Variasi List Linier
List Sirkuler

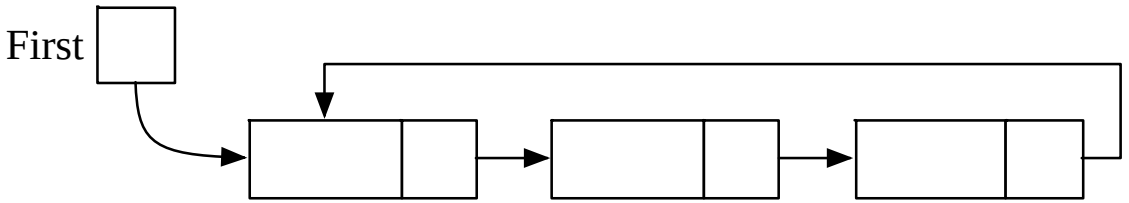
IF1210 – Algoritma dan Pemrograman 1
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

List Sirkuler

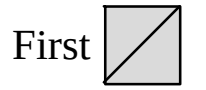
Elemen pertama: First(L)

Elemen terakhir: Last(L) = P; Next(P) = First(L)

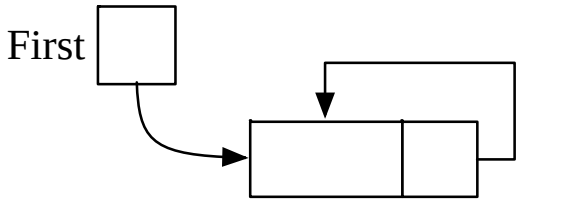
List kosong: First(L) = Nil



List sirkuler dengan tiga elemen



List kosong



List sirkuler dengan satu elemen

Beberapa catatan

List dengan representasi ini sebenarnya tidak mempunyai “First”

First adalah “Current Pointer”

Representasi ini dipakai jika dilakukan proses terus menerus terhadap anggota list (misalnya dalam round robin services pada sistem operasi)

Penambahan dan penghapusan pada elemen pertama akan berakibat harus melakukan traversal untuk mengubah Next dari elemen Last.

Bahasan - 3

Buatlah ADT list sirkuler + driver:

- Penunjuk First
- Representasi fisik: berkait dengan pointer

Rep. Fisik dengan Pointer

```
#define NIL NULL

typedef int ElType;
typedef struct node* Address;
typedef struct node { ElType info; address next; } Node;

/* Definisi list: */
/* List kosong: FIRST(l) = NIL */
/* Setiap elemen dengan address p dapat diacu INFO(p), NEXT(p) */
/* Elemen terakhir list: jika addressnya last,
   maka NEXT(last) = FIRST(l) */
typedef struct {
    address first;
} List;

/* Selektor */
#define  INFO(p)  (p)->info
#define  NEXT(p)  (p)->next
#define FIRST(l) ((l).first)
```

Beberapa primitif

Buatlah sebagai latihan:

```
boolean addrSearch(List l, Address p);
/* Mencari apakah ada elemen list l yang beralamat p */
/* Mengirimkan true jika ada, false jika tidak ada */
void insertFirst(List *l, ElType x);
/* I.S. List l terdefinisi */
/* F.S. Menambahkan elemen bernilai 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 */
```

