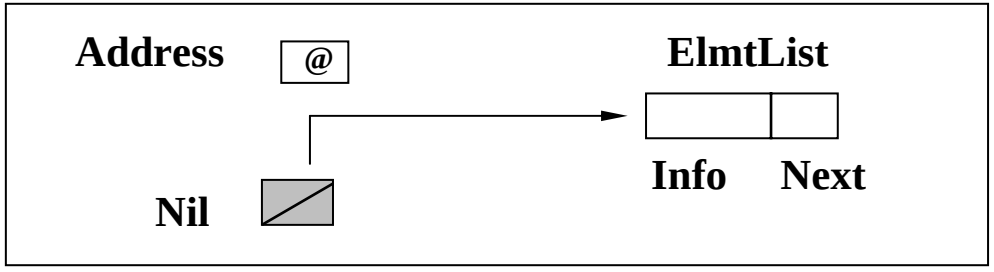
List Linier

IF2030/Algoritma dan Struktur Data

List Linier

- List linier:
  - Sekumpulan elemen ber-type sama yang mempunyai keterurutan tertentu dan setiap elemen terdiri atas 2 bagian:
    - Informasi mengenai elemen (Info)
    - Informasi mengenai alamat elemen suksesor (Next)

```
type ElmtList : < Info : InfoType,
                  Next : address   >
```



List Linier

- Sebuah list linier dikenali dari:
  - **elemen pertamanya**, biasanya melalui alamat elemen pertama yang disebut: **First**.
  - alamat **elemen berikutnya** (suksesor), jika kita mengetahui alamat sebuah elemen, yang dapat diakses melalui informasi NEXT.
  - setiap elemen mempunyai **alamat (address)**, yaitu tempat elemen disimpan dapat diacu. Untuk mengacu sebuah elemen, alamat harus terdefinisi.
  - **elemen terakhirnya**.

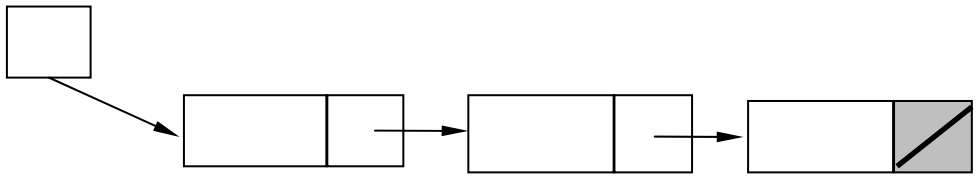
List Linier

- Jika **L** adalah list, dan **P** adalah address:
  - Alamat elemen pertama list L dapat diacu dengan notasi: **First(L)**
  - Elemen yang diacu oleh P dapat dikonsultasi informasinya dengan notasi **Selektor** :
    - **Info(P)**
    - **Next(P)**
- Beberapa definisi:
  - List L adalah list kosong, jika First(L) = Nil
  - Elemen terakhir dikenali, misalnya jika Last adalah alamat elemen terakhir, maka Next(Last) = Nil

List Linier

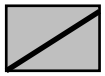
List dengan 3 elemen

First



List Kosong

First



Skema Dasar Pemrosesan List

- Traversal
- Sequential Searching

Skema Traversal

- Skema traversal digunakan untuk memroses setiap elemen list dengan cara yang sama
  - Mekanisme: mengunjungi setiap elemen list dimulai dari elemen pertama, suksesornya, dst. sampai elemen terakhir.
- Skema Traversal 1, 2, 3: Diktat Struktur Data hlm. 65-66.

