Doubly Linked List

Akmal, S.Si, MT

Mata Kuliah : Struktur Data

Tujuan

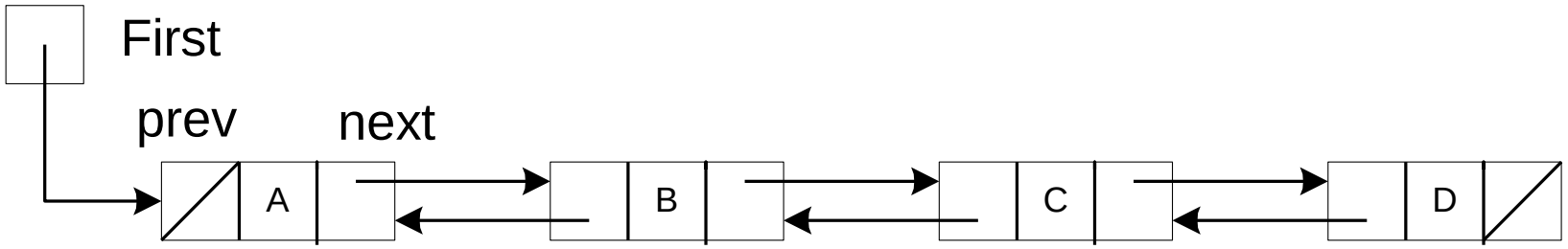
- Mahasiswa dapat : Mengoperasikan dan membuat program semua algoritma primitive list pada variasi list (doubly dan sirkular) dengan benar

Pokok Bahasan

- Operasi Traversal List doubly
- Menambah di depan, belakang dan tengah
- Menghapus di depan, belakang dan tengah
- Variasi List Berkepala
- Variasi List Circular

Pengertian List Berkait Doubly

- Doubly Linked List atau List berkait double adalah suatu bentuk tipe data abstrak yang merupakan kumpulan dari data berbentuk record dengan ciri setiap elemen data (record) memiliki karakteristik:
  - Ada informasi (*info*)
  - Ada 2 buah pengait (pointer) yaitu :
    - antara satu elemen dengan elemen yang berikutnya (*next*)
    - antara satu elemen dengan elemen yang sebelumnya (*prev*)



Pendeklarasian tipe struktur list berkait doubly

```
struct namaRecord {
   tipeInfo  info;                        //info
   . . . . . .
   namaRecord* namaPointer;    //namaPointer sbg pengait sebelum
   namaRecord* namaPointer;    //namaPointer sbg pengait sesudah
};

typedef namaRecord* Pointer;  //membuat tipe alias pointer
typedef Pointer List;         //membuat alias List
```

Contoh :

```
struct Node {
    char info;                    // char merupakan tipe info

    ElemtList* next;
    ElemtList* prev;
};

typedef Node* Pointer;
typedef Pointer List;
```

Operasi Pada List Berkait Doubly

Operasi dasar itu pada doubly linked list pada dasarnya hampir sama dengan singly linked list yaitu :

- Penyisipan/Insert
  - Penyisipan bisa dilakukan di depan (Insert First), di belakang (Insert Last) dan di tengah (Insert After dan Insert Before)
- Penghapusan/Delete
  - Penghapusan bisa dilakukan di depan (Delete First), di belakang (Delete Last) dan di tengah (Delete After, Delete Before dan DeletePCari)
- Penelusuran beruntun Linked List/Traversal
  - Mengunjungi semua elemen list dan biasanya dimulai dari elemen pertama
  - **Sama dengan Singly linked list**
- Pencarian elemen list/Searching
  - Melakukan searching berdasarkan suatu kunci untuk mencari apakah data yang diinginkan ada dalam list dan sekaligus untuk mendapatkan alamat dari elemen yang dicari.
  - **Sama dengan Singly Linked list**

Create List dan Create Elemen

```
void CreateList(List& First) {
   First = NULL;
}
```

Adapun langkah-langkah yang dilakukan dalam **create Element** adalah:

- alokasi suatu tempat yang dicatat oleh pointer pBaru
- Mengisi Info
- Memberi nilai NULL pada pointer pengait (next)
- Memberi nilai NULL pada pointer pengait (prev)

```
void createElement(pointer& pBaru) {
        pBaru = new Node;
        cin >> pBaru->info;      // pBaru->info.atributRecord
        pBaru->next = NULL;
        pBaru->prev = NULL;
}
```

Insert First Doubly

Insert First pada list doubly adalah proses penyisipan sebuah elemen list ke dalam sebuah list dimana penyisipan dilakukan di depan.

