Operasi Dasar Singly linked List (Primitive List)

Akmal, S.Si, MT

Mata Kuliah : Struktur Data

Tujuan

- Mahasiswa dapat : Mengoperasikan dan membuat program dari semua algoritma primitive list singly (insert, delete, traversal dan searching) dengan benar.

Operasi Dasar pada Singly linked list (Primitive List)

- Penyisipan/Insert
  - Penyisipan bisa dilakukan di depan (Insert First), di belakang (Insert Last) dan di tengah (Insert After)
- Penghapusan/Delete
  - Penghapusan bisa dilakukan di depan (Delete First), di belakang (Delete Last) dan di tengah (Delete After)
- Penelusuran beruntun Linked List/Traversal
  - Mengunjungi semua elemen list dan biasanya dimulai dari elemen pertama
- 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.

Create List dan Create Elemen

- Langkah pertama sebelum operasi list dilakukan adalah membuat sebuah fungsi untuk membuat sebuat list berkait kosong (**CreateList**).

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

- Proses yang dilakukan dalam **create Element** adalah :
  - alokasi suatu tempat yang dicatat oleh pointer pBaru
  - Mengisi Info
  - Memberi nilai NULL pada pointer pengait (next)
- Selanjutnya elemen yang memenuhi kondisi ini disebut sebagai elemen yang sudah terdefinisi.

```
void CreateElmt(pointer& pBaru) {
   pBaru = new ElemtList;                    // alokasi
   cout << "Masukkan satu huruf : ";
   cin >> pBaru->info;                       // isi info
   pBaru->next = NULL;                       //
}
```

Insert First (Penyisipan Di Depan)

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

- Secara umum langkah-langkah untuk operasi Insert First 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

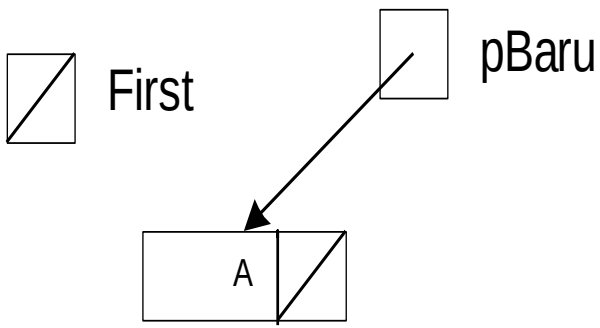
- Ada 2 kasus yang harus ditangani yaitu :
  - Jika list masih kosong
  - Jika list sudah ada isi

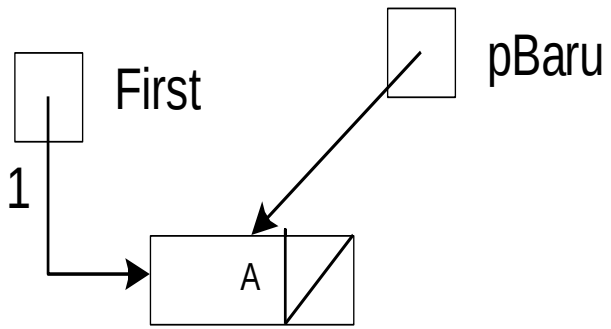
Insert First pada List Kosong

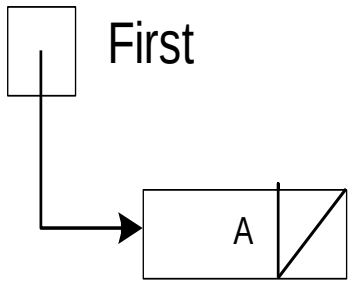
- Ada sebuah list yang kosong dengan ciri First==NULL
- Ada sebuah elemen yang sudah terdefinisi yang ingin disisipkan
- Langkah-langkah yang harus dilakukan adalah dengan menjadikan First menunjuk ke elemen yang dicatat oleh pBaru sebagai bagian dari list.

Initial State
(Keadaan awal)

Final State
(Keadaan akhir)