Skema Traversal 1


```
procedure SKEMAListTraversal1 (input L : List)
{ I.S. List L terdefinisi, mungkin kosong }
{ F.S. semua elemen list L "dikunjungi" dan telah diproses }
{ Traversal sebuah list linier. Dengan MARK, tanpa pemrosesan
   khusus pada list kosong }
KAMUS LOKAL
  P : address { address untuk traversal, type terdefinisi }
  procedure Proses (input P : address )
  { pemrosesan elemen ber-address P }
  procedure Inisialisasi { aksi sebelum proses dilakukan }
  procedure Terminasi
  { aksi sesudah semua pemrosesan elemen selesai }
ALGORITMA
  Inisialisasi
  P ← First(L)
  while (P ≠ Nil) do
    Proses (P)
    P ← Next(P)
  Terminasi
```

Skema Traversal 2


```
procedure SKEMAListTraversal2 (input L : List)
{ I.S. List L terdefinisi, mungkin kosong }
{ F.S. Semua elemen list L "dikunjungi" dan telah diproses }
{ Traversal sebuah list linier yang diidentifikasi oleh elemen pertamanya L }
{ Skema sekuensial dengan MARK dan pemrosesan khusus pada list kosong }
KAMUS LOKAL
  P : address {address untuk traversal }
  procedure Proses (input P : address ) { pemrosesan elemen beraddress P }
  procedure Inisialisasi { aksi sebelum proses dilakukan }
  procedure Terminasi { aksi sesudah semua pemrosesan elemen selesai }
ALGORITMA
  if First(L) = Nil then
    output ("List kosong")
  else
    Inisialisasi
    P ← First(L)
    repeat
      Proses (P)
      P ← Next(P)
    until (P = Nil)
    Terminasi
```

Skema Traversal 3


```
procedure SKEMAListTraversal3 (input L : List)
{ I.S. List L terdefinisi, tidak kosong : minimal mengandung satu elemen }
{ F.S. Semua elemen list L "dikunjungi" dan telah diproses }
{ Skema sekuensial tanpa MARK, tidak ada list kosong karena tanpa mark }
KAMUS LOKAL
  P : address { address untuk traversal, type terdefinisi}
  procedure Proses (input P : address ) { pemrosesas elemen beraddress P }
  procedure Inisialisasi { aksi sebelum proses dilakukan }
  procedure Terminasi { aksi sesudah semua pemrosesan elemen selesai }
ALGORITMA
  Inisialisasi
  P ← First(L)
  iterate
    Proses (P)
  stop (Next(P) = Nil)
    P ← Next(P)
  Terminasi
```

Skema Sequential Search

- Skema sequential search digunakan untuk mencari suatu elemen list
- Searching dapat dilakukan berdasarkan nilai atau berdasarkan alamat
  - Lihat kembali skema sequential search pada array (tabel)
- Diktat Struktur Data hlm. 66 – 68:
  - Search suatu nilai, output adalah address
  - Search suatu elemen yang beralamat tertentu
  - Search suatu elemen dengan kondisi tertentu

Search Nilai, Output Address (1)

```
procedure SKEMAListSearch1 (input L : List, input X : InfoType,
                            output P : address, output Found : boolean)
{ I.S. List linier L sudah terdefinisi dan siap dikonsultasi, X terdefinisi }
{ F.S. P : address pada pencarian beurutan, dimana X diketemukan, P = Nil
   jika tidak ketemu }
{ Found berharga true jika harga X yang dicari ketemu, false jika tidak
   ketemu }
{ Sequential Search harga X pada sebuah list linier L }
{ Elemen diperiksa dengan instruksi yang sama, versi dengan boolean }
KAMUS LOKAL
ALGORITMA
  P ← First(L)
  Found ← false
  while (P ≠ Nil) and (not Found) do
    if (X = Info(P)) then
      Found ← true
    else
      P ← Next(P)
  { P = Nil or Found}
  { Jika Found maka P = address dari harga yg dicari diketemukan }
  { Jika not Found maka P = Nil }
```

