Variasi List Linier

List dengan Pointer Ganda

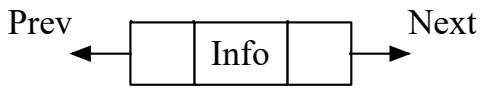
IF2110 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

List dengan Pointer Ganda

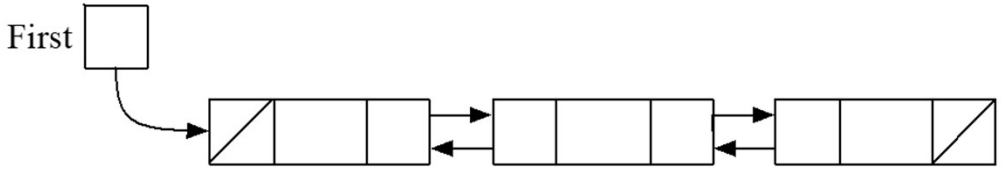
Elemen pertama: First(L)

Elemen terakhir: Next(P) = Nil

List kosong: First(L) = Nil

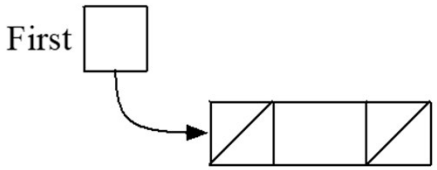


Elemen list dengan pointer ganda



List dengan tiga elemen

List kosong



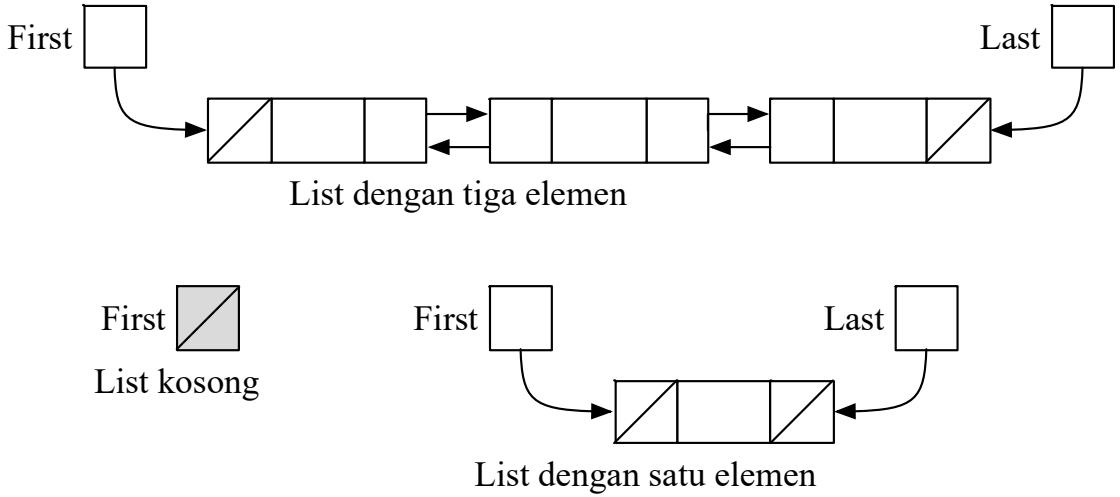
List dengan satu elemen

List dengan Pointer Ganda + Pencatatan Last

Elemen pertama: First(L)

Elemen terakhir: Last(L); Next(Last(L)) = Nil

List kosong: First(L) = Last (L) = Nil



Beberapa catatan

Dibutuhkan jika harus dilakukan banyak operasi terhadap elemen suksesor dan juga predesesor.

> Dengan tersedianya alamat predesesor pada setiap elemen list, maka memorisasi Prec pada beberapa algoritma yang pernah ditulis dapat dihindari

Operasi dasar menjadi sangat “banyak”

Memori yang dibutuhkan membesar

Jika list lojik semacam ini direpresentasi secara kontigu dengan tabel, maka sangat menguntungkan karena memorisasi Prev dan Next dilakukan dengan kalkulasi

Bahasan

Buatlah ADT list dengan elemen berpointer ganda dilengkapi driver:

- Representasi fisik: berkait dengan pointer
- Penunjuk First dan Last

Rep. Fisik dengan Pointer

```
#define NIL NULL
typedef int ElType;
typedef struct node *Address;
typedef struct node {
    ElType info;
    Address prev;
    Address next;
} Node;
/* Definisi list: */
/* List kosong: First = Nil and Last = Nil */
/* Setiap elemen dengan address P dapat diacu Info(P), Prev(P), Next(P) */
typedef struct {
    Address first; Address last;
} List;
/* Selektor */
#define  INFO(p) (p)->info
#define  PREV(p) (p)->prev
#define  NEXT(p) (p)->next
#define FIRST(l) ((l).first)
#define  LAST(l) ((l).last)
```

