ILK 2109
Struktur Data

LIST LINIER

Dr. Ir. Elviawaty Muisa Zamzami, MT, MM

P.S. S1 Ilmu Komputer
Fasilkom-TI USU
2023

Bahasan

- Pengertian List Linier
- Karakteristik List Linier
- Ilustrasi List Linier
- Elemen List
- Representasi
- Alokasi Memori
- Operasi Dasar (Primitif)
- Traversal
- Create
- Insert
- Referensi

Pengertian List Linier

List linier (*linked list*) adalah sekumpulan elemen bertype sama yang mempunyai keterurutan tertentu.

Menggunakan alokasi memori dinamis, mengalokasikan memori untuk elemen-elemen list baru sesuai kebutuhan.

Elemen pertama list diacu dari alamat elemen pertama yang dimuat pada 'First'.

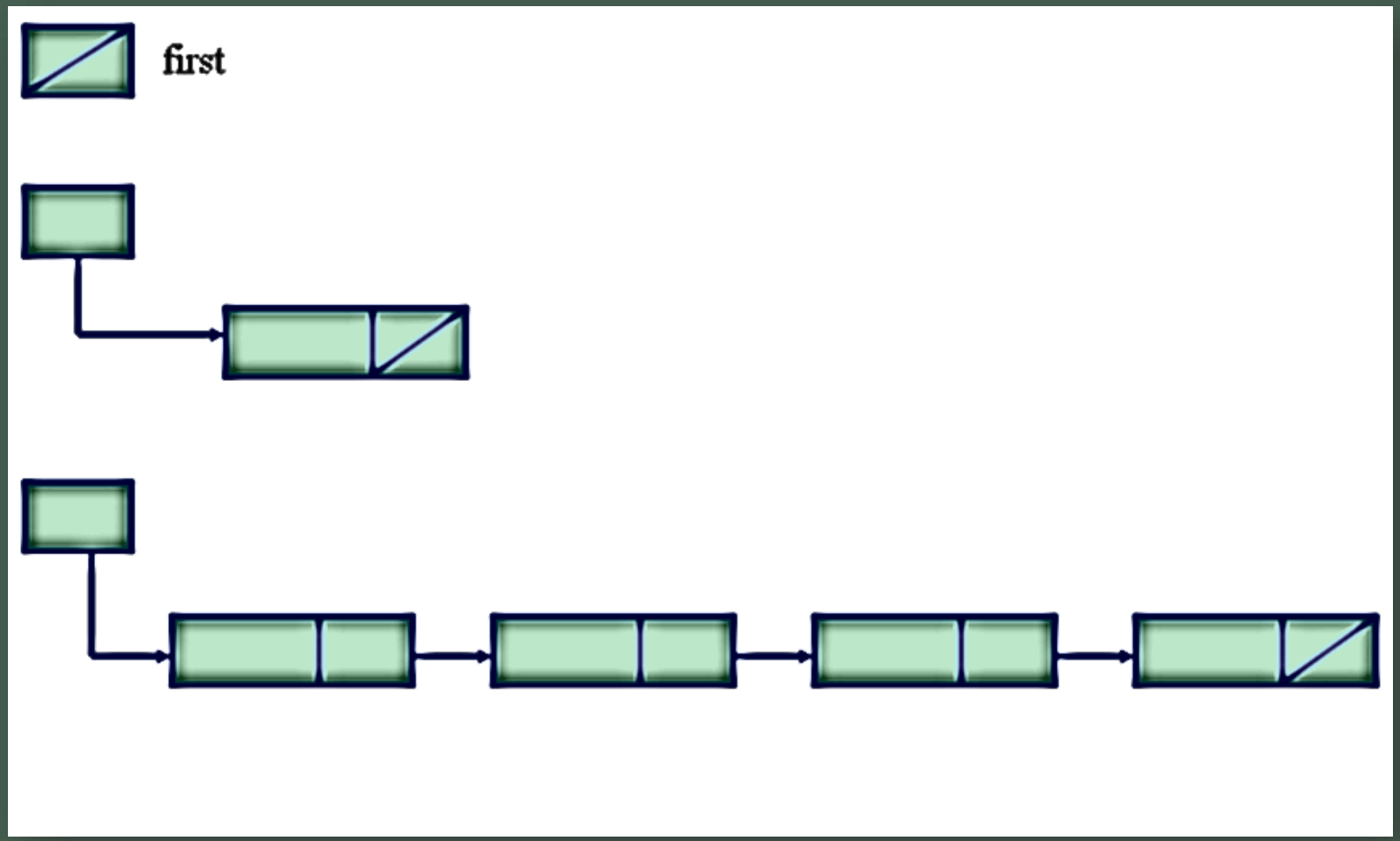
Alamat elemen berikutnya (suksesor) diakses melalui 'Next'.

