr ke node yang ditunjuk oleh ptrTujuan

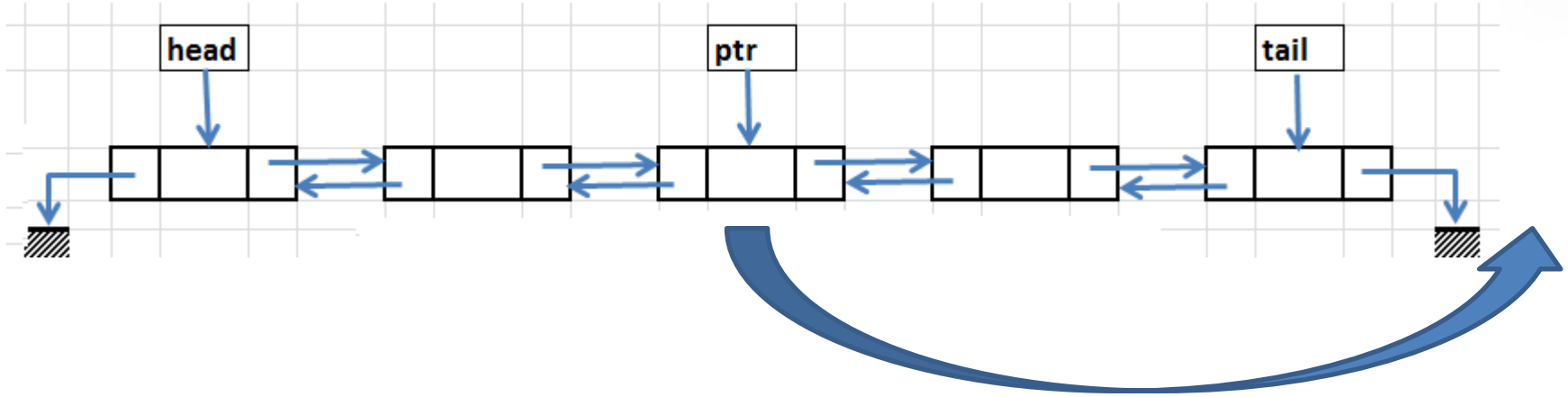
  Ptr->prev= ptrTujuan
- Isi *next link* dari node yang ditunjuk oleh Pke dengan isi dari *next link* node yang ditunjuk oleh variabel ptrTujuan

  Ptr->*next* = ptrTujuan->*next*
- Isi prev *link* dari node yang ditunjuk oleh next *link* Ptr ke ptr

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

  ptrTujuan->*next* = ptr

5.5. 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, serta alamat node didepan node tujuan ke variabel *pointer* ptrTujuan.

5.5. Memindahkan Node ditengah ke *Tail (lanj.)*

- Copykan *next link* dari node yang ditunjuk oleh Ptr ke *next link* dari node sebelum dari node tersebut

  Ptr->prev->*next* = Ptr->*next*

- Copykan prev *link* dari node yang ditunjuk oleh Ptr ke prev *link* dari node berikut dari node tersebut

  Ptr->*next*->prev = Ptr->prev

5.5. Memindahkan Node ditengah ke *Tail* (Lanj.)

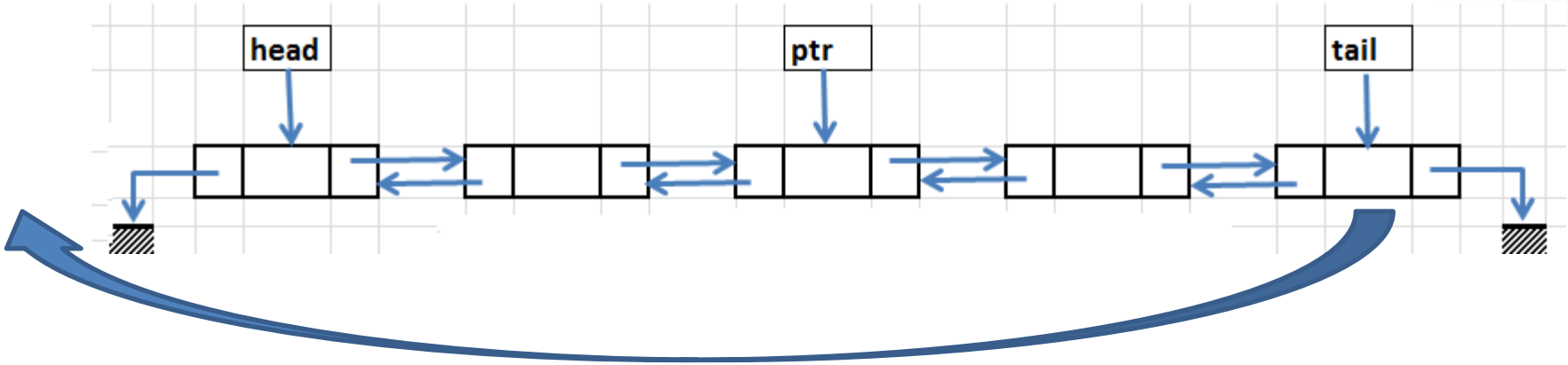
- Isi *next link* dari node yang ditunjuk oleh ptrTujuan dengan alamat node yang ditunjuk oleh variabel Ptr

  ptrTujuan->*next* = Ptr
- Isi prev *link* dari Ptr node yang ditunjuk oleh ptrTujuan

