STRUKTUR DATA
09 Linear Linked List

Semester Gasal 2024/2025

S1 Informatika FSM UNDIP

Ikhtisar

1) Pohon Konsep ADT
2) Konsensus List Linier
3) Skema Sekuensial List
4) Abstraksi dalam 3 Level

Struktur/Pohon Konsep

1) ADT Atomik/tunggal
2) ADT Majemuk/jamak/kolektif
    a) Representasi Kontigu ~> indexed array
    b) Representasi Berkait ~> linked list

Struktur/Pohon Konsep

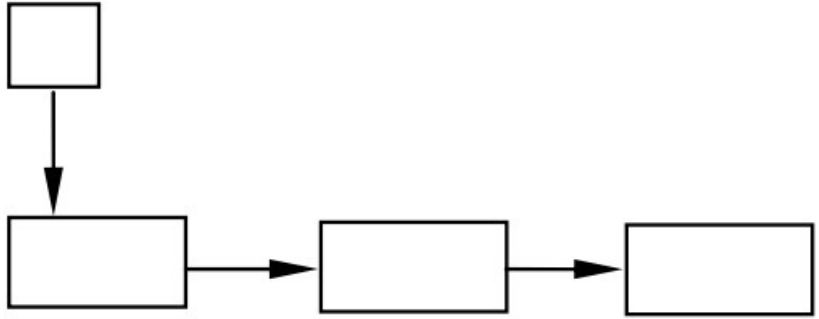
1) ADT Atomik/tunggal
2) ADT Majemuk/jamak/kolektif
   a) Representasi Kontigu ~> indexed array
      i. Tabel, 1 dimensi
      ii. Matriks, banyak dimensi
      iii. Stack, akses 1 pintu
      iv. Queue, akses 2 pintu
   b) Representasi Berkait ~> linked list

Struktur/Pohon Konsep

1) ADT Atomik/tunggal
2) ADT Majemuk/jamak/kolektif
   a) Representasi Kontigu ~> indexed array
   b) Representasi Berkait ~> linked list
      i. Linear: **single**, double
      ii. Circular: single, double
      iii. Tree: binary, N-ary
      iv. Graph: directed, indirected

Single Linear Linked-List

- **List Linier** = sekumpulan elemen bertipe sama yang memiliki “keterurutan” tertentu, dan setiap elemen memiliki dua komponen, yaitu informasi dirinya dan informasi alamat elemen setelahnya.



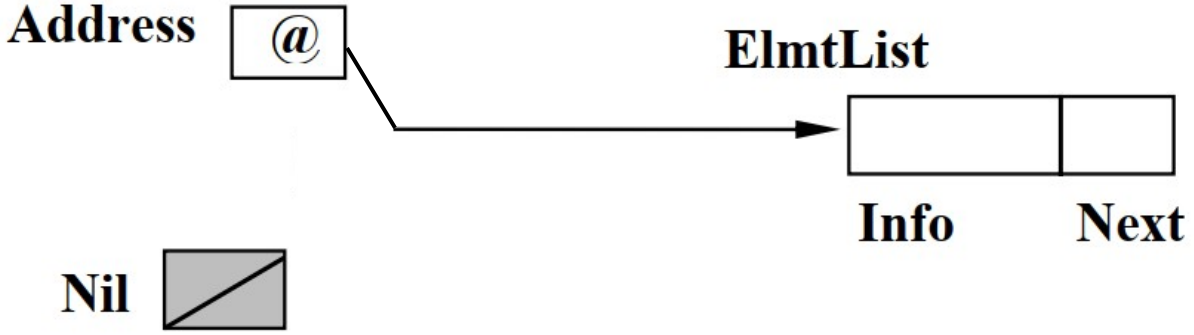
List Linier

Karakteristik Elemen List Linier

- setiap elemen mempunyai **alamat**, yaitu tempat elemen disimpan. Untuk mengacu sebuah elemen, alamat harus terdefinisi. Dari alamat tersebut, informasi dalam elemen dapat diakses.
- elemen pertama, biasanya diacu melalui alamat elemen pertama (**First**)
- alamat elemen **berikutnya** (suksesor), jika diketahui alamat sebuah elemen, yang dapat diakses melalui informasi **next**. Next mungkin ada secara eksplisit (seperti contoh di atas), atau secara implisit yaitu lewat kalkulasi atau fungsi suksesor.
- elemen terakhir, ada banyak cara untuk mengenali elemen akhir