Karakteristik List Linier

Jika L adalah list, dan P adalah address, maka:

Alamat elemen pertama list L diacu dengan First(L).	Elemen yang diacu oleh P dapat dikonsultasi informasinya dengan Info(P) dan Next(P).	List dikatakan kosong jika First(L) = nil.	Elemen terakhir pada list dikenali, salah satu cara mengenalinya adalah dengan Next(Last) = Nil.

Ilustrasi List Linier



Elemen List

Setiap elemen dibagi atas dua (2) bagian utama , yaitu bagian info dan bagian next.

Info	Next
Berisi data/informasi, sedangkan bagian	Berisi alamat elemen berikutnya.

Deklarasi elemen list:

Type ElmtList: < Info: InfoType, Next: Address >

Representasi Dengan Pointer (Pascal)

```
Type
   Infotype = integer;
   Address  = ^Elmtlist;
   Elmtlist = record
              Info : Infotype;
              Next : Address;
            End;
Var
   First : Address;
   P     : Address;
   {Akses:
       Info(P) : P^.Info
       Next(P) : P^.Next
   }
```

Representasi Dengan Tabel Berkait (Pascal)

```
Cons
    Nmax = 100;
    Nul  = 0;
Type
    Infotype = integer;
    Address  = [Nul..Nmax];
    Elmtlist = record
                Info : Infotype;
                Next : Address;
               End;
    TabElmtlist = array[1..Nmax] of Elmtlist;
Var
    First : Address;
    P     : Address;
    {Akses:
         Info(P) : TabElmtlist[P].Info
         Next(P) : TabElmtlist[P].Next
    }
```

Alokasi Memori

Alamat yang akan digunakan, akan terdefinisi jika sudah 'dialokasi'.

Dialokasi artinya sudah terdefinisi pada ruang memori.

Membuat suatu alamat menjadi terdefinisi dapat dilakukan dengan dua cara:

- Alamat (ruang memori untuk menyimpan nilai) sudah terdefinisi (dialokasi) sejak dideklarasikan).
- Pada saat dibutuhkan ketika program berjalan (run time).

Operasi Dasar (Primitif)

Traversal

Create

Conjugate

Insert

(Insert first, Insert after, Insert last)

Delete

(Delete first, Delete after, Delete last)

Traversal

Operasi traversal adalah operasi yang dilakukan untuk 'mengunjungi' setiap elemen list linier.

```
Procedure skema_traversal(Input L:List)
{I.S.:L adalah sebuah list terdefinisi,mungkin kosong    }
{F.S.:Semua elemen list L 'dikunjungi' dan telah diproses}
{     dengan MARK dan pemrosesan khusus pada list kosong }
```

```
Kamus Lokal

     P : address {address untuk traversal}
     Procedure Proses(input P:address)
     Procedure Inisialisasi
     Procedure Terminasi
```

```
Algoritma

     If (First(L)= nil)then
          Output('list kosong')
     Else {list tidak kosong}
          Inisialisasi
          P ← First(L)
          Repeat
               Proses(P)
               P ← next(P)
          Until (P = nil)
          Terminasi
```

Create

Operasi create dimaksudkan menciptakan sebuah elemen list linier dengan alamat yang terdefinisi.

I.S. : First tidak terdefinisi

F.S.: First terdefinisi, First = nil

first

```
Procedure Create(output First:address)
{I.S.: List tidak terdefinisi}
{F.S.: List terdefinisi,First=nil}

Kamus Lokal

Algoritma

        First ← nil
```

Create (Pascal)

```
Procedure Create(var First:address);
{I.S.: List tidak terdefinisi}
{F.S.: List terdefinisi, First=nil}
Begin

    First:=nil;

End;
```

