Variasi List Linier

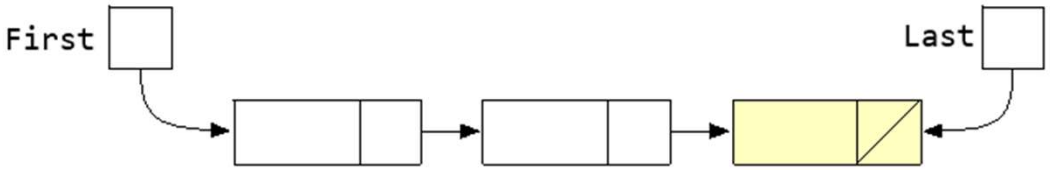
List Linier dengan Dummy Element

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

Bahasan

Buatlah ADT list + driver dengan elemen dummy di akhir:

- Penunjuk first dan last (last element menunjuk ke elemen dummy)
- Alamat dummy: tetap
- Representasi fisik: berkait dengan pointer



Representasi Fisik dengan Pointer

```
#define NIL NULL
#define IDX_UNDEF -1
typedef int ElType;
typedef struct node* Address;
typedef struct node { ElType info; Address next; } Node;

/* Definisi list: */
/* List kosong: First(L) = Last(L) = dummy@ */
/* Setiap elemen dengan address P dapat diacu Info(P), Next(P) */
/* Elemen dummy terletak pada last */
typedef struct {
    Address first;
    Address last;
} List;

/* Selektor */
#define INFO(P)  (P)->info
#define NEXT(P)  (P)->next
#define FIRST(L) ((L).first)
#define LAST(L)  ((L).last)
```

Beberapa primitif

Buatlah sebagai latihan:

```
/* PROTOTYPE */
/****************** TEST LIST KOSONG ******************/
boolean isEmpty(List l);
/* Mengirim true jika list kosong: FIRST(l) = dummy@
   dan LAST(L) = dummy@ */
/****************** PEMBUATAN LIST KOSONG ******************/
void CreateList(List *l);
/* I.S. sembarang */
/* F.S. Terbentuk list l kosong, dengan satu elemen dummy */
/*      Jika gagal maka FIRST(l) = LAST(l) = NIL dan list gagal terbentuk */
/****************** SEARCHING ******************/
int indexOf(List l, ElType x);
/* Mengembalikan indeks node list dengan nilai X, atau IDX_UNDEF jika tidak ada */
```

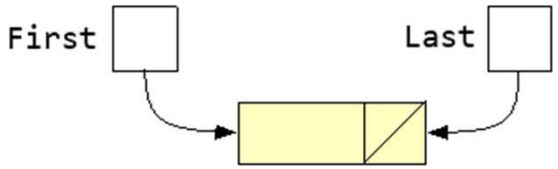
Beberapa primitif

Buatlah sebagai latihan:

```
void insertFirst(List *l, ElType x);
/* I.S. List l terdefinisi */
/* F.S. Menambahkan elemen x sebagai elemen pertama List l */
void insertLast(List *l, ElType x);
/* I.S. List l terdefinisi */
/* F.S. x ditambahkan sebagai elemen terakhir l yang baru, */
/*      yaitu menjadi elemen sebelum elemen dummy */
void deleteFirst(List *l, ElType *x);
/* I.S. List l tidak kosong */
/* F.S. x adalah nilai 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 terakhir sebelum dummy pada list sebelum penghapusan */
/*      Elemen list berkurang satu (mungkin menjadi kosong) */
```

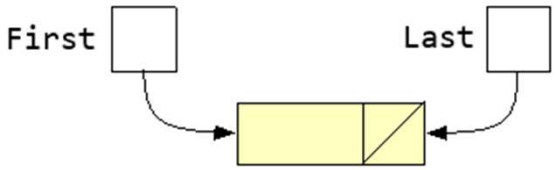
Beberapa primitif

```
boolean isEmpty(List l) {
/* Mengirim true jika list kosong */
    /* Kamus Lokal */

    /* Algoritma */
    return (FIRST(l) == LAST(l));
}
```