Konsensus Konsep List (1)

- **Alokasi** = pendefinisian alamat elemen list.
- **First** = alamat elemen pertama list, sehingga elemen berikutnya dapat diakses secara suksesif dari field **Next** elemen pertama tersebut.
- **Nil** = konstanta alamat yang tidak terdefinisi.



Konsensus Konsep List (2)

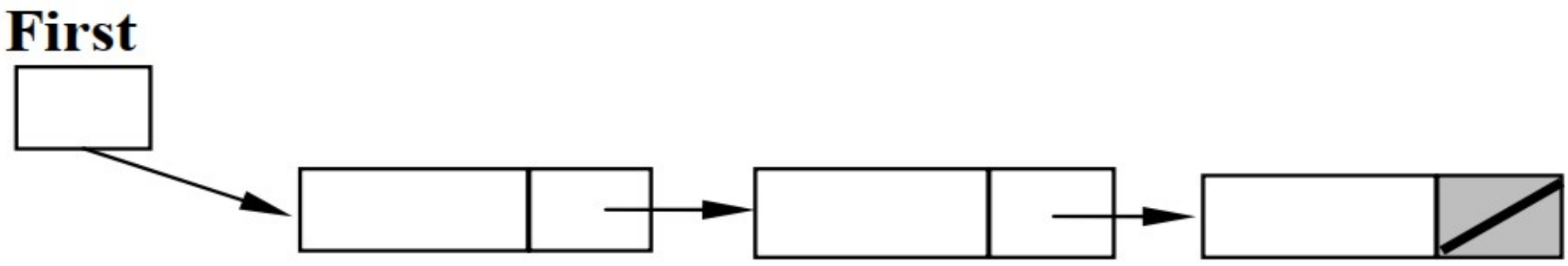
- Jika L adalah list, dan P adalah address:
- Alamat elemen pertama list L dapat diacu dengan notasi :
  - **First(L)**
- Elemen yang diacu oleh P dapat dikonsultasi/dibaca informasinya dengan notasi Selektor :
  - **Info(P)**
  - **Next(P)**

Konsensus Konsep List (3)

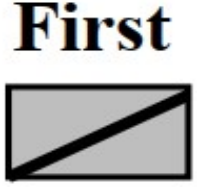
- List L adalah list kosong, jika
  - **First(L) = Nil**.
- Elemen terakhir dikenali, misalnya jika Last adalah alamat elemen terakhir, maka
  - **Next(Last) = Nil**.

Ilustrasi List Linier

List dengan 3 elemen



List Kosong



Skema Sekuensial List

- Ingat kembali **model**/bentuk perulangan di semester II !
- Model perulangan dijadikan dasar (“dibungkus”) dalam menyusun skema sekuensial pemrosesan elemen list
- Skema Penjelajahan (***traversal***) List ada 3 macam
- Skema Pencarian (***search***) List ada 2 macam

Skema Traversal List

- Penjelajahan dimulai dari elemen pertama hingga elemen terakhir
- Skema Traversal 1: dengan MARK tanpa proses kasus list kosong
- Skema Traversal 2: dengan MARK, kasus kosong diproses terpisah
- Skema Traversal 3: tanpa MARK, tidak ada kasus kosong

Skema 1: MARK tanpa kasus kosong

```
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 2: MARK dengan kasus kosong