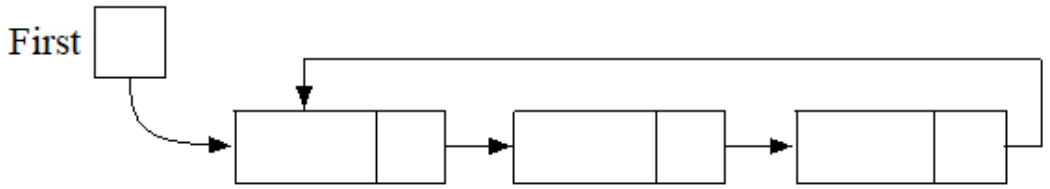
Beberapa primitif

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 sebelum penghapusan */
/*      Elemen list berkurang satu (mungkin menjadi kosong) */
/*      Last element baru adalah predesesor elemen pertama yg lama, jika ada */
void displayList(List l);
/* I.S. List l mungkin kosong */
/* F.S. Jika list tidak kosong, semua nilai (info) yg disimpan pada elemen list diprint */
/*      Jika list kosong, hanya menuliskan "list kosong" */
```

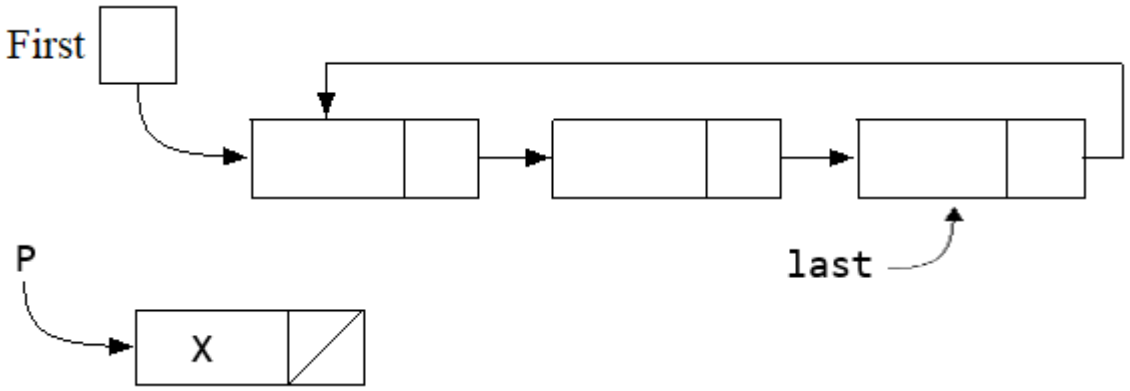
Beberapa primitif

```
boolean addrSearch(List l, Address p) {
/* Mencari apakah ada elemen list l yang beralamat p */
/* Mengirimkan true jika ada, false jika tidak ada */
  /* Kamus Lokal */
    Address pt;
  /* Algoritma */
    if (isEmpty(l)) {
        return false;
    } else {
        pt = FIRST(l);
        while ((NEXT(pt) != FIRST(l)) && (pt != p)) {
            pt = NEXT(pt);
        }
        /* NEXT(pt) = FIRST(l) or pt = p */
        return pt == p;
    }
}
```



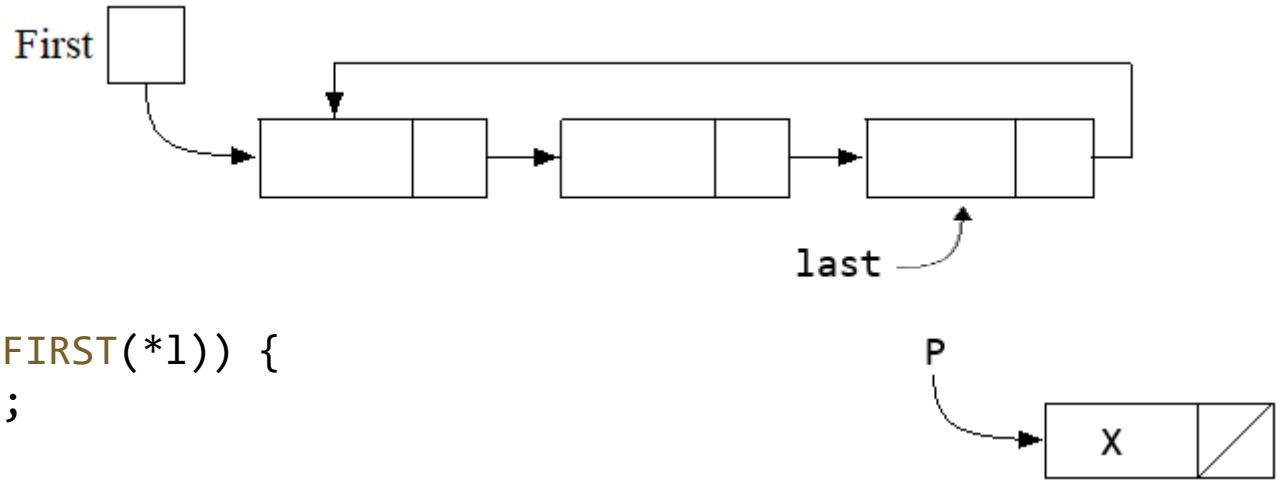
Beberapa primitif