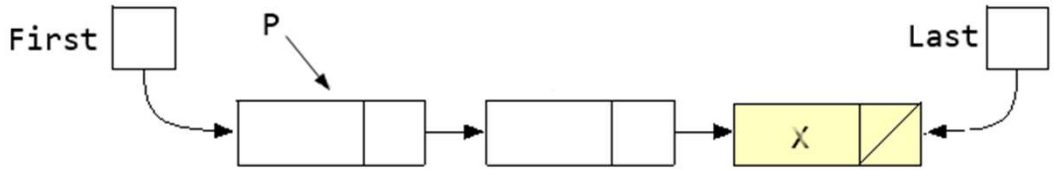
Beberapa primitif

```
void CreateList(List *l) {
/* I.S. Sembarang */
/* F.S. Terbentuk list l kosong, dengan satu elemen dummy,
        jika alokasi dummy berhasil */
/*      Jika gagal maka FIRST(l) = LAST(l) = NIL dan list gagal terbentuk */
    /* Kamus Lokal */
    Address pDummy;

    /* Algoritma */
    pDummy = newNode(0); /* INFO(pDummy) tidak didefinisikan */
    if (Pdummy != Nil) {
        FIRST(*L) = pDummy;
        LAST(*L) = pDummy;
    } else /* List gagal terbentuk */ {
        FIRST(*L) = NIL;
        LAST(*L) = NIL;
    }
}
```



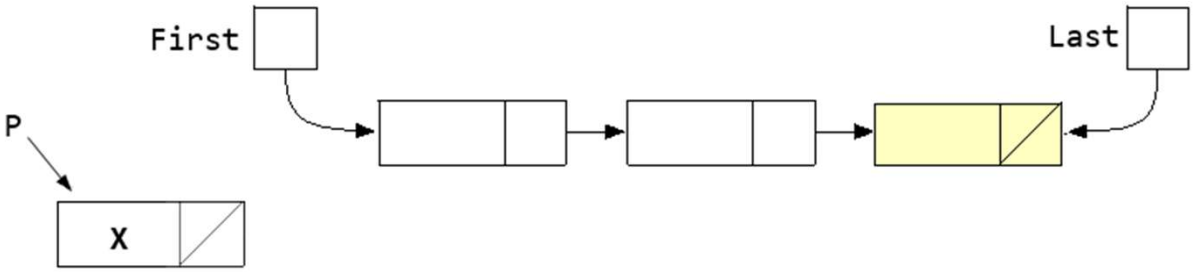
Beberapa primitif

```
int indexOf(List l, ElType x) {
/* Mengembalikan indeks node list dengan nilai X, atau IDX_UNDEF jika tidak ada */
    /* Kamus Lokal */
    Address p;
    int idx;
    /* Algoritma */
    INFO(LAST(l)) = x; /* letakkan X di sentinel */
    p = FIRST(l);
    idx = 0;
    while (INFO(p) != x) {
        p = NEXT(p);
        idx++;
    } /* INFO(p) = x */
    if (p != LAST(l)) { /* bukan ketemu di sentinel */
        return idx;
    } else /* p = LAST(l), ketemu di sentinel */ {
        return IDX_UNDEF;
    }
}
```