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

  Ptr->*next* = NULL

5.6. Memindahkan Node pada *Tail* ke *Head*



- Memindahkan Node dapat dilakukan dengan bantuan variabel *pointer* Ptr.
- Simpan alamat node terakhir (*Tail*) ke Ptr
- NULL kan *next link* dari node yang ditunjuk oleh node sebelum node yang akan dipindahkan

  Ptr->prev->*next*=NULL
- Arahkan *next link* dari node yang ditunjuk oleh Ptr ke *Head*

  Ptr->*next* = *Head*

5.6. Memindahkan Node pada *Tail* ke *Head (lanj.)*

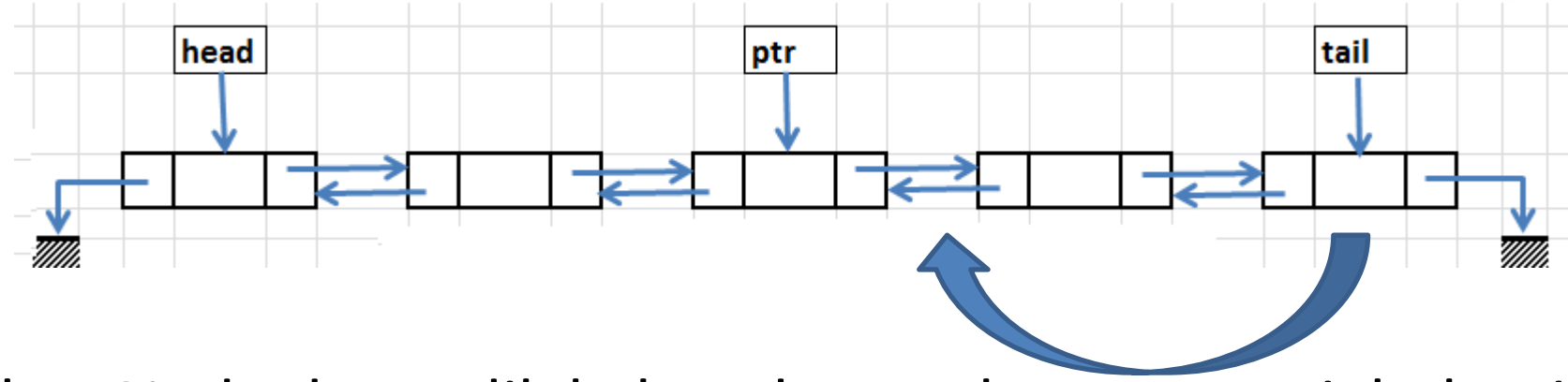
- Arahkan prev *link* dari *Head* ke node yang ditunjuk oleh Ptr

  Head->prev = Ptr
- Pindahkan *Head* ke node yang ditunjuk oleh Ptr

  Head = Ptr
- Jangan lupa NULL prev *link* dari *HEAD* yang baru

  Head->prev = NULL atau Ptr->prev = NULL
- Jangan lupa jika mempunyai variabel *Tail*, copykan ptrTujuan ke *Tail* (*Tail*=ptrAsal)

5.7. Memindahkan Node pada *Tail* ke Tengah



- Memindahkan Node dapat dilakukan dengan bantuan variabel *pointer* ptrTujuan serta sebuah variabel *pointer* Ptr.
- Simpan alamat node yang akan dipindahkan (*Tail*) ke variabel *pointer* Ptr serta simpan alamat node didepan node tujuan ke variabel *pointer* ptrTujuan.

5.7. Memindahkan Node pada *Tail* ke Tengah (lanj.)

- Karena yang dipindahkan ada pada *Tail*, proses pertama sebelum memindah *Tail* ke tempat tujuan, NULL kan terlebih dahulu node sebelum *Tail*

  Ptr->prev->*next* = NULL

- Isi prev *link* dari node yang ditunjuk oleh Ptr ke node yang ditunjuk oleh ptrTujuan

  Ptr->prev= ptrTujuan

5.7. Memindahkan Node pada *Tail* ke Tengah (Lanj.)

- Isi *next link* dari node yang ditunjuk oleh Pke dengan isi dari *next link* node yang ditunjuk oleh variabel ptrTujuan

  Ptr->*next* = ptrTujuan->next

- Isi prev *link* dari node yang ditunjuk oleh *next link* Ptr ke ptr

  ptrTujuan->*next*->prev=ptr atau Ptr->*next*->prev = ptr

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

  ptrTujuan->*next* = Ptr

Ringkasan

- Penggunaan *Double Linked List* dapat mengurangi pemakaian variabel yang diperlukan pada proses tambah, hapus dan pindah node
- Proses hapus, pindah dan tambah node pada *double linked-list* jangan sampai melupakan proses untuk memindahkan *previous link* dari node-node yang dipindahkan, dan node-node yang berhubungan pemindahan tersebut
- 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 Program