Secara umum langkah-langkah untuk operasi Insert First doubly adalah

a. Menciptakan elemen baru (Create Element)

b. Menambahkan elemen baru tersebut ke list dengan cara

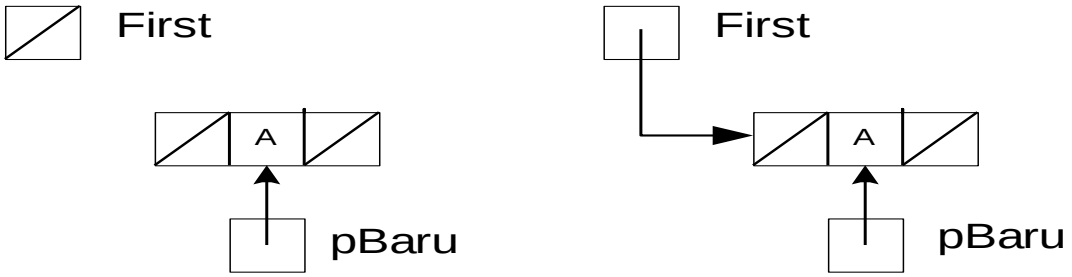
- Menambah sebuah elemen list baru di awal list atau di depan.
- Elemen baru tersebut menjadi elemen pertama dan di catat oleh pengenal list

Ada 2 kasus yang harus ditangani yaitu :

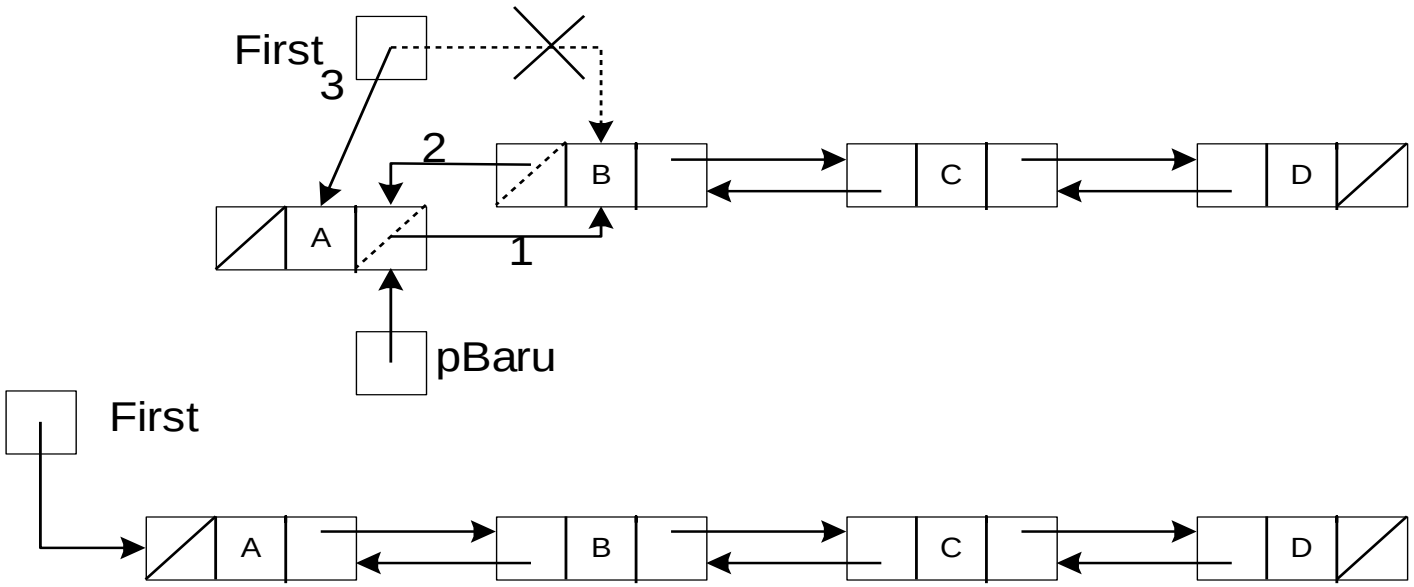
- Jika list masih kosong
- Jika list sudah ada isi

Insert First Doubly

- Kasus kosong



- Kasus ada isi



Fungsi Insert First Doubly

```
void insertFirstDoubly(List& First, pointer pBaru){
   // I.S  List First mungkin kosong  dan pBaru sudah
   //      terdefinisi
   // F.S  List bertambah satu elemen didepan dengan pBaru

   if (First==NULL)                          // kasus kosong
         First=pBaru;
   else {                                    //  kasus ada isi
         pBaru->next=First;                  //  1
         First -> prev = pbaru;              //  2
         First=pBaru;                        //  3
  }
}
```

Insert Last Doubly

- Insert Last Doubly adalah proses penyisipan sebuah elemen list ke dalam sebuah list dimana penyisipan dilakukan di belakang.
- Secara umum langkah-langkah untuk operasi Insert Last adalah sama seperti Insert First yaitu:

  a. Menciptakan elemen baru (Create Element)

  b. Menambahkan elemen baru tersebut ke list dengan cara
    - Menambah sebuah elemen list baru di akhir list atau di belakang.
    - Elemen baru tersebut menjadi elemen terakhir yang memiliki nilai next nya adalah NULL.