```
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 ber-
                                             address 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 3: tanpa MARK

```
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 ber-
                                             address 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 Pencarian di List

- Untuk menemukan suatu elemen list berdasarkan nilai informasi yang disimpan pada elemen yang dicari.
- Biasanya dengan alamat yang ditemukan, akan dilakukan suatu proses terhadap elemen list tersebut.
- Skema Search 1: dengan BOOLEAN, pemeriksaan seragam
- Skema Search 2: tanpa BOOLEAN, elemen terakhir diproses khusus

Skema 1 dengan Boolean, Seragam

```
procedure SKEMAListSearch1 (input L : List, input X : InfoType,
                            output P : address, input 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 }
```

Skema 2 tanpa Boolean, Elemen Akhir

```
procedure SKEMAListSearch2 (input L : List, input X : InfoType,
                            output P : address, input 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")
   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
```

Level Abstraksi

1) Definisi Fungsional/Konseptual

   nama tipe bentukan, operasi fungsional primitif

2) Representasi Logik

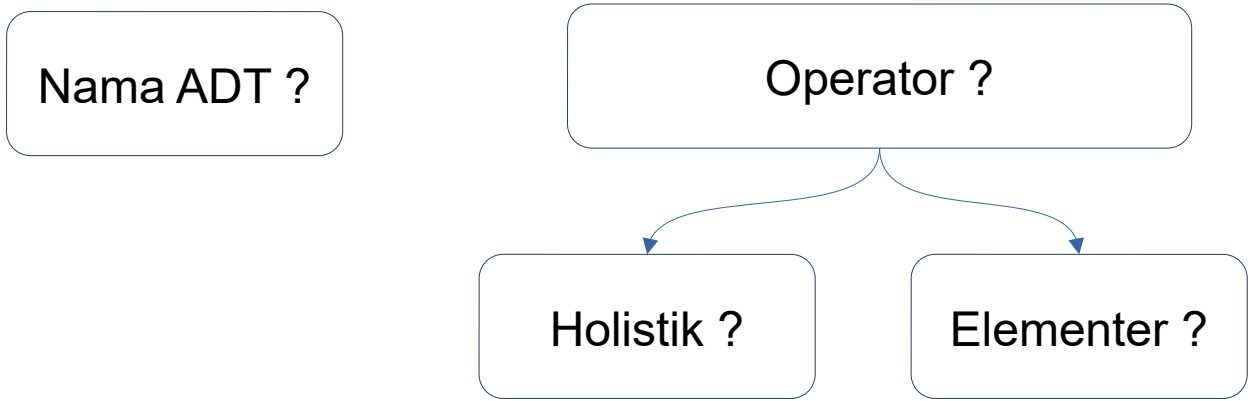
   struktur tipe bentukan, spesifikasi fungsi/prosedur

3) Representasi/implementasi Fisik

   a) representasi kontigu, struktur bersifat statis

   b) representasi berkait, struktur bersifat dinamis

Abstraksi Level 1
Definisi Fungsional
List Linier Kait Tunggal



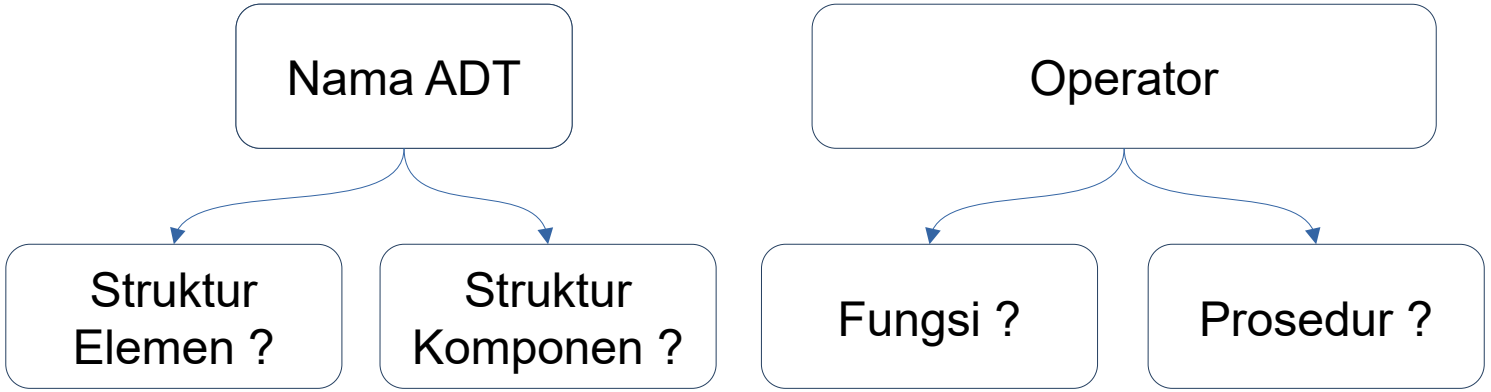
Definisi Fungsional List Linier

- Operator **Holistik**: terkait list secara utuh, misalnya penciptaan, pemeriksaan kekosongan, pemusnahan.