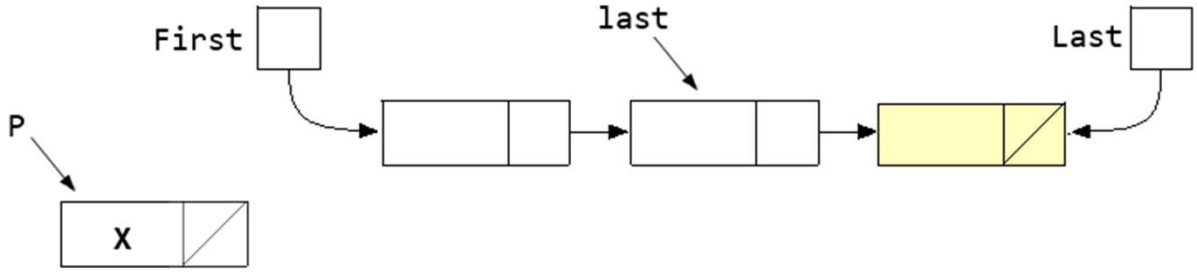
Beberapa primitif

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

    /* Algoritma */
    p = newNode(x);
    if (p != NIL) {
      NEXT(p) = FIRST(*l);
      FIRST(*l) = p;
    }
}
```



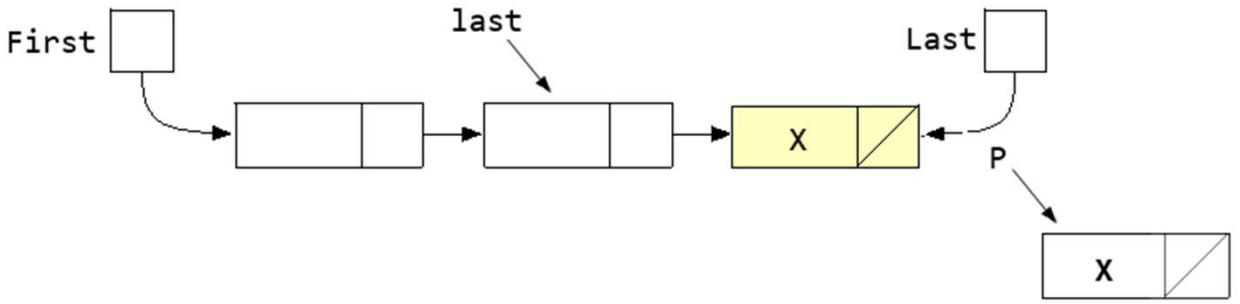
Beberapa primitif

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

    /* Algoritma */
    if (isEmpty(*l)) {
        insertFirst(l,x);
    } else {
        p = newNode(x);
        if (p != NIL) {
            last = FIRST(*l);
            while (NEXT(last) != LAST(*l)) {
                last = NEXT(last);
            } /* NEXT(last) == LAST(*l) alias dummy */
            NEXT(last) = p;
            NEXT(p) = LAST(*l);
        }
    }
}
```



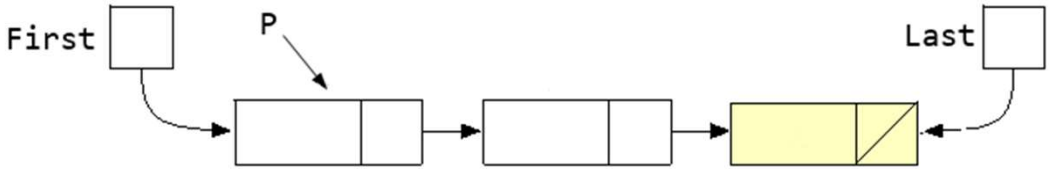
Beberapa primitif

```
void insertLast(List *l, ElType x) {
/* I.S. List l terdefinisi */
/* F.S. x ditambahkan sebagai elemen terakhir l yang baru */
/* Versi jika alamat elemen dummy boleh berubah */
    /* Kamus Lokal */

    /* Algoritma */
    if (isEmpty(*l)) {
        insertFirst(l,x);
    } else {
        INFO(LAST(*l)) = x;
        p = newNode(x); /* dummy baru */
        if (p != NIL) {
            NEXT(LAST(*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);
    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 terakhir yg lama, jika ada*/
    /* Kamus Lokal */
    Address last, precLast;

    /* Algoritma */
    last = FIRST(*l); precLast = NIL;
    while (NEXT(last) != LAST(*l)) {
        precLast = last; last = NEXT(last);
    }
    *x = INFO(last);
    if (precLast == NIL) { /* kasus satu elemen */
        FIRST(*l) = LAST(*l);
    } else {
        NEXT(precLast) = LAST(*l);
    }
    free(last);
}
```