Beberapa primitif

Buatlah sebagai latihan:

```
void insertFirst(List *l, ElType x);
/* I.S. List l terdefinisi */
/* F.S. Menambahkan elemen baru x sebagai elemen pertama */

void insertLast(List *l, ElType x);
/* I.S. List l terdefinisi */
/* F.S. x ditambahkan sebagai elemen terakhir l yang baru */
```

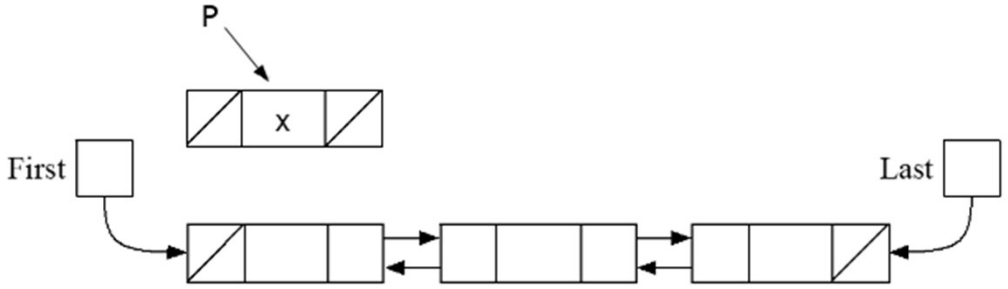
Beberapa primitif

Buatlah sebagai latihan:

```
void deleteFirst(List *l, ElType *x);
/* I.S. List l tidak kosong */
/* F.S. x adalah elemen pertama list l sebelum penghapusan */
/*      Elemen list berkurang satu (mungkin menjadi kosong) */
/*      First element yg baru adalah suksesor elemen pertama yang lama */

void deleteLast(List *l, ElType *x) {
/* I.S. List l tidak kosong */
/* F.S. x adalah elemen terakhir list l sebelum penghapusan */
/*      Elemen list berkurang satu (mungkin menjadi kosong) */
/*      Last element baru adalah predesesor elemen pertama yg lama, jika ada */
```

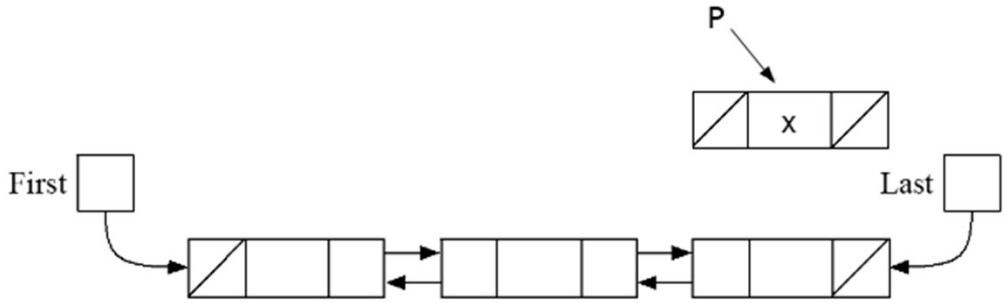
Beberapa primitif

```
void insertFirst(List *l, ElType x){
/* I.S. List l terdefinisi */
/* F.S. Menambahkan elemen baru x sebagai elemen pertama */
    /* Kamus Lokal */
    Address p;

    /* Algoritma */
    p = newNode(x);
    if (p != NIL) {
        NEXT(p) = FIRST(*l);
        if (!isEmpty(*l)) {
            PREV(FIRST(*l)) = p;
        } else /* l kosong */ {
            LAST(*l) = p;
        }
        FIRST(*l) = p;
    }
}
```



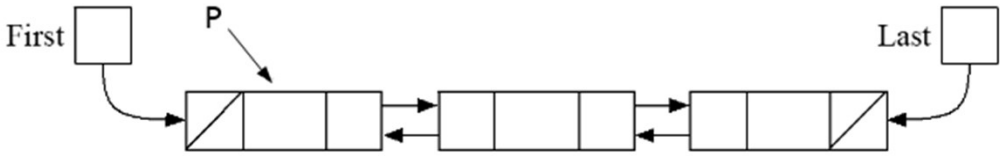
Beberapa primitif

```
void insertLast(List *l, ElType x){
/* I.S. List l terdefinisi */
/* F.S. x ditambahkan sebagai elemen terakhir l yang baru */
    /* Kamus Lokal */
    Address p;

    /* Algoritma */
    p = newNode(x);
    if (p != NIL) {
        PREV(p) = LAST(*l);
        if (!isEmpty(*l)) {
            NEXT(LAST(*l)) = p;
        } else /* l kosong */ {
            FIRST(*l) = p;
        }
        LAST(*l) = p;
    }
}
```