Search Nilai, Output Address (2)


```
procedure SKEMAListSearch2 (input L : List, input X : InfoType,
                            output P : address, output Found : boolean)
{ I.S. List linier L sudah terdefinisi dan siap dikonsultasi, X terdefinisi }
{ F.S. P : address pada pencarian beurutan, dimana X ditemukan, P = Nil jika
tidak ketemu }
{ Found berharga true jika harga X yang dicari ketemu, false jika tidak
ditemukan }
{ Sequential Search harga X pada sebuah list linier L }
{ Elemen terakhir diperiksa secara khusus, versi tanpa boolean }
KAMUS LOKAL
ALGORITMA
  { List linier L sudah terdefinisi dan siap dikonsultasi }
  if (First(L) = Nil) then
    output ("List kosong")
    Found ← false; P ← Nil
  else { First(L) ≠ Nil, suksesor elemen pertama ada }
    P ← First(L)
    while ((Next(P) ≠ Nil) and (X ≠ Info(P)) do
        P ← Next(P)
    { Next(P) = Nil or X = Info(P) }
    depend on P, X
        X = Info(P) : Found ← true
        X ≠ Info(P) : Found ← false; P ← Nil
```

Search Elemen Beralamat Tertentu

```
procedure SKEMAListSearch@ (input L : List, input P : address,
output Found : boolean)
{ I.S. List linier L sudah terdefinisi dan siap dikonsultasi, P
   terdefinisi }
{ F.S. Jika ada elemen list yang beralamat P, Found berharga true }
{ Jika tidak ada elemen list beralamat P, Found berharga false }
{ Sequential Search address P pada sebuah list linier L }
{ Semua elemen diperiksa dengan instruksi yang sama }
KAMUS LOKAL
  Pt : address
ALGORITMA
  Pt ← First(L)
  Found ← false
  while (Pt ≠ Nil) and (not Found) do
    if (Pt = P) then
      Found ← true
    else
      Pt ← Next(Pt)
  { Pt = Nil or Found }
  { Jika Found maka P adalah elemen list }
```

Search Elemen Berkondisi Tertentu

```
procedure SKEMAListSearchX (input L : List, input Kondisi(P) : boolean,
output P : address, input Found : boolean)
{ I.S. List linier L sudah terdefinisi dan siap dikonsultasi, Kondisi(P)
   adalah suatu ekspresi boolean yang merupakan fungsi dari elemen
   beralamat P }
{ F.S. Jika ada elemen list P yang memenuhi Kondisi (P), maka P adalah
   alamat dari elemen yang memenuhi kondisi tersebut, Found berharga true }
{ Jika tidak ada elemen list P yang memenuhi Kondisi(P), maka Found
   berharga false dan P adalah Nil }
{ Semua elemen diperiksa dengan instruksi yang sama }
KAMUS LOKAL
  P : address
ALGORITMA
  { List linier L sudah terdefinisi dan siap dikonsultasi }
  P ← First(L)
  Found ← false
  while (P ≠ Nil) and (not Found) do
    if Kondisi(P) then
        Found ← true
    else
        P ← Next(P)
  { P = Nil or Found}
  { Jika Found maka P adalah elemen list dengan Kondisi(P) true }
```

Definisi Fungsional List Linier

Diberikan L, L1 dan L2 adalah list linier dengan elemen ElmtList:

```
IsEmptyList : L → boolean   { Tes apakah list kosong }
CreateList  : → L           { Membentuk sebuah list linier
                              kosong }
Insert      : ElmtList x L → L  { Menyisipkan sebuah elemen
                                  ke dalam list }
Delete      : L → L x ElmtList  { Menghapus sebuah elemen
                                  list }
UpdateList  : ElmtList x L → L  { Mengubah informasi sebuah
                                  elemen list linier }
Concat      : L1 x L2 → L       { Menyambung L1 dengan L2 }
```

Operasi Primitif

- Pemeriksaan apakah list kosong
  - Diktat Struktur Data hlm. 69
- Pembuatan list kosong
  - Diktat Struktur Data hlm. 69-70

Pemeriksaan Apakah List Kosong

```
function IsEmptyList (L : List) → boolean
{ Tes apakah sebuah list L kosong. Mengirimkan true jika
   list kosong, false jika tidak kosong }
KAMUS LOKAL
ALGORITMA
  → (First(L) = Nil)
```