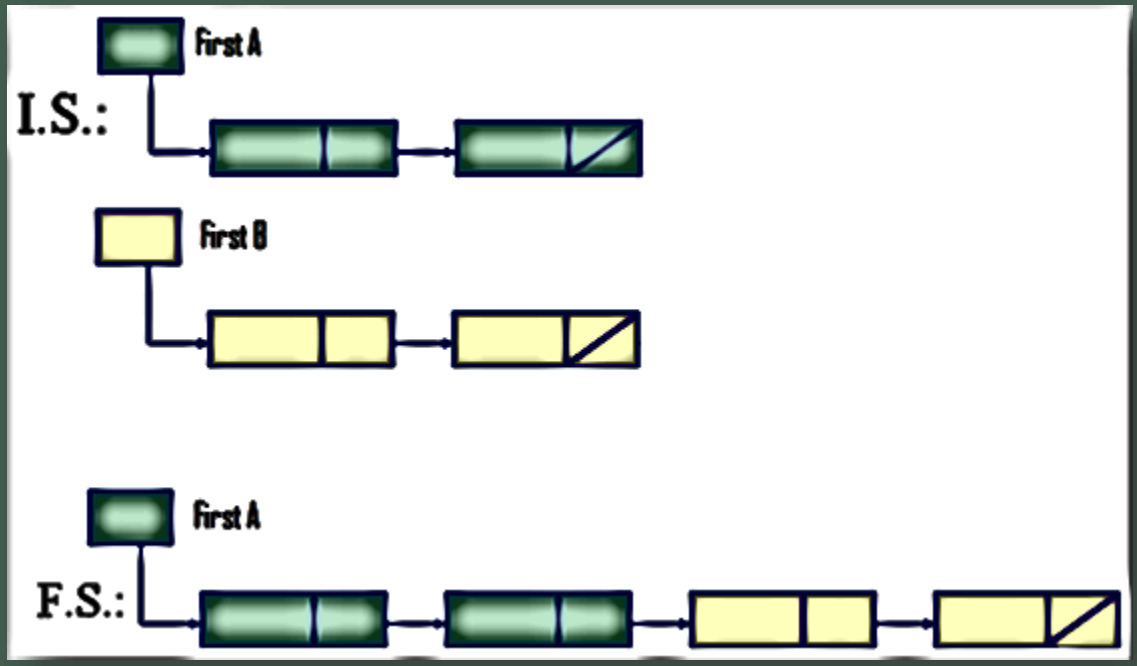
Pada Pascal terdapat perintah 'new' dan 'dispose'.

Perintah New(P) untuk mengalokasi sebuah alamat P.	Perintah Dispose(P) untuk dealokasi sebuah alamat P yang digunakan.

Conjugate

Operasi *conjugate* (konjugasi) digunakan untuk menggabungkan beberapa list. Misalkan terdapat dua (2) buah list linier A dan B.

Contoh, dilakukan konjugasi terhadap A dan B, dimana B 'digabungkan' setelah A.



Insert

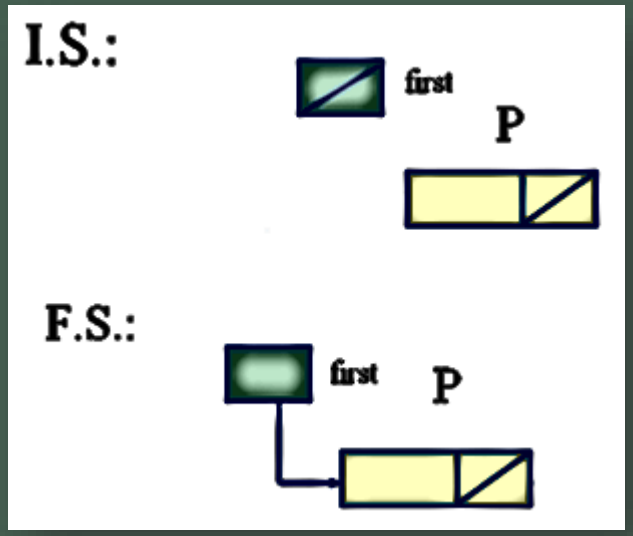
Operasi insert bertujuan untuk menyisipkan sebuah elemen pada list linier.