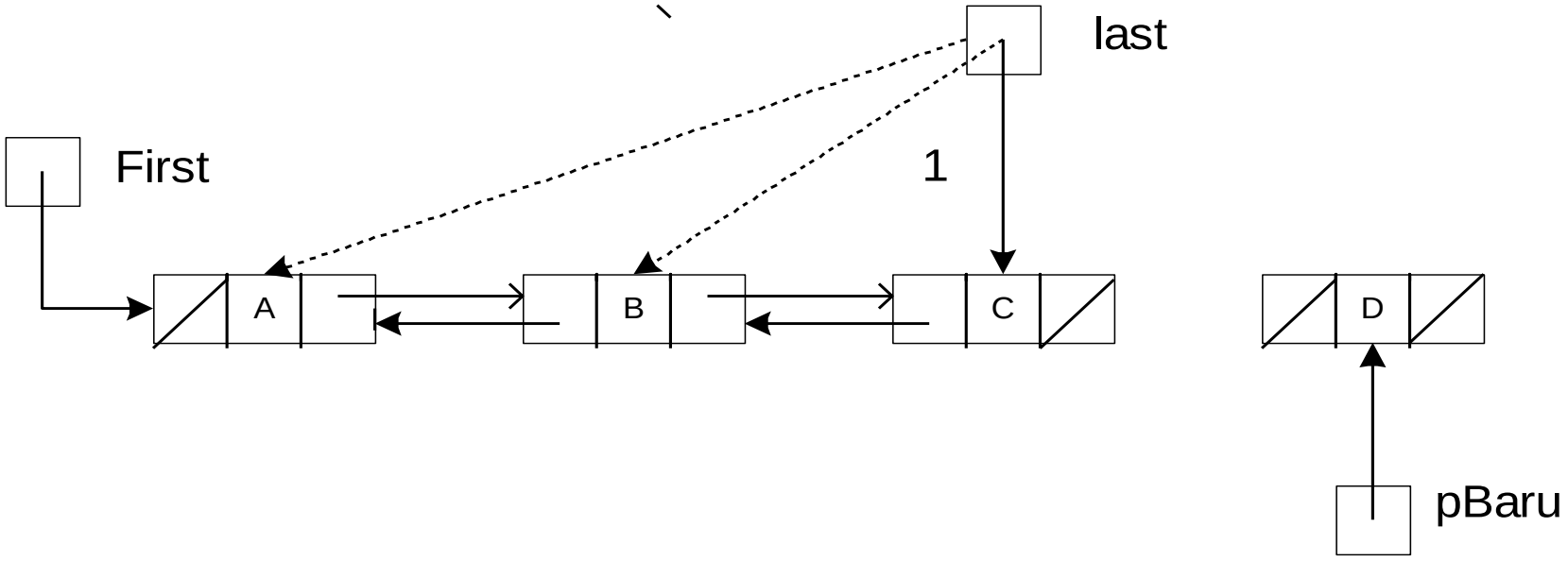
Ada 2 kasus yang harus ditangani yaitu :

- Jika list masih kosong (sama seperti Insert First)
- Jika list sudah ada isi

Insert Last Doubly

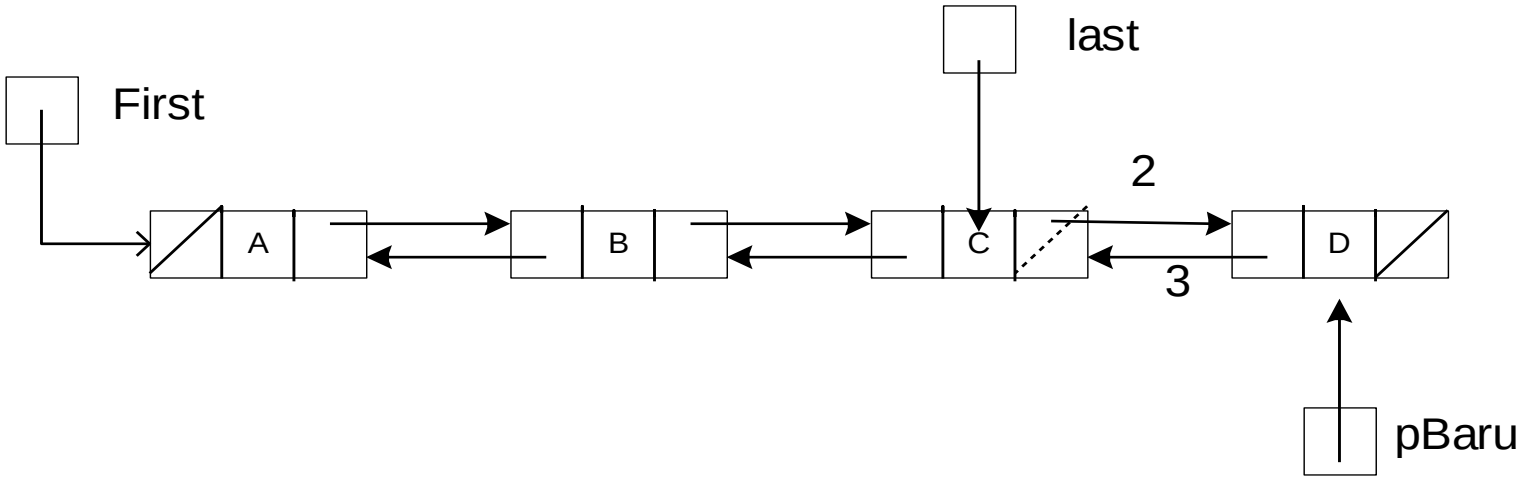
Kasus list ada isi :