Pembuatan List Kosong


```
procedure CreateList (output L : List)
{ I.S. Sembarang }
{ F.S. Terbentuk list L kosong: First(L) diinisialisasi dengan
   NIL }
KAMUS LOKAL
ALGORITMA
  First(L) ← Nil
```

Operasi Primitif

- Penyisipan elemen pada list:
  - Penyisipan elemen di awal (insert first):
    - Menyisipkan elemen yang diketahui alamatnya
    - Menyisipkan elemen yang diketahui nilainya → sebelum disisipkan harus dialokasi terlebih dahulu, penyisipan hanya bisa dilakukan jika alokasi berhasil
    - Lihat Diktat Struktur Data hlm. 72

Operasi Primitif
Insert-First (Diketahui Alamat)

```
procedure InsertFirst (input/output L : List, input P : address)
{ I.S. List L mungkin kosong, P sudah dialokasi, P ≠ Nil,
   Next(P)=Nil }
{ F.S. P adalah elemen pertama list L }
{ Insert sebuah elemen beralamat P sebagai elemen pertama list
   linier L yang mungkin kosong }
KAMUS LOKAL
ALGORITMA
  Next(P) ← First(L)
  First(L) ← P
```

Operasi Primitif
Insert First (diketahui nilai)


```
procedure InsFirst (input/output L : List, input InfoE : InfoType)
{ I.S. List L mungkin kosong }
{ F.S. Sebuah elemen dialokasi dan menjadi elemen pertama list L, jika
   alokasi berhasil. Jika alokasi gagal list tetap seperti semula. }
{ Insert sebuah elemen sbg. elemen pertama list linier L yang mungkin
   kosong. }
KAMUS LOKAL
  function Alokasi (X : infotype) → address
  { Menghasilkan address yang dialokasi. Jika alokasi berhasil,
  Info(P)=InfoE, dan Next(P)=Nil. Jika alokasi gagal, P=Nil }
  P : address
ALGORITMA
  P ← Alokasi(InfoE)
  if P ≠ Nil then
    Next(P) ← First(L); First(L) ← P
```

Operasi Primitif

- Penyisipan elemen pada list:
  - Penyisipan elemen beralamat P sebagai suksesor sebuah elemen list yang beralamat Prec (insert after)
    - Menyisipkan elemen yang diketahui alamatnya (P)
    - Menyisipkan elemen yang diketahui nilainya → sebelum disisipkan harus dialokasi terlebih dahulu, penyisipan hanya bisa dilakukan jika alokasi berhasil
    - Lihat Diktat Struktur Data hlm. 70-71

Operasi Primitif
Insert After


```
procedure InsertAfter (input P, Prec : address)
{ I.S. Prec adalah elemen list, Prec ≠ Nil, P sudah dialokasi, P
   ≠ Nil, Next(P) = Nil }
{ F.S. P menjadi suksesor Prec }
{ Insert sebuah elemen beralamat P pada List Linier L }
KAMUS LOKAL
ALGORITMA
  Next(P) ← Next(Prec)
  Next(Prec) ← P
```

Operasi Primitif

- Penyisipan elemen pada list:
  - Penyisipan elemen di akhir list (insert last)
    - Ada 2 kemungkinan:
      - List kosong → insert first
      - List tidak kosong → insert sebagai elemen terakhir
    - Menyisipkan elemen yang diketahui alamatnya
    - Menyisipkan elemen yang diketahui nilainya → sebelum disisipkan harus dialokasi terlebih dahulu, penyisipan hanya bisa dilakukan jika alokasi berhasil
    - Lihat Diktat Struktur Data hlm. 73-74