- Operator **Elementer**: terkait elemen kumpulan, misalnya penambahan elemen, pengurangan elemen, update informasi elemen, pencarian elemen, penjelajahan.

Definisi Fungsional List Linier

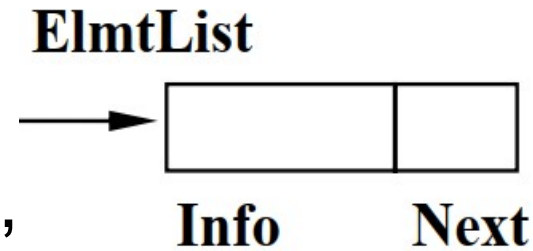
- L, L1 dan L2 adalah list linier dengan elemen ElmList
- **Create** : → L { Membentuk sebuah list linier kosong }
- **IsEmpty** : L → boolean { Tes apakah list kosong }
- **Insert** : ElmList x L → L { Menambah 1 elemen ke dalam list }
- **Delete** : L → L x ElmList { Menghapus sebuah elemen list }
- **Update** : ElmList x L → L { Mengubah info elemen list }
- **Concat** : L1 x L2 → L { Menyambung L1 dengan L2 }

Abstraksi Level 2
Representasi Logik
List Linier Kait Tunggal



Representasi Logik **Elemen** List

- `type` **`ElmList`** = `<`**`Info`**`:InfoType,`**`Next`**`:address>`
- `type` **`address`** = `pointer to ElmList`



- **InfoType** adalah sebuah type terdefinisi yang menyimpan informasi sebuah elemen, bisa tipe primitif, atomik, ataupun majemuk
- **Next** adalah address ("alamat") milik elemen berikutnya (suksesor).

Representasi Logik **ADT** List

- Cara 1 prosedural/sekuensial :

```
type List1 = address
L : List1 {maka First(L) = L}
```

- Cara 2 berbasis objek (**rekomendasi**) :

```
type List1 = < First : address >
L : List1 {maka First(L) = L.First}
```

Operator Holistik ADT List

1. Penciptaan
2. Pemusnahan
3. Pemeriksaan
4. Penyambungan
5. Pemecahan
6. Penggandaan

Operator Elementer ADT List

0. Alokasi Memori
1. Penambahan Elemen
2. Penghapusan Elemen
3. Pengubahan Isi Elemen
4. Pencarian Elemen
5. Penjelajahan Elemen
6. Operator lain

Pustaka

1. Inggriani Liem. Diktat Struktur Data. ITB. 2008
2. Debdutta Pal, Suman Halder. Data Structures and Algorithms with C. Alpha Science International Ltd. 2018.
3. AHO, Alfred V., John E. Hopcroft, Jeffrey D. Ullman. Data Structures and Algorithm. Addison Weshley Publishing Compani.1987