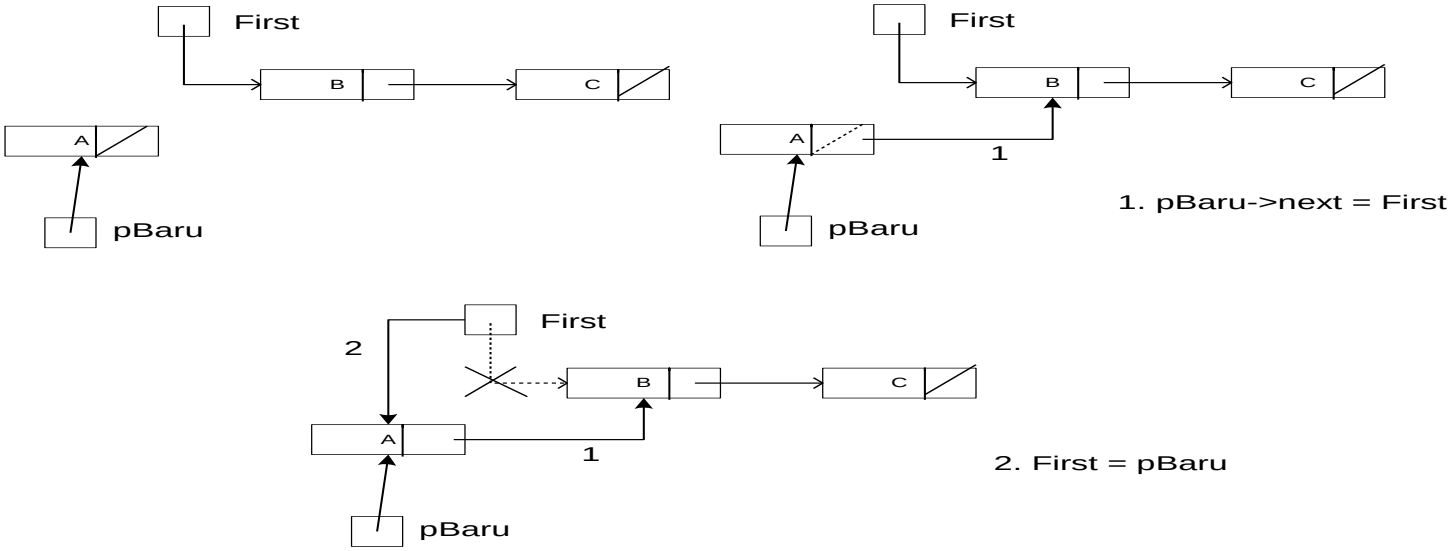




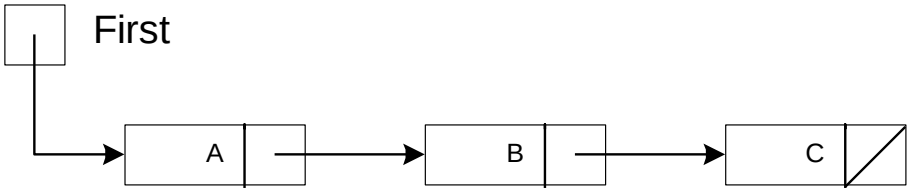
1. First = pBaru

Insert First pada List Sudah Ada Isi

- Keadaan Awal



- Setelah fungsi selesai bekerja maka pointer pBaru akan lepas (karena bersifat lokal)
- Keadaan Akhir