Penyisipan tersebut dapat dilakukan pada:

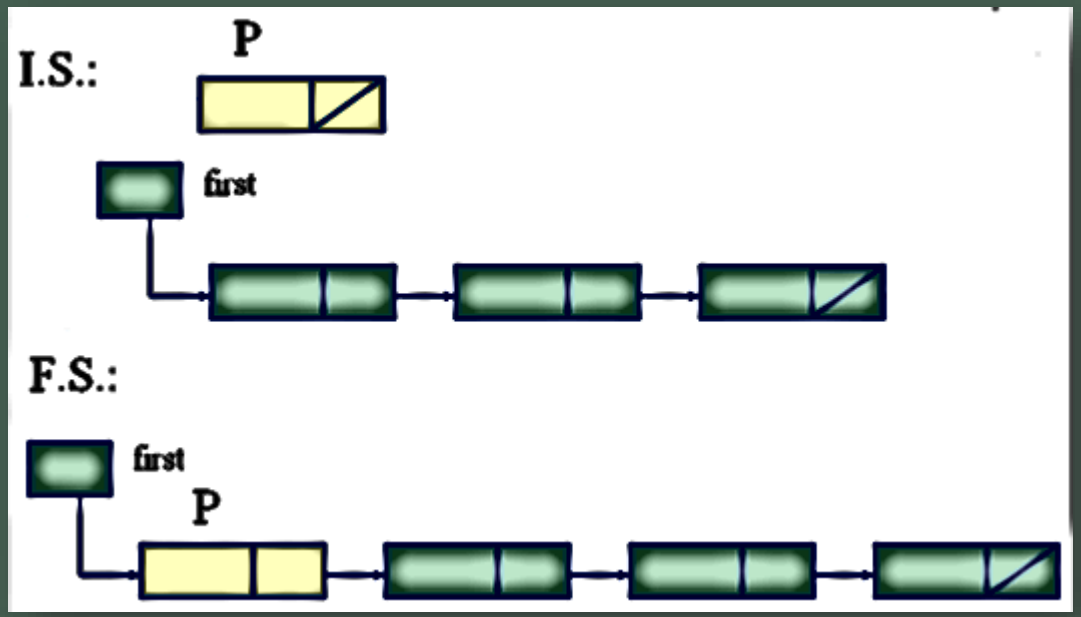
- Awal list (insert first).
- Setelah elemen tertentu (insert after).
- Elemen terakhir list (insert last).

Insert First (1)