Operasi Primitif
Insert Last (Diketahui Alamat)


```
procedure InsertLast (input/output L : List, input P : address)
{ I.S. List L mungkin kosong, P sudah dialokasi, P ≠ Nil, Next(P)=Nil }
{ F.S. P adalah elemen trerakhir list L }
{ Insert sebuah elemen beralamat P sbg elemen terakhir dari list linier L
   yg mungkin kosong }
KAMUS LOKAL
  Last : address { address untuk traversal,
  pada akhirnya address elemen terakhir }
ALGORITMA
  if First(L) = Nil then { insert sebagai elemen pertama }
    InsertFirst (L,P)
  else
    { Traversal list sampai address terakhir }
    { Bagaimana menghindari traversal list untuk mencapai Last? }
    Last ← First(L)
    while (Next(Last) ≠ Nil) do
      Last ← Next(Last)
    { Next(Last) = Nil, Last adalah elemen terakhir }
    { Insert P after Last }
    InsertAfter(P,Last)
```

Operasi Primitif
Insert Last (Diketahui Nilai)


```
procedure InsLast (input/output L : List, input InfoE : InfoType)
{ I.S. List L mungkin kosong }
{ F.S. Jika alokasi berhasil, InfoE adalah nilai elemen terakhir
   L }
{ Jika alokasi gagal, maka F.S. = I.S. }
{ Insert sebuah elemen beralamat P (jika alokasi berhasil)
   sebagai elemen terakhir dari list linier L yg mungkin kosong }
KAMUS LOKAL
  function Alokasi(X : InfoType) → address
  { Menghasilkan address yang dialokasi. Jika alokasi berhasil,
  Info(P)=InfoE, dan Next(P)= Nil. Jika alokasi gagal, P=Nil }
  P : address
ALGORITMA
  P ← Alokasi(InfoE)
  if P ≠ Nil then { insert sebagai elemen pertama }
    InsertLast(L,P)
```

Operasi Primitif

- Penghapusan elemen list:
  - Dapat menyebabkan list menjadi kosong
  - Penghapusan elemen pertama list (delete first):
    - Elemen yang dihapus dicatat alamatnya
    - Elemen yang dihapus dicatat informasinya dan alamat elemen yang dihapus didealokasi
    - Diktat Struktur Data hlm.74-75

Operasi Primitif
Delete First (1)


```
procedure DeleteFirst (input/output L : List, output P : address)
{ I.S. List L tidak kosong, minimal 1 elemen, elemen pertama
   pasti ada }
{ F.S. First "maju", mungkin bernilai Nil (list menjadi kosong) }
{ Menghapus elemen pertama L, P adalah @ elemen pertama L sebelum
   penghapusan, L yang baru adalah Next(L) }
KAMUS LOKAL
ALGORITMA
  P ← First(L)
  First(L) ← Next(First(L))
  Next(P) ← Nil
  { Perhatikan bahwa tetap benar jika list menjadi kosong }
```

Operasi Primitif
Delete First (2)

```
procedure DeleteFirst (input/output L : List, output E :
   InfoType)
{ I.S. List L tidak kosong, minimal 1 elemen, elemen pertama
   pasti ada }
{ F.S : Menghapus elemen pertama L, E adalah nilai elemen
   pertama L sebelum penghapusan, L yang baru adalah Next(L)
   }
KAMUS LOKAL
  procedure DeAlokasi (input P : address)
  { I.S. P pernah dialokasi. F.S. P=Nil }
  { F.S. Mengembalikan address yang pernah dialokasi. P=Nil}
  P : address
ALGORITMA
  P ← First(L); E ← Info(P)
  First(L) ← Next(First(L)) { List kosong : First(L)
   menjadi Nil }
  Next(P) ← Nil; Dealokasi(P)
```

Operasi Primitif

- Penghapusan elemen list:
  - Penghapusan suksesor sebuah elemen Prec (delete after)
    - Elemen yang dihapus dicatat alamatnya
    - Elemen yang dihapus dicatat informasinya dan alamat elemen yang dihapus didealokasi
    - Diktat Struktur Data hlm.75-76

