

Implementasi ADT List dengan Struktur Berkait

IF1210 – Algoritma dan Pemrograman 1
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Kembali ke definisi List

List, dikenal juga dengan *sequence*, merupakan sekumpulan elemen **bertipe sama** yang memiliki suatu **keterurutan tertentu** (*ordered*, tidak harus *sorted*).

Operasi-operasi:

- isEmpty
- indexOf
- length
- akses (getElmt, setElmt)
- concat
- insert-
- delete-
- pola traversal

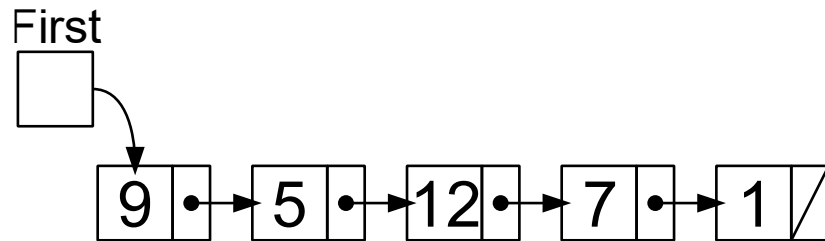
Implementasi List dengan struktur berkait

Elemen-elemen direpresentasikan dengan **Node** $\langle \text{Info}, \text{Next} \rangle$ yang saling berkait.

List diacu melalui **Address** elemen pertamanya (**first**).

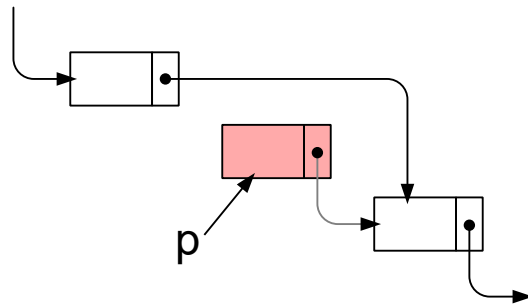
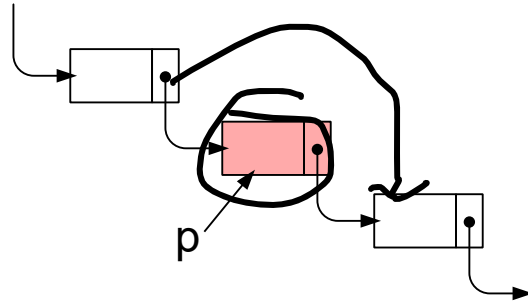
Alamat elemen berikutnya (suksesor) diakses dengan **next**.

Elemen terakhir ditandai dengan Next menunjuk ke **NIL**.

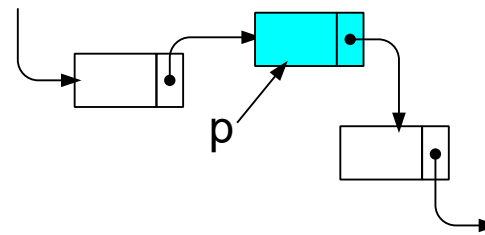
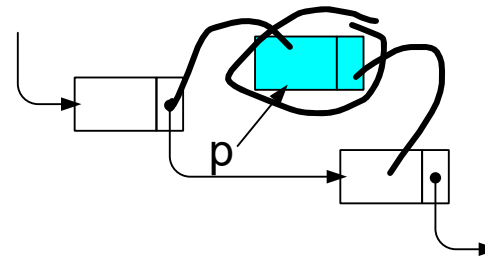


Karakteristik list linier

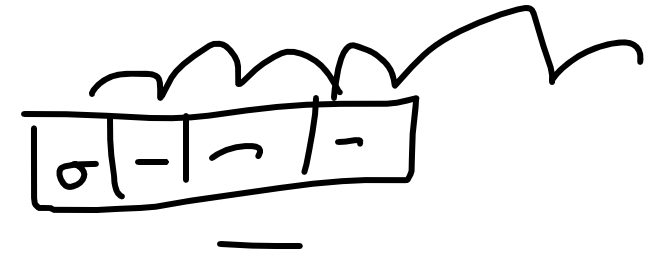
Penambahan & penghapusan elemen sangat sederhana.



penghapusan elemen



penambahan elemen



Tidak efisien untuk mengakses elemen melalui indeksnya.
(Harus menelusuri mulai dari *node* pertama sambil mencacah.)

Implementasi List dengan struktur berkait

Jika l adalah List, dan p adalah Address:

- Karena l diacu melalui alamat node pertamanya, maka $\text{first} = l$.
- Elemen yang diacu oleh p dapat diakses informasinya dengan notasi:
 - $p \uparrow . \text{info}$: nilai yang disimpan
 - $p \uparrow . \text{next}$: alamat elemen berikutnya

Definisi List Kosong

- List l adalah list kosong: $l = \mathbf{NIL}$

Definisi Elemen terakhir

- $\text{last} \uparrow . \text{next} = \mathbf{NIL}$, dengan last adalah alamat elemen terakhir

Dalam notasi algoritmik

```
{ Deklarasi tipe }
```

```
type ElType: integer
type Address: pointer to Node
type Node: < info: ElType,
            next: Address >
type List: Address
```

```
{ Deklarasi variabel }
```

```
l: List
p1: Address
p2: Address
```

```
{ Inisialisasi List }
```

```
CreateList(l)
```

```
{ Akses node pertama: }
```

```
p1 ← l
```

```
{ Cetak isi p1 & akses elemen  
  setelah p1 }
```

```
output(p1↑.info)
```

```
p2 ← p1↑.next
```

Dalam bahasa C

```
/* Deklarasi tipe */
```

```
typedef int ElType;
typedef struct tNode* Address;
typedef struct tNode {
    ElType info;
    Address next; } Node;
typedef Address List;
```

```
/* Deklarasi variabel */
```

```
List l;
Address p1;
Address p2;
```

```
/* Inisialisasi List */
```

```
CreateList(*l)
```

```
/* Akses node pertama: */
```

```
p1 = l
```

```
/* Cetak isi p1 & akses elemen
   setelah p1 */
```

```
printf("%d\n", p1->info);
p2 = p1->next;
```