Insert First pada List Kosong



Insert First pada List Tidak Kosong



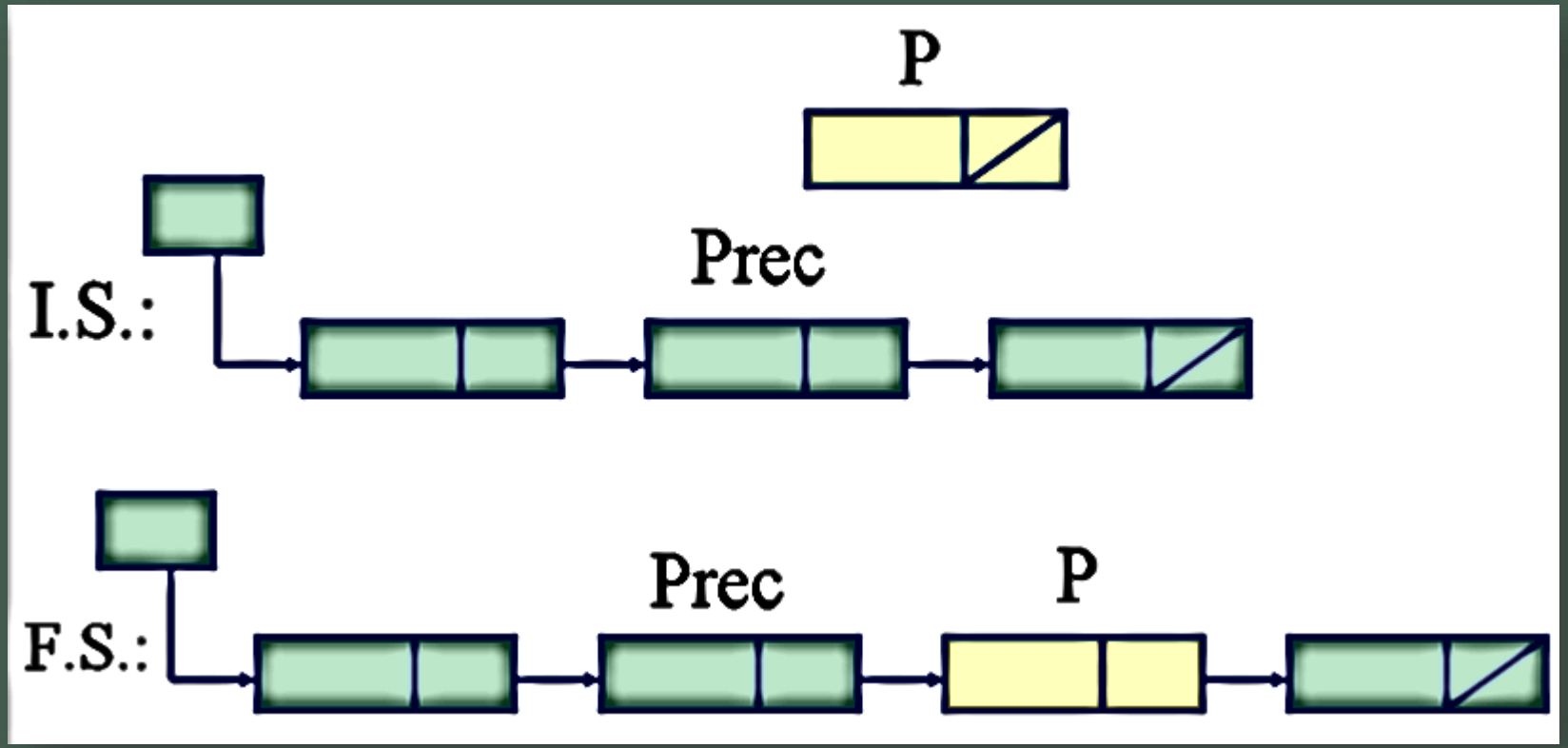
Insert First (2)

```
Procedure insert_first(Input/output L:List;input:P:address)
{I.S.:L adalah sebuah list terdefinisi,mungkin kosong}
{      P adalah sebuah address,next(P)=nil             }
{F.S.:P disisip menjadi elemen pertama pada list       }

Kamus Lokal

Algoritma

     If (First(L)= nil)then
          {list kosong}
          First(L) ← P
     Else {list tidak kosong}
          Next(P) ← First(L)
          First(L) ← P
```

```
Procedure insert_first(var First:address; P:address);
{I.S.:First adalah sebuah alamat awal list,mungkin kosong}
{      P adalah sebuah address,next(P)=nil                 }
{F.S.:P disisip menjadi elemen pertama pada list           }

Begin
     If (First = nil)then
     Begin
          {list kosong}
          First := P;
     End
     Else
     Begin
          {list tidak kosong}
          P^.Next := First;
          First := P;
     End;
End;
```

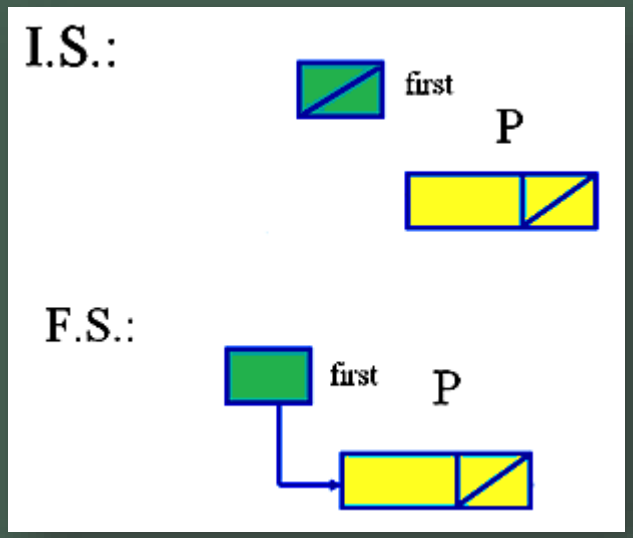
Insert After

Menyisip sebuah elemen setelah elemen tertentu (mis, Prec).