- Langkah-langkah yang dilakukan adalah :
  1. Menemukan / mencari elemen terakhir dengan ciri yaitu elemen yang next-nya bernilai NULL. Pemeriksaan dilakukan mulai dari elemen pertama. Selanjutnya akan diperiksa elemen-elemen yang next-nya bernilai NULL. Gunakan pointer bantuan (last) untuk mencatat elemen terakhir tersebut.



Insert Last Doubly

2. Sambungkan elemen terakhir yang dicatat oleh last dengan elemen yang ditunjuk oleh pBaru



2. *last → next = pbaru;*
3. *pbaru → prev = last;*

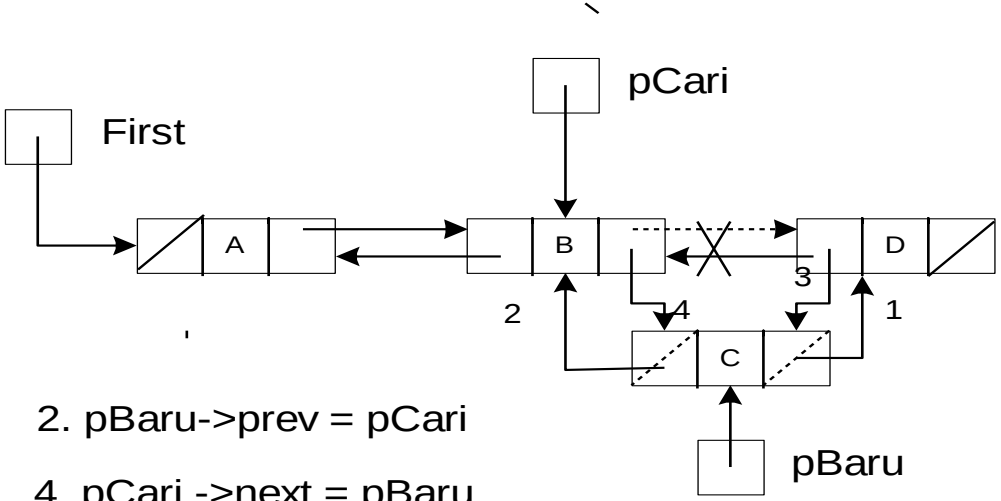
Fungsi InsertLast Doubly

```
void InsertLastDoubly(List& First, pointer pBaru){
   // I.S  List First mungkin kosong  dan pBaru sudah
   //      terdefinisi
   // F.S  List bertambah dibelakang (pBaru sesudah last)

   pointer last;    // last untuk mencatat elemen terakhir

   cout<<"Insert Last"<<endl;
   if (First==NULL){                    //kosong
       First=pBaru;
   }
   else {                               // ada isi
     last=First;
     while (last->next!=NULL){
          last=last->next;
     }

     last->next=pBaru;                  // sambungkan
     pBaru->prev = last;
   }
}
```

Insert After Doubly / Penyisipan Di Tengah

- Insert After doubly adalah primitif untuk menambah sebuah elemen list baru setelah elemen tertentu pada list berkait doubly.
- Langkah-langkah penyisipan :

1. Temukan suatu elemen (Searching) dimana elemen yang baru akan disisipkan sesudah elemen tersebut.
2. Jika ditemukan, maka ada 2 kasus yang mungkin terjadi yaitu
   - Elemen yang dicari berada di belakang sehingga akan terjadi penyisipan di belakang (insert Last)
   - Elemen yang dicari berada di tengah sehingga elemen yang baru akan berada diantara 2 buah elemen (di tengah)