Fungsi Insert First

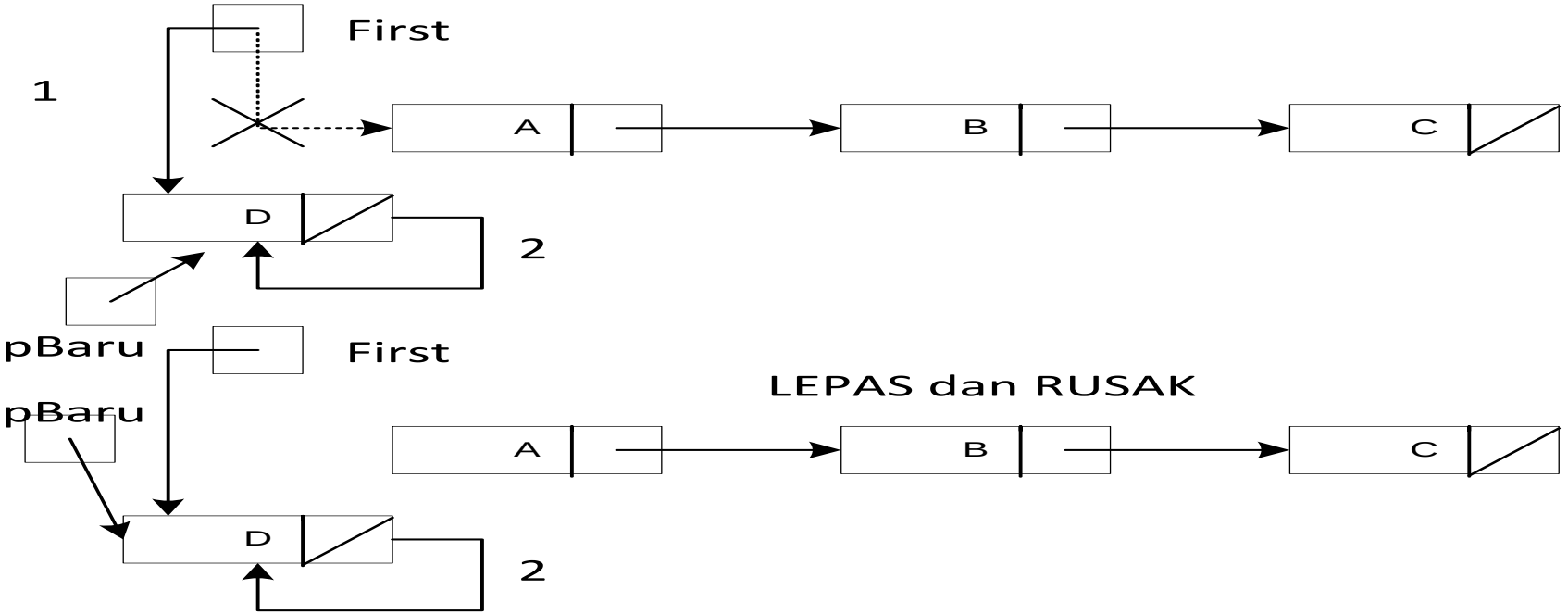
```
void insertFirst(List& First, pointer pBaru){
   // I.S  List First mungkin kosong  dan pBaru sudah terdefinisi
   // F.S  List bertambah satu elemen di depan dengan pBaru

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

- **Hati-hati :** ketika menganalisis keadaan list dengan ada isi, **JANGAN sampai tertukar** antara tahap 1 dengan tahap 2, karena akibatnya list akan menjadi rusak.

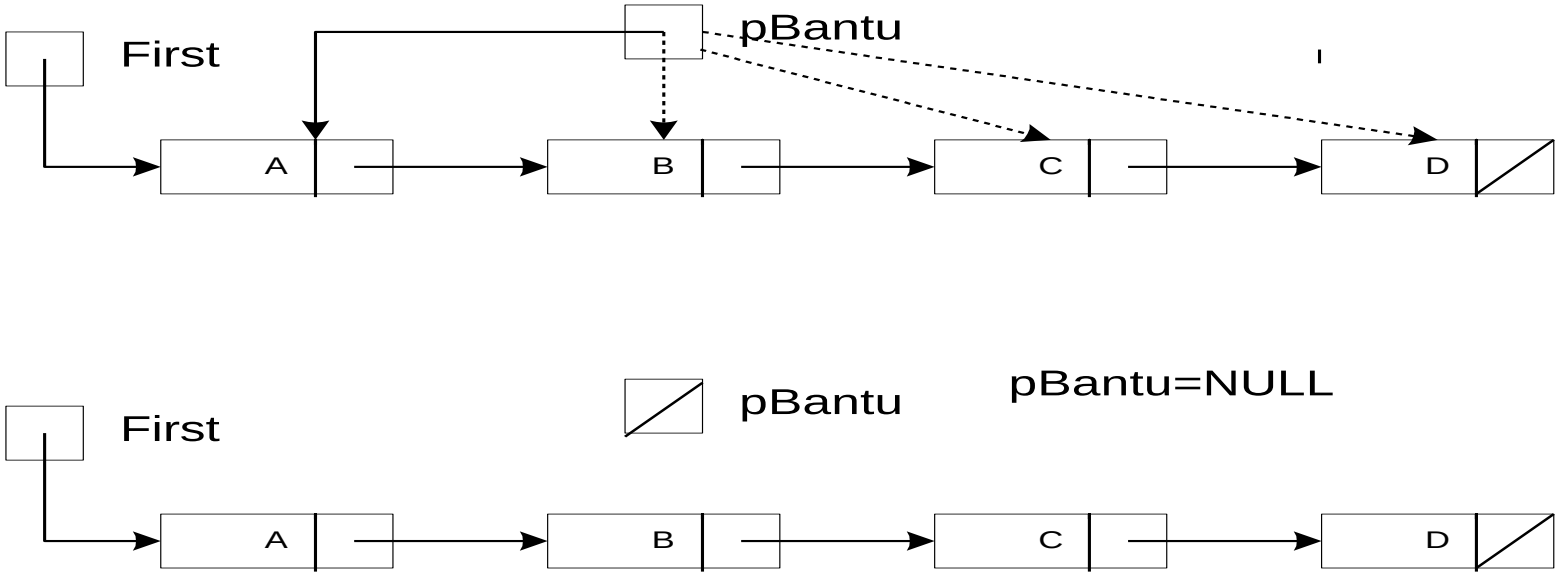
Kesalahan algoritma Insert First

- Perhatikan gambaran keadaan list berikut ketika keadaan ini dilakukan yaitu :
  - First = pBaru
  - pBaru->next = First



Traversal

- Traversal berarti menelusuri / mengunjungi semua elemen list berkait.
- Dalam hal ini kunjungan dimulai dari elemen pertama (yang dicatat oleh First).
- Untuk itu diperlukan suatu pointer Bantuan (pBantu) untuk mendatangi setiap elemen satu persatu.
- Selama pointer bantuan tidak bernilai NULL artinya masih ada elemen yang dikunjungi maka elemen tersebut bisa diproses (dicetak, dll).
- Setelah satu elemen selesai di proses maka pointer bantuan akan berpindah ke elemen berikutnya



Fungsi Traversal

```
void traversal(List First){
   // I.S  List mungkin kosong
   // F.s  Semua elemen list didatangi dan ditampilkan mulai dari elemenpertama

   pointer pBantu;
   pBantu=First;                              // Catat elemen pertama
   while(pBantu != NULL) {
       cout << pBantu->info;                  // proses cetak
      pBantu = pBantu->next;                  // berpindah ke berikutnya
   }
}
```

- **Untuk penanganan kasus kosong** secara khusus maka bisa dilakukan dengan suatu pola algoritma sbb :

```
void traversal(List First){
      pointer pBantu;
   if (First==NULL) {
       cout << "List kosong "<<endl;
   }
   else {
        pBantu=First;                         // Catat elemen pertama
        do {
             cout << pBantu->info;            // proses cetak
             pBantu = pBantu->next;           // pindah ke berikutnya
        } while(pBantu != NULL);
   }
}
```

Fungsi Utama (main)

- Dengan menggunakan fungsi-fungsi yang sudah didefinisikan sebelumnya (create, insertFirst dan traversal) maka semua fungsi tersebut bisa dirangkai dalam sebuah program utama sebagai berikut :