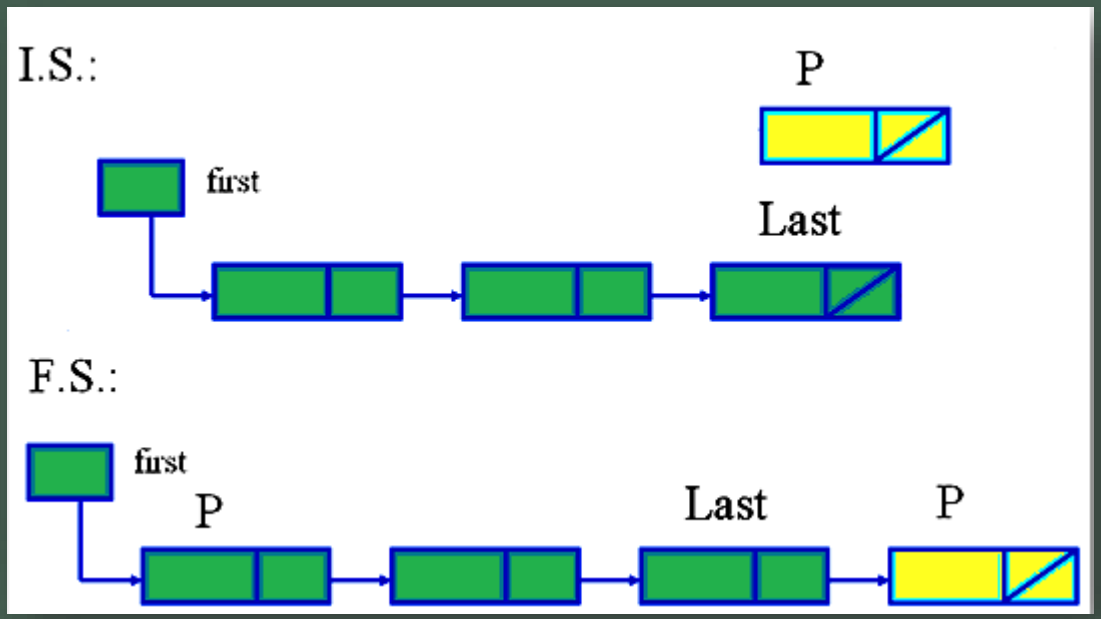


Insert Last

Insert Last pada List Kosong



Insert Last pada List Tidak Kosong



Delete

Operasi delete digunakan untuk menghapus sebuah elemen dari list linier.

Untuk menghapus sebuah elemen harus dipastikan list tidak kosong.

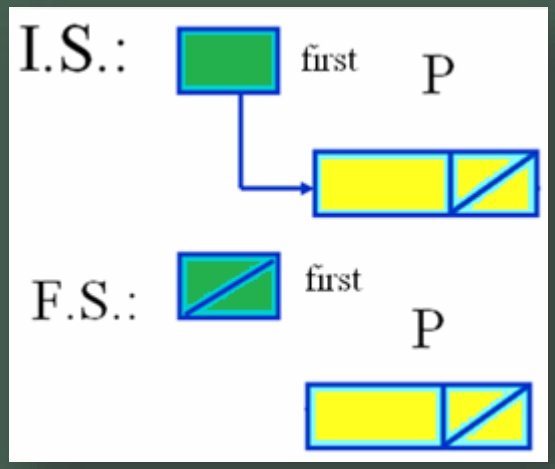
Juga jangan sampai 'kehilangan' alamat yang disimpan pada elemen tersebut.

Operasi delete dibedakan atas :

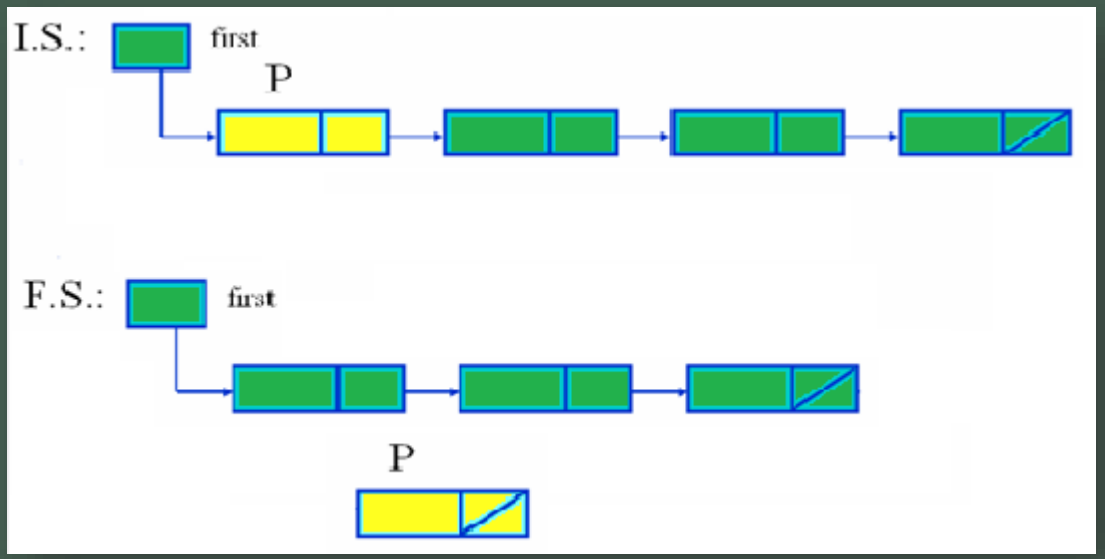
- Awal list (delete first).
- Setelah elemen tertentu (delete after).
- Elemen terakhir list (delete last).