1. pBaru ->next=pCari ->next

2. pBaru->prev = pCari

3. pBaru->next->prev=pBaru

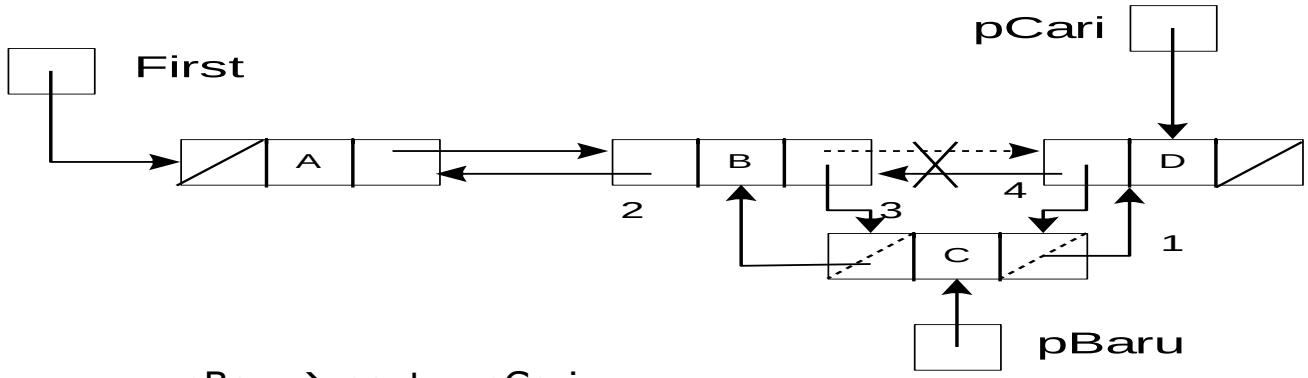
4. pCari ->next = pBaru

Insert Before Doubly / Penyisipan Di Tengah

- Insert Before doubly adalah primitif untuk menambah sebuah elemen list baru sebelum elemen tertentu pada list berkait doubly.

Langkah-langkah penyisipan :

1. Temukan suatu elemen (Searching) dimana elemen yang baru akan disisipkan sesudah elemen tersebut.
2. Jika ditemukan, maka ada 2 kasus yang mungkin terjadi yaitu
   - Elemen yang dicari berada di awal list sehingga akan terjadi penyisipan di depan (insert First)
   - Elemen yang dicari berada di tengah sehingga elemen yang baru akan berada diantara 2 buah elemen (di tengah)



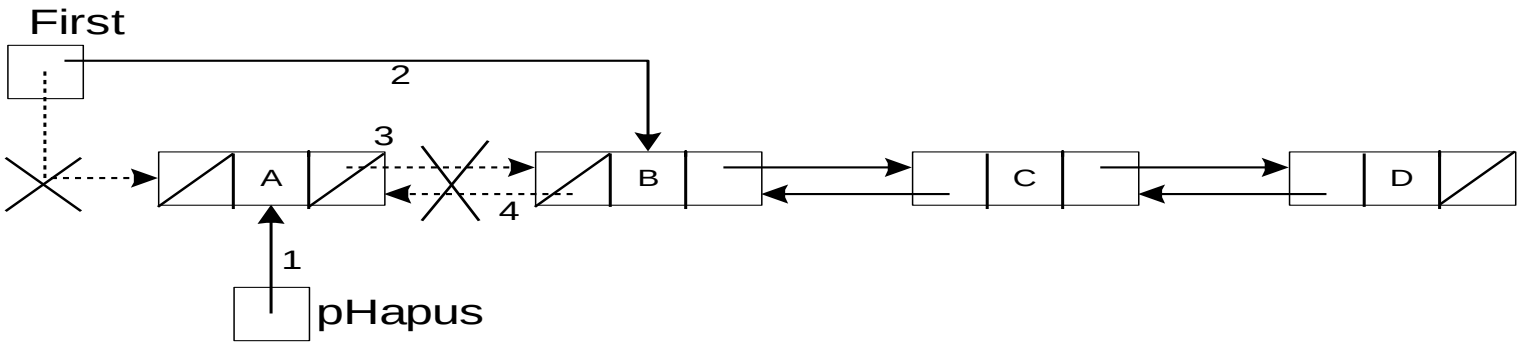
1. pBaru→ next = pCari;
2. pBaru→ prev = pCari → prev;
3. pCari→ prev → next = pBaru;
4. pCari→ prev = pBaru;

Delete First Doubly / Penghapusan Di Depan

- Merupakan penghapusan sebuah elemen yang ada di depan. Elemen List yang akan dihapus adalah elemen yang dicatat / ditunjuk oleh First

Secara umum ada 3 kasus yang perlu ditangani untuk operasi Delete First yaitu:

a. Kasus List kosong
b. Kasus List dengan isi 1 elemen
c. Kasus List dengan isi lebih dari 1 elemen
   - Menghapus elemen yang depan / pertama yang alamatnya dicatat oleh First,, sehingga list akan berkurang 1 elemen.
   - Setelah elemen tersebut dihapus maka elemen berikutnya menjadi elemen pertama yang di tunjuk oleh First