Operasi Primitif
Delete After


```
procedure DeleteAfter (input Prec : address, output P : address)
{ I.S. List tidak kosong, Prec adalah elemen list, Next(Prec)
  ≠ Nil }
{ F.S. Next(Prec), yaitu elemen beralamat P dihapus dari List.
Next(P)=Nil }
{ Menghapus suksesor Prec, P adalah @ suksesor Prec sebelum
  penghapusan, Next(Prec) yang baru adalah suksesor dari
  suksesor Prec sebelum penghapusan }
KAMUS LOKAL
ALGORITMA
  P ← Next(Prec)
  Next(Prec) ← Next(Next(Prec))
  Next(P) ← Nil
```

Operasi Primitif
DeleteP


```
procedure DeleteP (input/output L : List, input P : address)
{ I.S. List L tidak kosong, P adalah elemen list L }
{ F.S. Menghapus P dari list L, P mungkin elemen pertama,
   "tengah", atau terakhir }
KAMUS LOKAL
  Prec : address { alamat predesesor P }
ALGORITMA
  { Cari predesesor P }
  if (P = First(L)) then { Delete list dengan satu elemen }
    DeleteFirst(L,P)
  else
    Prec ← First(L)
    while (Next(Prec) ≠ P) do
      Prec ← Next(Prec)
    { Next(Prec) = P, hapus P }
    DeleteAfter(Prec,P)
```

Operasi Primitif

- Penghapusan elemen list:
  - Penghapusan elemen di akhir (delete last)
    - Memanfaatkan prinsip delete after → alamat sebelum Last (PrecLast) harus diketahui
    - Kasus: list menjadi kosong dan list menjadi tidak kosong
    - Elemen yang dihapus dicatat alamatnya
    - Elemen yang dihapus dicatat informasinya dan alamat elemen yang dihapus didealokasi
    - Diktat Struktur Data hlm.76-77

Operasi Primitif
Delete Last


```
procedure DeleteLast (input First : List, output P : address)
{ I.S. List L tidak kosong, minimal mengandung 1 elemen }
{ F.S. P berisi alamat elemen yang dihapus. Next(P)=Nil. List berkurang
elemennya }
{ Menghapus elemen terakhir dari list L, list mungkin menjadi kosong }
KAMUS LOKAL
  Last, PrecLast : address { address untuk traversal, type terdefinisi
  pada akhirnya Last adalah alamat elementerakhir dan PrecLast adalah
   alamat sebelum yg terakhir }
ALGORITMA
  { Find Last dan address sebelum Last }
  Last ← First(L)
  PrecLast ← Nil { predesesor dari L tak terdefinisi }
  while (Next(Last) ≠ Nil) do
  { Traversal list sampai @ terakhir }
    PrecLast ← Last
    Last ← Next(Last)
  { Next(Last) = Nil, Last adalah elemen terakhir }
  { PrecLast = sebelum Last }
  P ← Last
  if (PrecLast = Nil) then { list dengan 1 elemen, jadi kosong }
    First(L) ← Nil
  else
    Next(PrecLast) ← Nil
```

Operasi Primitif

- Konkatenasi dua buah list
  - Menggabungkan dua buah list (misal L1 dan L2)
  - Diktat Struktur Data hlm. 78
- Update List
  - Gunakan skema sequential search
  - Buat sebagai latihan

Operasi Primitif
Konkatenasi


```
procedure CONCAT (input L1, L2 : List, output L3 : List)
{ I.S. L1 ≠ L2, L1 ≠ L3, dan L3 ≠ L2; L1, L2 mungkin kosong }
{ F.S. L3 adalah hasil Konkatenasi ("Menyambung") dua buah list linier, L2
ditaruh di belakang L1 }
KAMUS LOKAL
  Last1 : address { alamat elemen terakhir list pertama }
ALGORITMA
  CreateList(L3) { inisialisasi list hasil }
  if First(L1) = Nil then
    First(L3) ← First(L2)
  else { Traversal list 1 sampai address terakhir, hubungkan Last1 dengan
   First(L2) }
    First(L3) ← First(L1)
    Last1 ← First(L1)
    while (Next(Last1) ≠ Nil) do
      Last1 ← Next(Last1)
      { Next(Last1) = Nil, Last adalah elemen terakhir }
      Next(Last1) ← First(L2)
```