Delete First

Delete First pada List dengan 1 Elemen

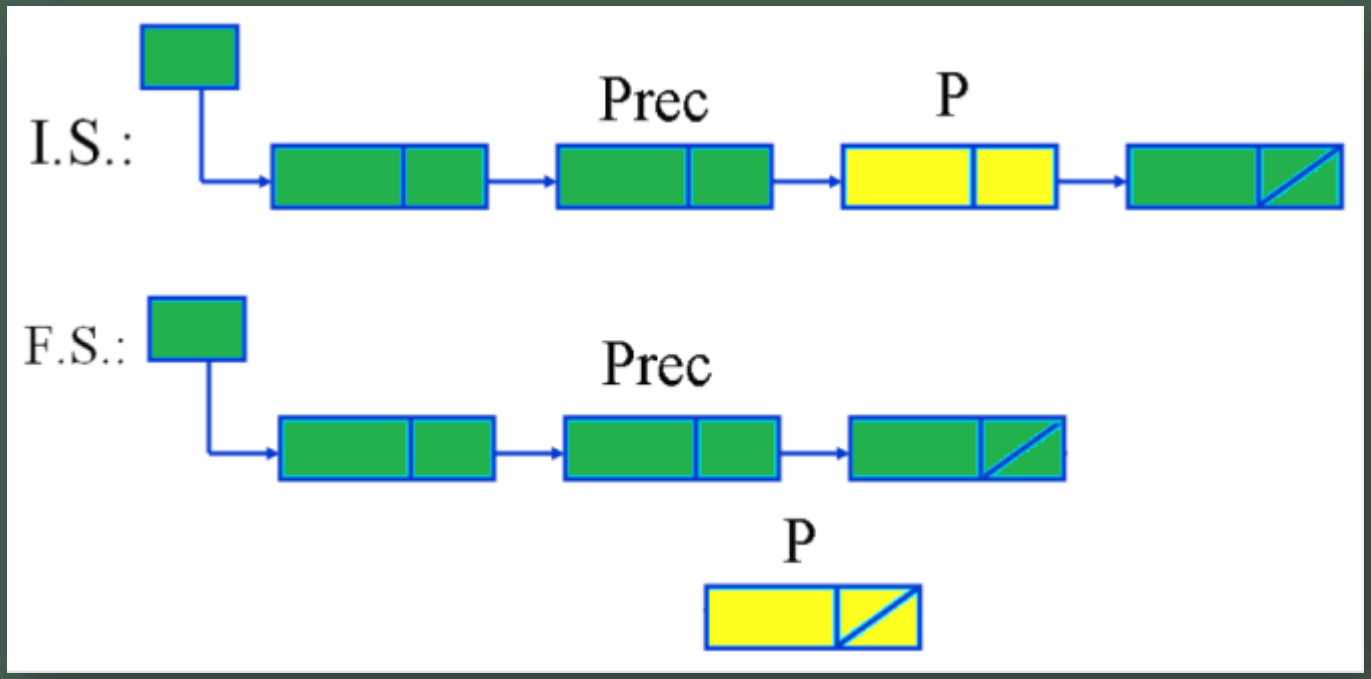


Delete First pada List dengan lebih dari 1 Elemen



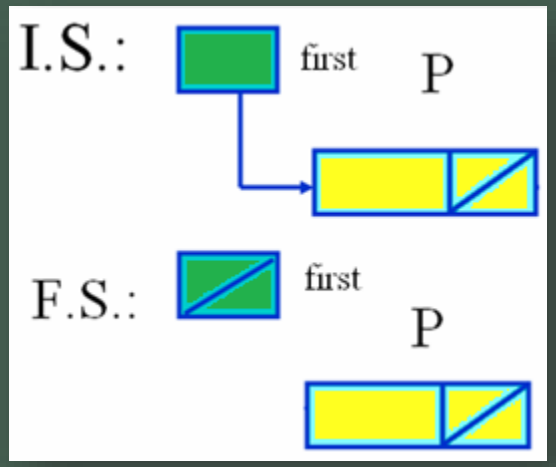
Delete After

Menghapus sebuah elemen setelah elemen tertentu (mis, Prec).