```
main() {
    pointer p;
    List TI21;

    createList(TI21);                    // List kosong

    createElmt(p);                       // C
    insertFirst(TI21,p);
    traversal(TI21);                     // C

    createElmt(p);                       // B
    insertFirst(TI21,p);
    traversal(TI21);                     // B  C

    createElmt(p);                       // A
    inserFirst(TI21,p);
    traversal(TI21);                     // A  B  C
  }
```

Delete First (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:

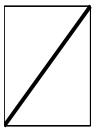
  a. Kasus List kosong
  - Tidak ada elemen yang dihapus

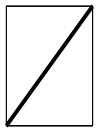
  b. Kasus List dengan isi 1 elemen
  - Elemen pertama yang dicatat oleh First, dihapus sehingga list menjadi kosong

  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

Delete First pada Kasus Kosong

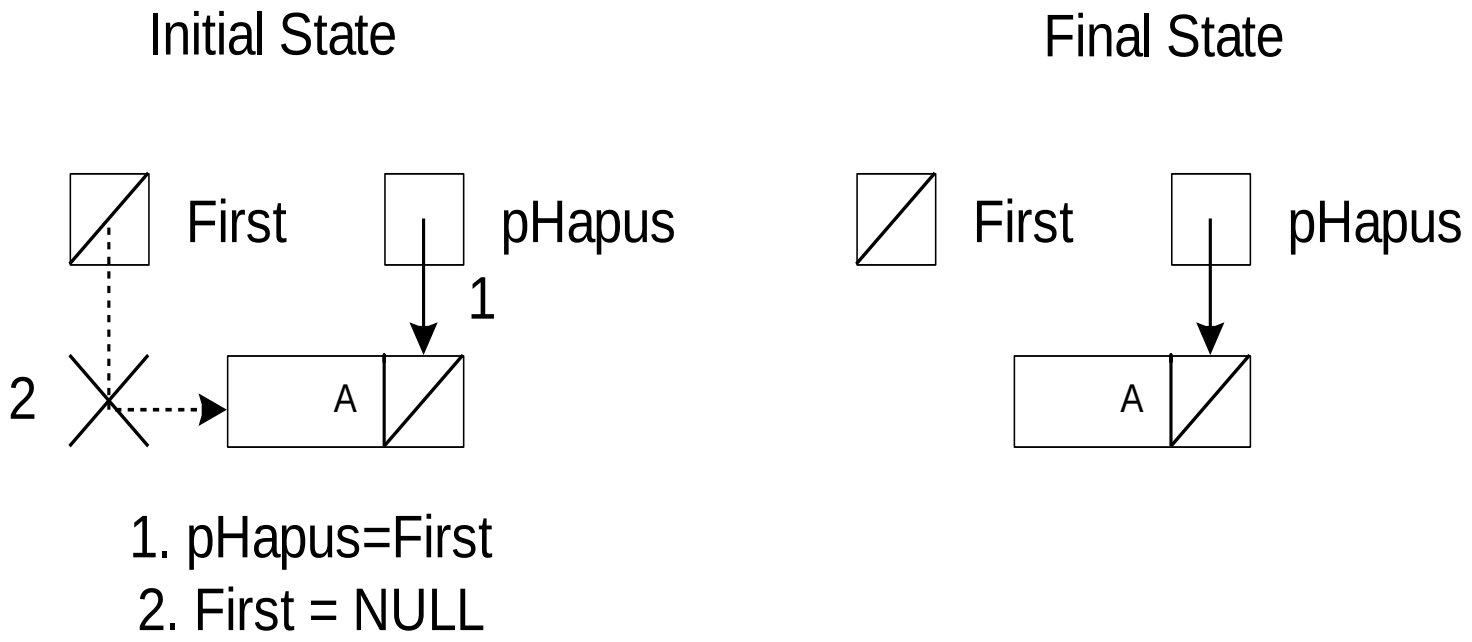
- Tidak ada elemen yg dihapus





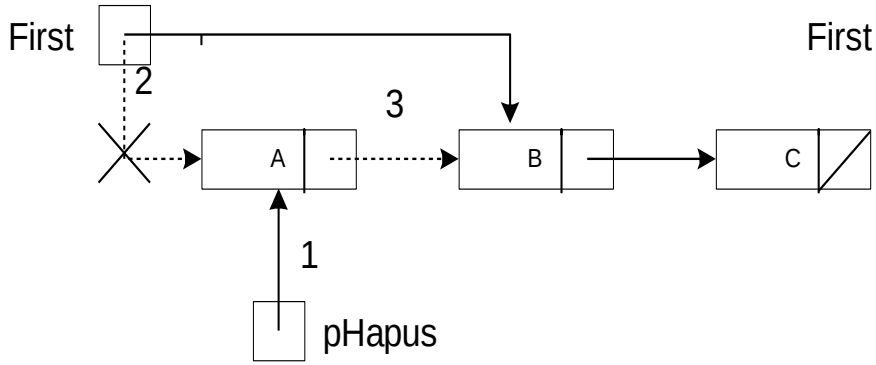
pHapus=NULL

Delete First pada Kasus Isi 1 Elemen

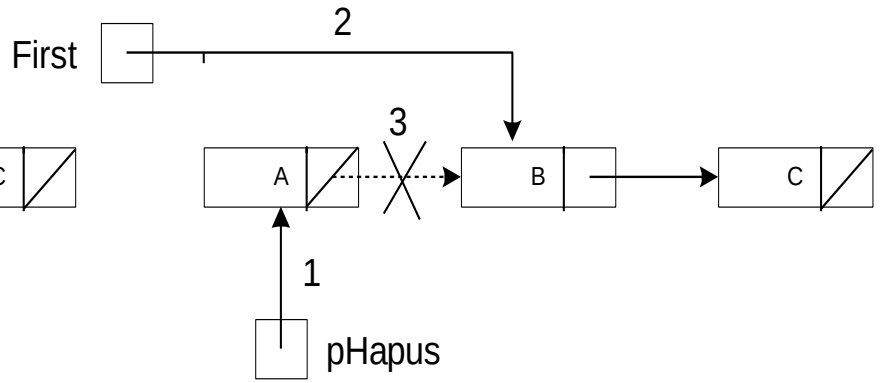


Delete First pada Kasus Isi > 1 Elemen

Initial State

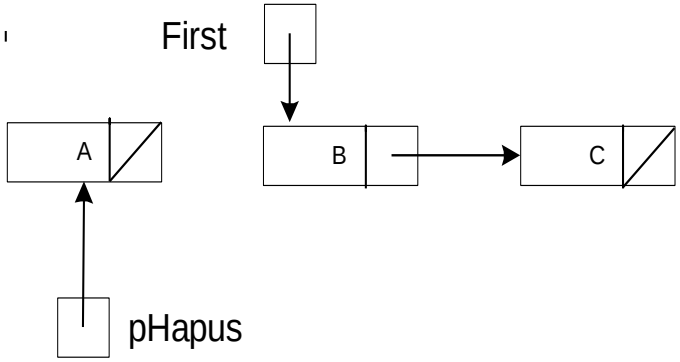


1. pHapus = First



2. First = First->next
3. pHapus->next = NULL

Final State



Fungsi Delete First