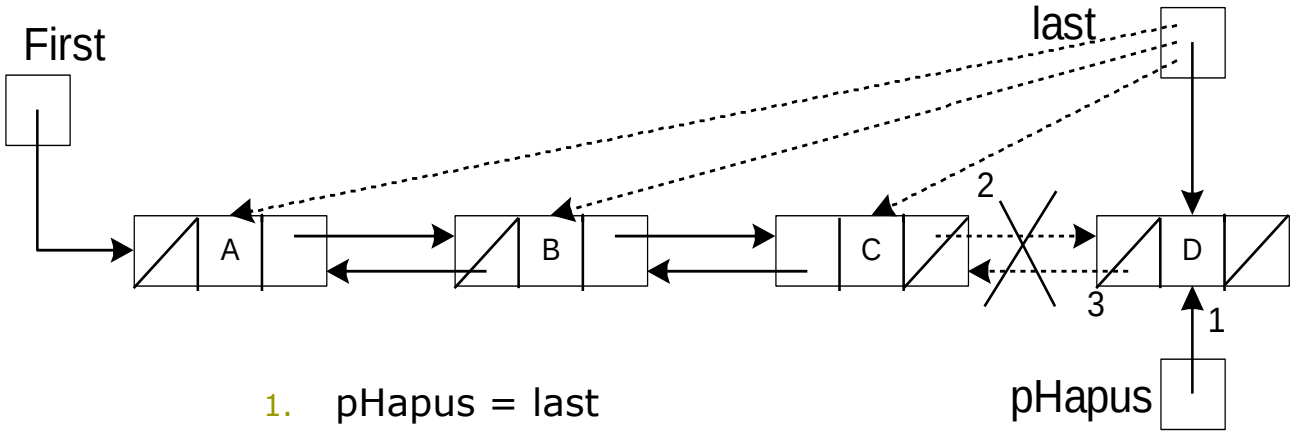
1. pHapus = First
2. First=First->next
3. pHapus->next=NULL
4. First->prev=NULL

Delete Last Doubly / Penghapusan Di Belakang

- Merupakan penghapusan sebuah elemen yang ada di belakang. Elemen List yang akan dihapus adalah elemen yang nextnya bernilai NULL.

Secara umum ada 3 kasus yang perlu ditangani untuk operasi Delete Last yaitu:

a. Kasus List kosong
b. Kasus List dengan isi 1 elemen
c. Kasus List dengan isi lebih dari 1 elemen
   - Menemukan / mencari elemen terakhir dengan ciri yaitu elemen yang next-nya bernilai NULL dengan menggunakan pointer bantuan (last)
   - Tidak perlu precLast untuk penghapusan elemen terakhir



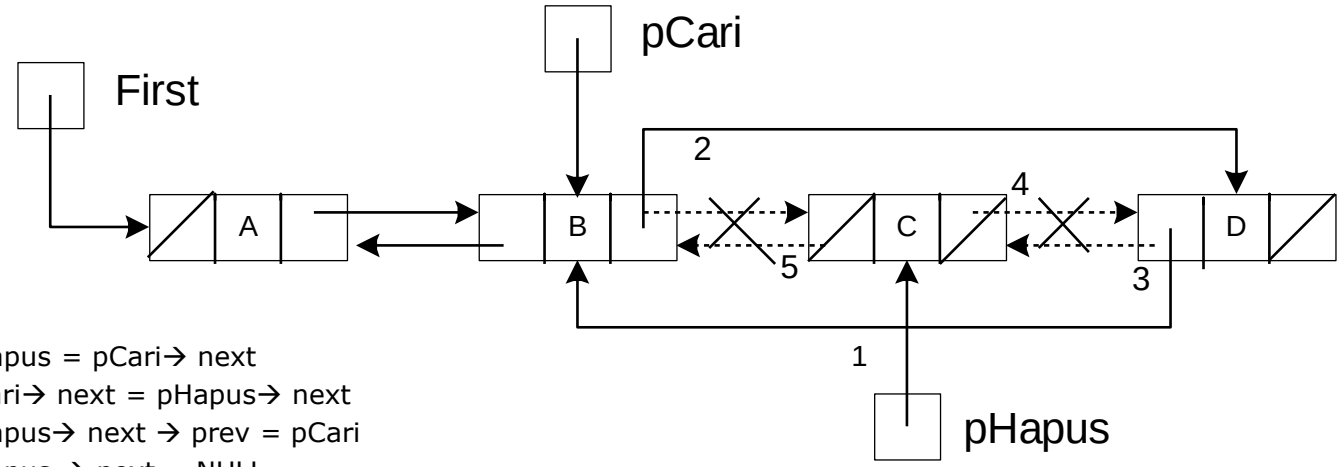
1. pHapus = last
2. last->prev->next = NULL
3. pHapus->prev = NULL

Delete After Doubly / Penghapusan Di Tengah

- Menghapus sebuah elemen list setelah elemen tertentu

Langkah-langkah penghapusan :

1. Temukan suatu elemen dimana elemen yang akan dihapus berada sesudah elemen tersebut.
2. Jika ditemukan, maka ada 3 kasus yang mungkin terjadi yaitu
   a. Elemen yang dicari (pCari) berada di belakang sehingga tidak ada yang akan dihapus sesudah pCari tersebut.
   b. pCari menunjuk elemen sebelum yang terakhir maka akan dilakukan penghapusan di belakang (delete Last)
   c. pCari berada pada selain a dan b.
      - pHapus adalah elemen sesudah pCari
      - lepaskan kaitan antara yang ditunjuk oleh pCari dengan yang akan dihapus
      - putuskan kaitan antara yang ditunjuk oleh pHapus dengan elemen sesudahnya