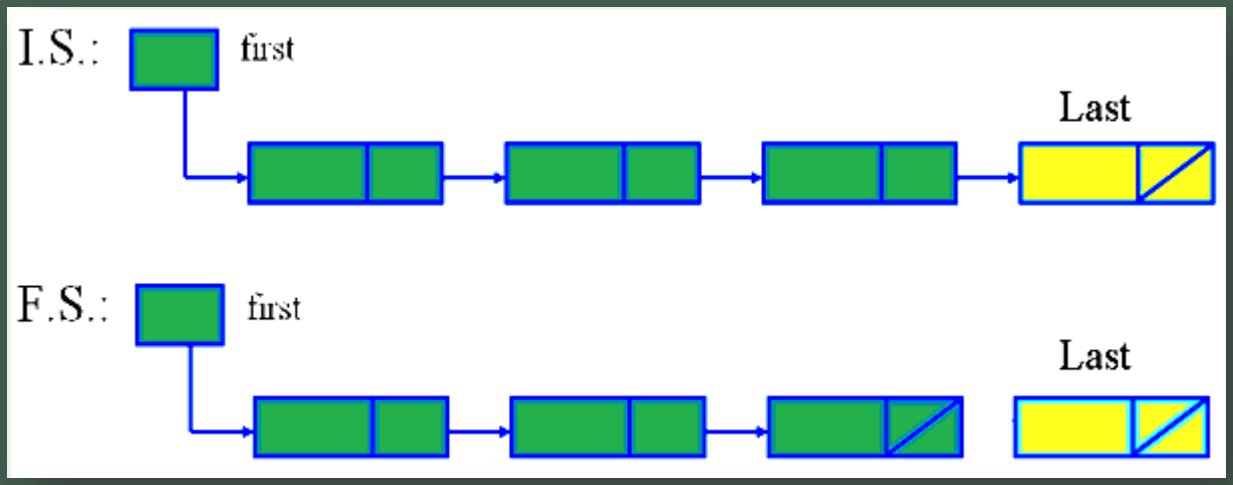


Delete Last (1)

Delete Last pada List dengan 1 Elemen



Delete Last pada List dengan lebih dari 1 Elemen



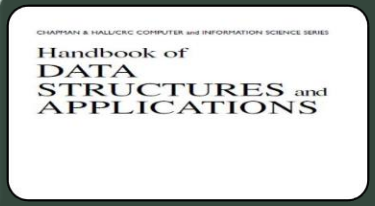
Delete Last (2)

```
Procedure DeleteLast(Input First:List, output P:address)
{I.S : List L tidak kosong, minimal terdiri dari 1 elemen}
{F.S : Menghapus elemen terakhir dari list, list mungkin}
{      menjadi kosong, P adalah alamat elemen terakhir}
{      dari list sebelum penghapusan}
```

```
Kamus Lokal

     Last, Preclast : address
```

```
Algoritma

     {cari Last dan Preclast}
     Last ← First(L)
     Preclast ← Nil
     While (next(Last) ≠ Nil) do
          Preclast ← Last
          Last ← next(Last)
     {next(Last)=nil}
     P ← Last
     If (Preclast = nil) then
          {list 1 elemen, menjadi kosong}
          First(L) ← Nil
     Else {list > 1 elemen}
          Next(Preclast) ← Nil
```

Referensi



Mehta,Dinesh P., and Sahni,Sartaj, *Handbook of Data Structures and Applications*, Chapman & Hall/CRC, 2005.

Liem,Inggriani, *Diktat Kuliah Algoritma dan Pemrograman Prosedural*, ITB, Bandung, 1997.

Shaffer,Clifford A., *A Practical Introduction to Data Structures and Algorithm Analysis*, 3rd Edition, 2010.

Bandyopadhyay,Samir K., and Dey,Kashi N., *Data Structures Using C*, Pearson Education India, 2008.