```
void deleteFirst(List& First, pointer& pHapus){
   // I.S  List First mungkin kosong
   // F.S. List berkurang satu di depan, yang dihapus
   //      dikembalikan

   if (First==NULL){                          // kosong
         pHapus=NULL;
         cout<<"List kosong, tidak ada yang dihapus"<<endl;
   }
   else if (First->next==NULL){               //isi 1 elemen
         pHapus=First;
         First=NULL;
   }
   else {                                     // isi > 1 elemen
      pHapus=First;                           // 1
      First=First->next;                      // 2
      pHapus->next=NULL;                      // 3
   }
}
```

- **Hati-hati** untuk penghapusan dengan isi list lebih dari 1 elemen, **JANGAN tertukar** antar tahap 2 dengan 3 karena bisa mengakibatkan list menjadi KOSONG! (kenapa ??)

Insert Last (Penyisipan Di Belakang)

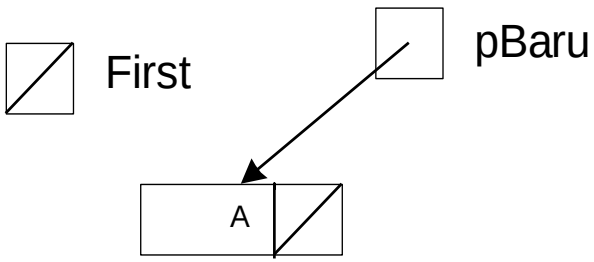
- Insert Last adalah proses penyisipan sebuah elemen list ke dalam sebuah list dimana penyisipan dilakukan di belakang.
- Secara umum langkah-langkah untuk operasi Insert Last adalah sama seperti Insert First yaitu:

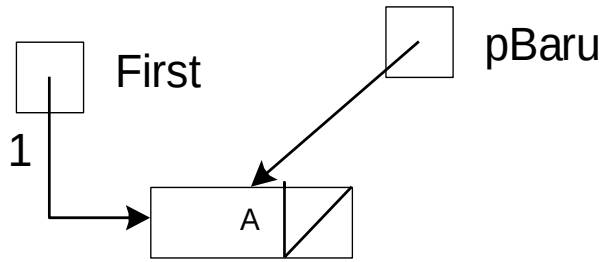