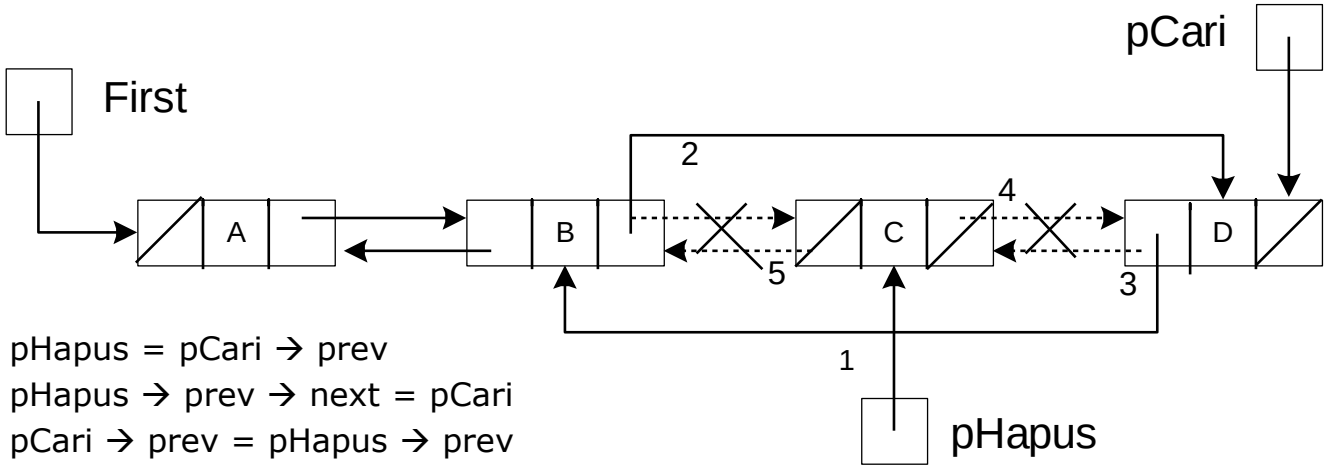
1. pHapus = pCari→ next
2. pCari→ next = pHapus→ next
3. pHapus→ next → prev = pCari
4. pHapus → next = NULL
5. pHapus → prev = NULL

Delete Before Doubly / Penghapusan Di Tengah

- Menghapus sebuah elemen list sebelum elemen tertentu

Langkah-langkah penghapusan :

1. Temukan suatu elemen dimana elemen yang akan dihapus berada sebelum elemen tersebut.
2. Jika ditemukan, maka ada 3 kasus yang mungkin terjadi yaitu
   a. Elemen yang dicari (pCari) berada di depan sehingga tidak ada yang akan dihapus sebelum pCari tersebut.
   b. pCari menunjuk elemen sesudah First (yang kedua) maka akan dilakukan penghapusan di depan (delete First)
   c. pCari berada pada selain a dan b.
      - pHapus adalah elemen sebelum pCari
      - lepaskan kaitan antara yang ditunjuk oleh sebelum pHapus dengan yang akan dihapus
      - putuskan kaitan antara yang ditunjuk oleh pHapus dengan elemen sesudahnya



1. pHapus = pCari → prev
2. pHapus → prev → next = pCari
3. pCari → prev = pHapus → prev
4. pHapus → next = NULL
5. pHapus → prev = NULL

Delete pCari / Penghapusan Di Tengah

Menghapus sebuah elemen list yang dicari

Langkah-langkah penghapusan :

1. Temukan suatu elemen dimana elemen yang yang akan dihapus adalah elemen itu sendiri.
2. Jika ditemukan, maka ada 3 kasus yang mungkin terjadi yaitu

a. Elemen yang dicari (pCari) berada di depan (== First), maka dilakukan penghapusan di depan (deleteFirst)

Memiliki 2 buah sub kasus yaitu :

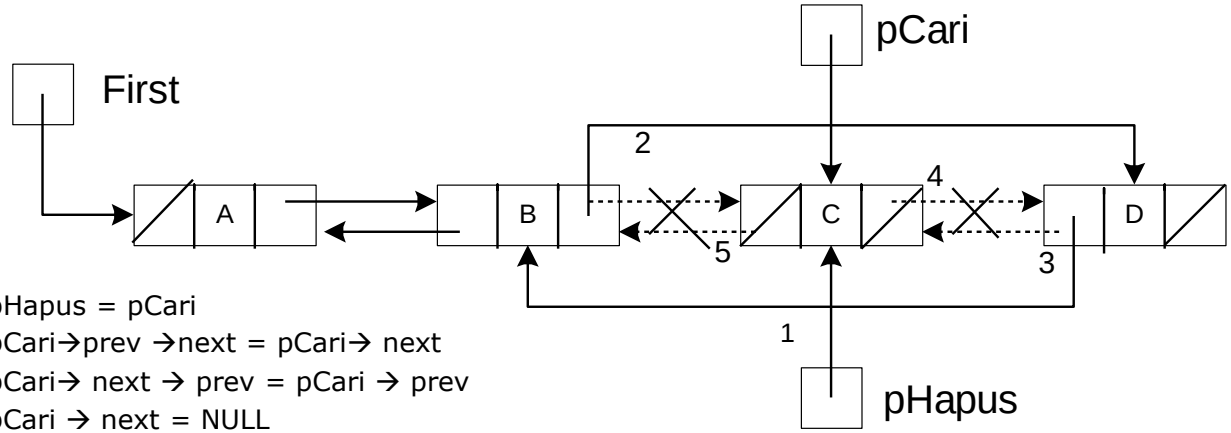
List isinya 1 elemen

List isinya lebih dari 1 elemen

b. Elemen yang dicari berada di belakang maka akan dilakukan penghapusan di belakang (delete Last)

c. pCari berada diantara 2 buah elemen (selain a dan b).