- Berikut ini contoh program double linked List, kali ini dengan memanfaatkan function isiCell()

Bagian deklarasi

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <string.h>

//enumerasi untuk membuat tipe data arah
enum Arah{kiri, kanan};

//Record Definition
struct TheCell
{
  int dat;
  struct TheCell *sebelum,*berikut;
};
```

Function IsiCell()

```
struct TheCell *isiCell(int dat, struct TheCell *cellCurrent,
                                 struct TheCell *kepala,
                                 struct TheCell *ekor,
                                 enum Arah arahIsi)
{
  struct TheCell *baru;
  baru=(struct TheCell *) malloc(sizeof(struct TheCell));
  if (kepala == NULL)
  {
    //Mengisi Linked List dari keadaan kosong
    cellCurrent = baru;
    cellCurrent->dat=dat;
    //Tentukan kedua Link  pada simpul/node baru
    //di tentukan sebagai NULL karena ini adalaha node perdana
    cellCurrent->sebelum = NULL;
    cellCurrent->berikut = NULL;
    return(cellCurrent);
  }
```

Function IsiCell() - Lanjutan

```
else //Mengisi pada Linked List yang sudah terbentuk
{
  if (arahIsi==kiri)
  {
    baru->berikut = cellCurrent;
    if (cellCurrent==kepala) //Penambahan di kepala
    {
      baru->sebelum = NULL;
    }
    else //penyisipan di kiri currentCell
    {
      baru->sebelum = cellCurrent->sebelum;
      cellCurrent->sebelum->berikut=baru;
    }
    cellCurrent->sebelum = baru;
    cellCurrent = cellCurrent->sebelum;
  }
```

Function IsiCell() - Lanjutan

```
else //kanan
    {
      baru->sebelum = cellCurrent;
      if (cellCurrent==ekor) //Penambahan di ekor
      {
        baru->berikut = NULL;
      }
      else //Penyisipan di kanan currentCell
      {
        baru->berikut = cellCurrent->berikut;
        cellCurrent->berikut->sebelum=baru;
      }
      cellCurrent->berikut = baru;
      cellCurrent = cellCurrent->berikut;
    }
    //Isi Data
    cellCurrent->dat=dat;
    return(cellCurrent);
  }
}
```

Program Utama

```
//program utama
void main()
{
  int i;
  struct TheCell *ptrCell=NULL;
  struct TheCell *kepala=NULL;
  struct TheCell *ekor=NULL;
  for (i=1;i<=10;i++)
  {
    if (kepala == NULL)
    {
      //Mengisi Linked List dari keadaan kosong
      ptrCell = isiCell(i, NULL, NULL, NULL, kanan);
      kepala = ptrCell;
      ekor = ptrCell;
    }
```

Program Utama (lanjutan)

```
  else
  {
    //Mengisi pada Linked List yang sudah terbentuk
    if (i % 2 == 0) //Genap ke kanan
    {
      ptrCell = isiCell(i, ekor, kepala, ekor, kanan);
      ekor = ptrCell;
    }
    else //ganjil ke kiri
    {
      ptrCell = isiCell(i, kepala, kepala, ekor, kiri);
      kepala = ptrCell;
    }
  }
}
```

Program Utama (lanjutan)

```
 //Menampilkan Isi Linked List dari awal
  if (kepala!=NULL)  //Memastikan Linked List tidak kosong, yg ditandai dengan kepala != NULL
  {
    cout<<"Kepala: "<<kepala<<endl;
    cout<<"Ekor: "<<ekor<<endl;
    cout<<"[ sebelum | {cell: Alamat} Data | berikut ]"<<endl;
    //Arahkan ptrCell ke alamat yang ditunjuk oleh kepala
    ptrCell = kepala;
    do
    {
      //Tampilkan isi Node / Simpul
      cout<<"["<<ptrCell->sebelum<<" | {cell: "<<ptrCell<<"}  "<<ptrCell->dat<<" | "<<ptrCell->berikut<<"]"<<endl;
      //Arahkan ptrCell ke Node/Simpul berikutnya
      ptrCell = ptrCell->berikut;
    }while (ptrCell != NULL); //keluar dari loop ketika ptrCell menunjukkan alamat ekor
    cout<<endl<<endl;
```

Program Utama (lanjutan)

```
    do
    {
      //Tampilkan isi Node / Simpul
      cout<<"["<<ptrCell->sebelum<<" | {cell: "<<ptrCell<<"}
           "<<ptrCell->dat<<" | "<<ptrCell->berikut<<"]"<<endl;
      //Arahkan ptrCell ke Node/Simpul berikutnya
      ptrCell = ptrCell->berikut;
    }while (ptrCell != NULL); //keluar dari loop
                              //ketika ptrCell menunjukkan alamat ekor
    cout<<endl<<endl;
  }
  getch();
}
```

Terimakasih

TUHAN Memberkati Anda

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