  a. Menciptakan elemen baru (Create Element)

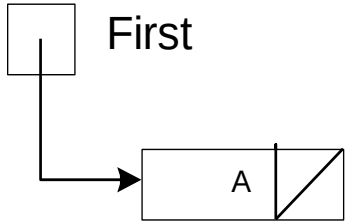
  b. Menambahkan elemen baru tersebut ke list dengan cara
  - Menambah sebuah elemen list baru di akhir list atau di belakang.
  - Elemen baru tersebut menjadi elemen terakhir yang memiliki nilai next nya adalah NULL.
- Ada 2 kasus yang harus ditangani yaitu :
  - list kosong
  - list sudah ada isi

Insert Last pada Kasus Kosong

- Ada sebuah list yang kosong dengan ciri First==NULL
- Ada sebuah elemen yang sudah terdefinisi yang ingin disisipkan
- Langkah-langkah yang harus dilakukan adalah dengan menjadikan First menunjuk ke elemen yang dicatat oleh pBaru sebagai bagian dari list.





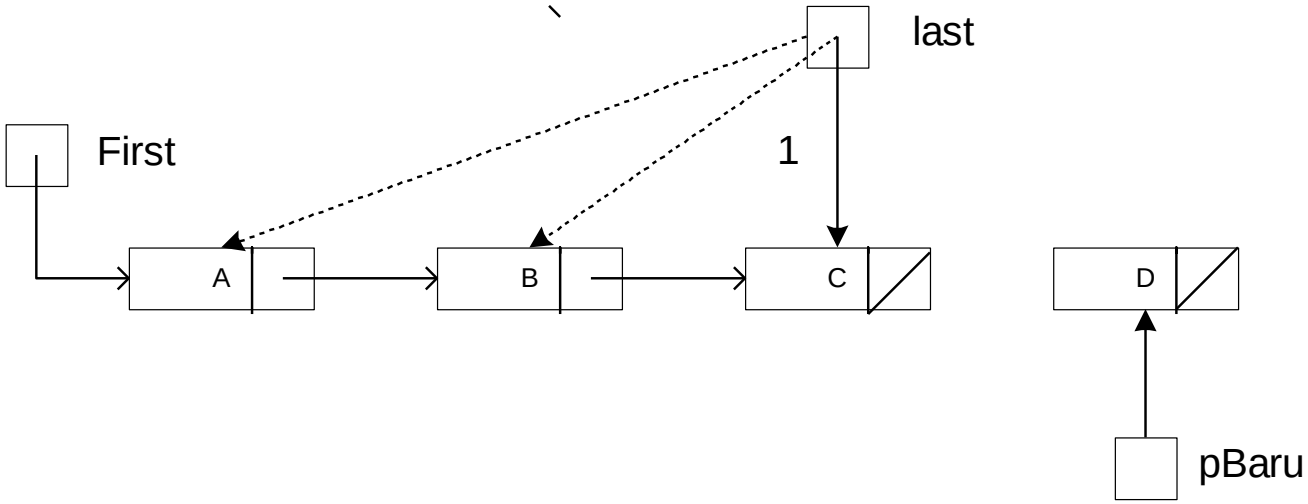


1. First = pBaru

Insert Last pada Kasus List Ada Isi (1)

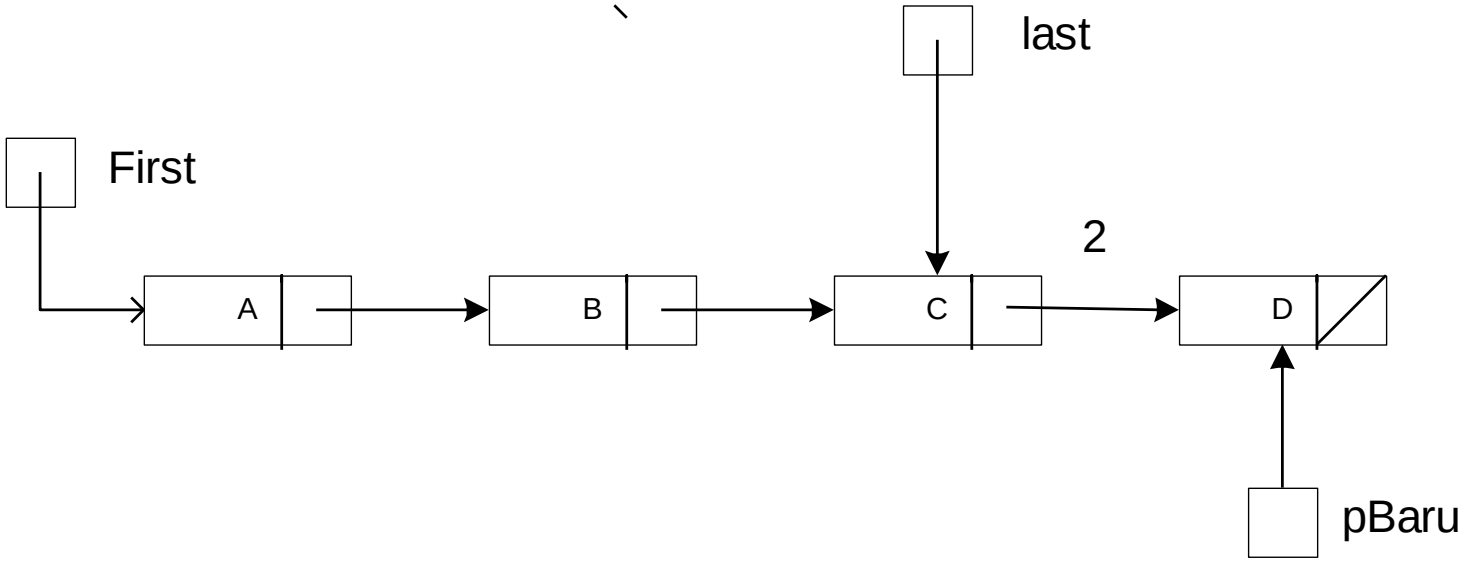
- Langkah-langkah yang dilakukan adalah :
  1. Menemukan / mencari elemen terakhir dengan ciri yaitu elemen yang next-nya bernilai NULL. Pemeriksaan dilakukan mulai dari elemen pertama. Selanjutnya akan diperiksa elemen-elemen yang next-nya bernilai NULL. Gunakan pointer bantuan (last) untuk mencatat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