```
void insertFirst(List *l, ElType x) {
/* I.S. List l terdefinisi */
/* F.S. Menambahkan elemen bernilai x sebagai elemen pertama l */
  /* Kamus Lokal */
    address p, last;
  /* Algoritma */
    p = newNode(x);
    if (p != NIL) {
        if (isEmpty(*l)) {
            NEXT(p) = p;
        } else /* *l tidak kosong */ {
            last = FIRST(*l);
            while (NEXT(last) != FIRST(*l)) {
                last = NEXT(last);
            } /* NEXT(last) = FIRST(*l) ==> elemen terakhir */
            NEXT(p) = FIRST(*l);
            NEXT(last) = p;
       }
       FIRST(*l) = p;
    }
}
```



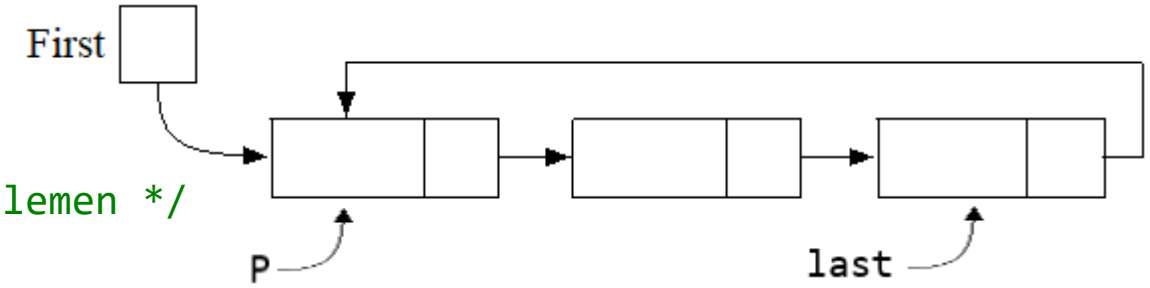
Beberapa primitif

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



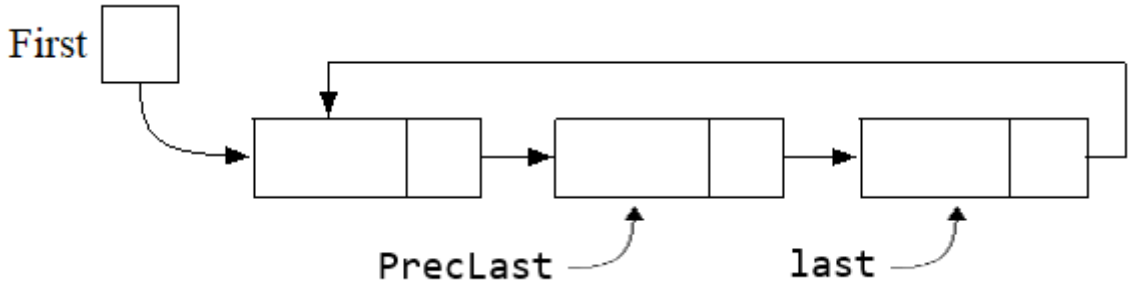
Beberapa primitif

```
void deleteFirst(List *l, ElType *x) {
/* I.S. List 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, last;
  /* Algoritma */
    p = FIRST(*l);  *x = INFO(p);
    if (NEXT(FIRST(*l)) == FIRST(*l)) { /* satu elemen */
        FIRST(*l) = NIL;
    } else {
        last = FIRST(*l);
        while (NEXT(last) != FIRST(*l)) {
            last = NEXT(last);
        }
        /* NEXT(last) = FIRST(*l) */
        FIRST(*l) = NEXT(FIRST(*l)); NEXT(last) = 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 last, precLast;
  /* Algoritma */
    last = FIRST(*l); precLast = NIL;
    while (NEXT(last) != FIRST(*l)) {
        precLast = last; last = NEXT(last);
    } /* NEXT(last) = FIRST(*l) */
    if (precLast == NIL) { /* kasus satu elemen */
        FIRST(*l) = NIL;
    } else {
        NEXT(precLast) = FIRST(*l);
    }
    *x = INFO(last);
    free(last);
}
```



Beberapa primitif

```
void displayList(List l) {
/* I.S. List l mungkin kosong */
/* F.S. Jika list tidak kosong, semua nilai (info) yg disimpan pada elemen list diprint */
/*      Jika list kosong, hanya menuliskan "list kosong" */
    /* Kamus Lokal */
    address p;

    /* Algoritma */
    if (isEmpty(l)) {
        printf("List Kosong \n");
    } else {
        p = FIRST(l);
        printf("List: \n");
        do {
            printf("%d \n", INFO(p));
            p = NEXT(p);
        } while (p != FIRST(l));
    }
}
```