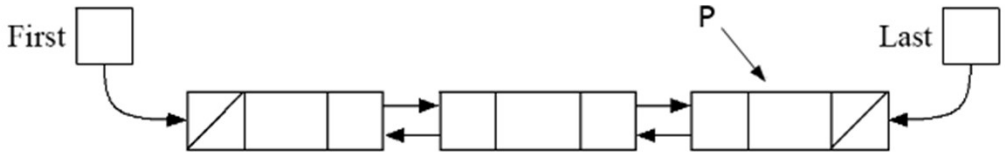
Beberapa primitif

```
void deleteFirst(List *l, ElType *x);
/* I.S. List l tidak kosong */
/* F.S. x adalah elemen pertama list l sebelum penghapusan */
/*      Elemen list berkurang satu (mungkin menjadi kosong) */
/*      First element yg baru adalah suksesor elemen pertama yang lama */
    /* Kamus Lokal */
    Address p;

    /* Algoritma */
    p = FIRST(*l);
    *x = INFO(p);
    if (FIRST(*l) == LAST(*l)) { /* l hanya 1 elemen */
        LAST(*l) = NIL;
    } else { /* l > 1 elemen */
        PREV(NEXT(FIRST(*l))) = NIL;
    }
    FIRST(*l) = NEXT(FIRST(*l));
    free(P);
}
```



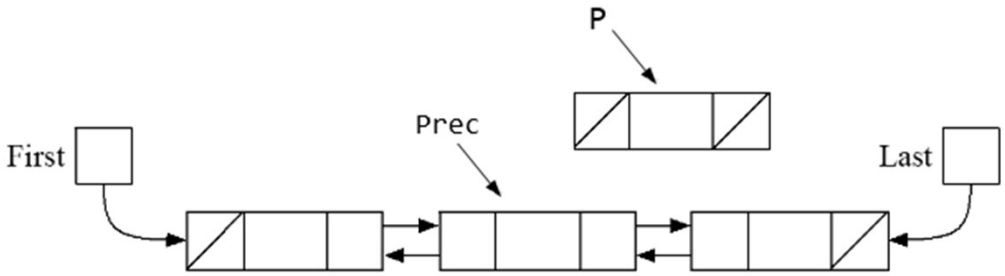
Beberapa primitif

```
void deleteLast(List *l, ElType *x) {
/* I.S. List l tidak kosong */
/* F.S. x adalah elemen terakhir list l sebelum penghapusan */
/*      Elemen list berkurang satu (mungkin menjadi kosong) */
/*      Last element baru adalah predesesor elemen pertama yg lama, jika ada */
    /* Kamus Lokal */
    Address p;

    /* Algoritma */
    p = LAST(*l);
    *x = INFO(p);
    if (FIRST(*l) == LAST(*l)) { /* l hanya 1 elemen */
        FIRST(*l) = NIL;
    } else { /* L > 1 elemen */
        NEXT(PREV(LAST(*l))) = NIL;
    }
    LAST(*l) = PREV(LAST(*l));
    free(p);
}
```



Beberapa primitif

```
void InsertAfter(List *L, address P, address Prec) {
/* I.S. Prec pastilah elemen list; P sudah dialokasi */
/* F.S. Insert P sebagai elemen sesudah elemen beralamat Prec */
    /* Kamus Lokal */

    /* Algoritma */
    if (Next(Prec) != Nil) { /* Prec bukan elemen terakhir */
        Prev(Next(Prec)) = P;
    } else { /* Prec elemen terakhir */
        Last(*L) = P;
    }
    Next(P) = Next(Prec);
    Prev(P) = Prec;
    Next(Prec) = P;
}
```



Beberapa primitif

```
void DeleteAfter(List *L, address *Pdel, address Prec) {
/* I.S. List tidak kosong. Prec adalah anggota list.
        Next(Prec)!=Nil */
/* F.S. Menghapus Next(Prec): */
/*      Pdel adalah alamat elemen list yang dihapus */
    /* Kamus Lokal */

    /* Algoritma */
    *Pdel = Next(Prec);
    if (Next(Next(Prec))!=Nil) { /* Prec tidak jadi elemen terakhir */
        Prev(Next(Prec)) = Prec;
    } else { /* Prec jadi elemen terakhir */
        Last(*L) = Prec;
    }
    Next(Prec) = Next(Next(Prec));
    Next(*Pdel) = Nil;
    Prev(*Pdel) = Nil;
}
```