```
last = First;
while (last->next != NULL) {
      last = last->next;
}
```



Insert Last pada Kasus List Ada Isi (2)

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



2. last->next = pBaru

Fungsi Insert Last

```
void insertLast(List& First, pointer pBaru){
   // I.S  List First mungkin kosong  dan pBaru sudah terdefinisi
   // F.S  List bertambah satu elemen di belakang (sesudah last)

   pointer last;              //last utk mencatat elemen terakhir

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

Delete Last (Penghapusan Di Belakang) (1)

- 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
   - Tidak ada elemen yang dihapus

b. Kasus List dengan isi 1 elemen
   - Elemen pertama yang dicatat oleh First, dihapus sehingga list menjadi kosong

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)

```
last = First;
while (last->next != NULL) {
    last = last->next;
}
```

Delete Last (Penghapusan Di Belakang) (2)

- Untuk bisa memutuskan kaitan antara elemen terakhir dengan elemen sebelum yang terakhir maka diperlukan lagi suatu pointer bantuan lagi (precLast)

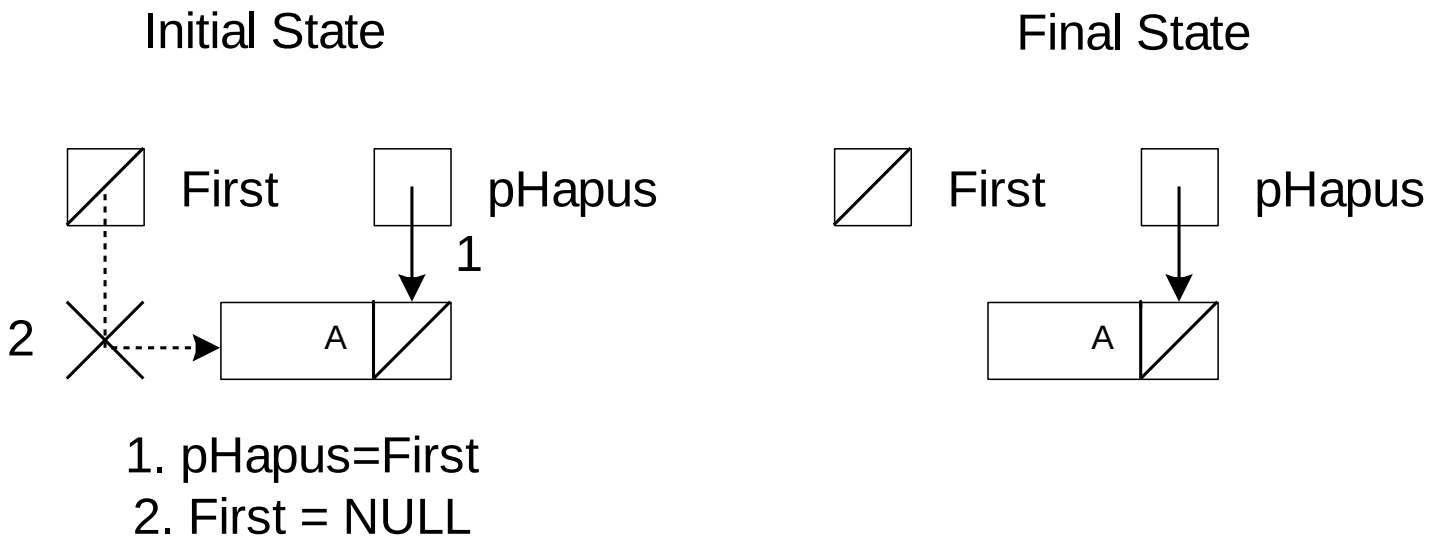
```
precLast = First;
while (precLast->next != last) {
        precLast = precLast->next;
}
```

- Menemukan kedua alamat ini bisa dilakukan dalam satu tahap looping yaitu sbb :