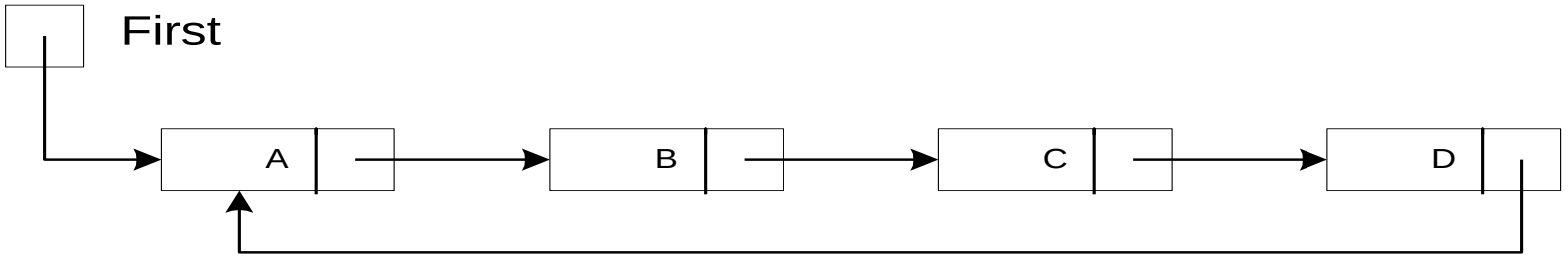
- pHapus adalah elemen yang ditunjuk oleh pCari
- lepaskan kaitan antara yang ditunjuk oleh sebelum pHapus dengan yang akan dihapus
- putuskan kaitan antara yang ditunjuk oleh pHapus dengan elemen sesudahnya



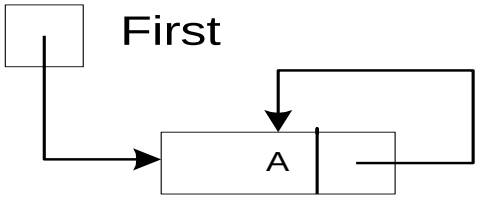
1. pHapus = pCari
2. pCari→prev →next = pCari→ next
3. pCari→ next → prev = pCari → prev
4. pCari → next = NULL
5. pCari → prev = NULL

List Sirkular Singly

- List circular dg isi lebih dari 1 elemen



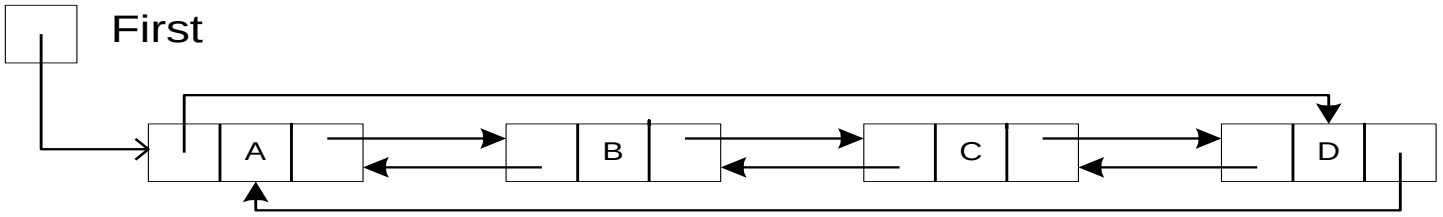
Kosong



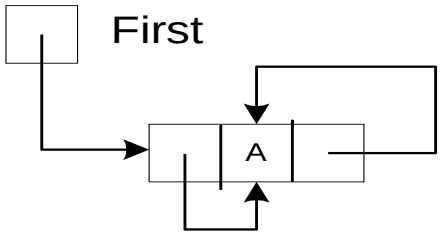
isi 1 elemen

List Sirkular Doubly

List circular dg isi lebih dari 1 elemen



Kosong



isi 1 elemen

Latihan (30 menit)

- Buat list doubly Mahasiswa (npm,nama,nilai):
  - Insert First, Delete Last, Insert After, Delete PCari

Tugas

Buatlah program lengkap semua primitive list doubly linked list dengan pengelolaan menggunakan suatu menu. Kasus list yang ditangani adalah list pegawai dengan atribut NIP, nama, alamat, gol, dan gaji

Gunakan 2 struct :

struct Pegawai dan struct Node