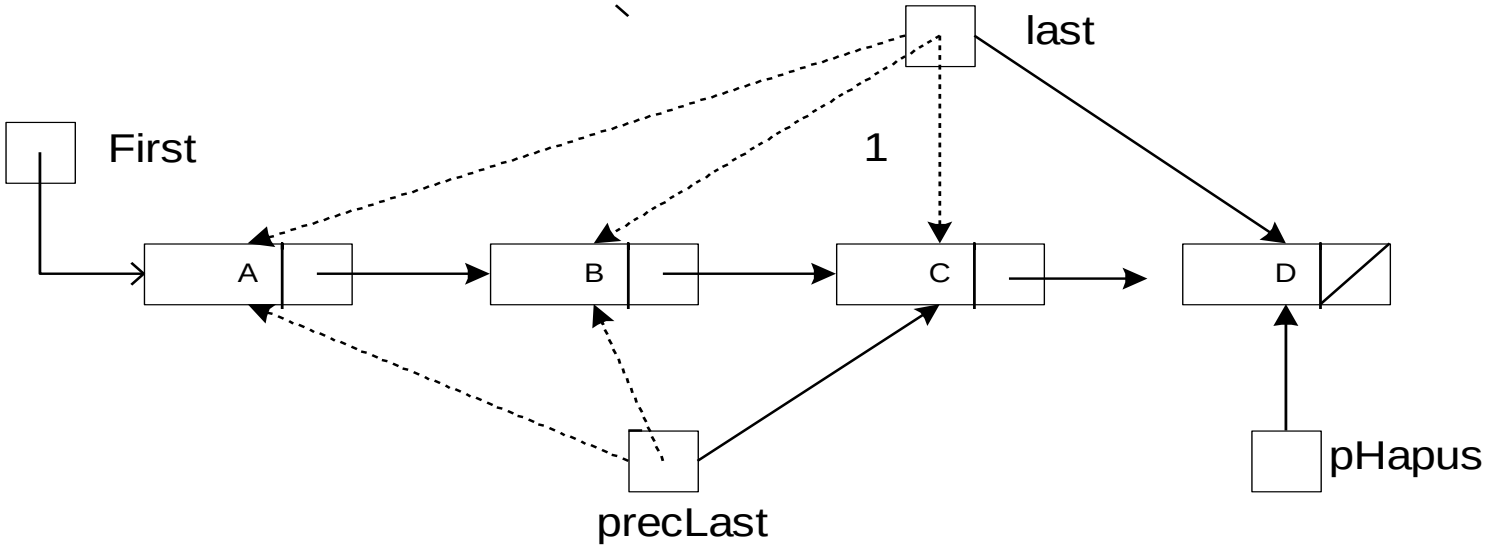
```
precLast=NULL;
last = First;
while (last->next != NULL) {
        precLast=last;
        last = last->next;
}
```

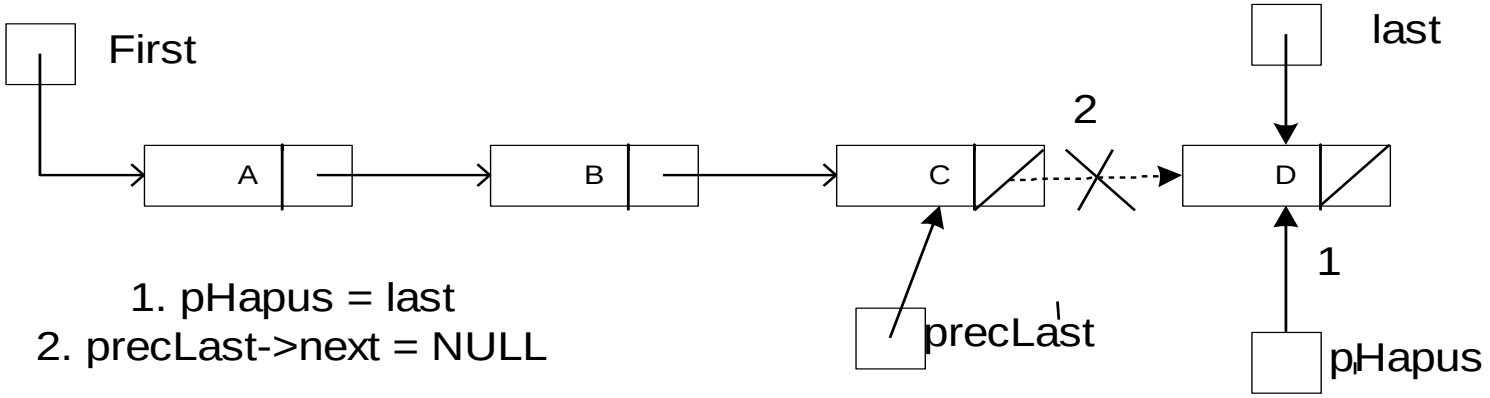
Delete Last pada Kasus Kosong

Delete Last pada Kasus Isi 1 Elemen



Delete Last pada Kasus Isi > 1 Elemen





```
void deleteLast(List& First, pointer& pHapus){
 // I.S List First mungkin kosong
 // F.S. List berkurang satu di belakang, yang dihapus
 //      dikembalikan

   pointer last,precLast;
   cout<<"Delete Last"<<endl;
   if (First==NULL){                    // kosong
      pHapus=NULL;
      cout<<"List kosong, tidak ada yang dihapus"<<endl;
   }
   else if (First->next==NULL){         //isi 1 elemen
      pHapus=First;
      First=NULL;                       // list jadi kosong
   }
   else {                               // isi > 1 elemen
      last=First;                       // menemukan elemen terakhir
      precLast=NULL;
      while (last->next!=NULL){
         precLast=last;          // preclast mencatat yg akan ditinggalkan Last
         last=last->next;        // last berpindah
      }
       pHapus=last;
      precLast->next=NULL;
    }

}
```