STRUKTUR DATA
09 Linear Linked List

Semester Gasal 2024/2025

S1 Informatika FSM UNDIP

Intisari

1) Struktur Logik Elemen List Linier
2) Struktur Logik Komponen ADT List Linier
3) Fungsi/Prosedur Holistik List Linier
4) Fungsi/Prosedur Elementer List Linier

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

Level Abstraksi

1) Definisi Fungsional/Konseptual

   nama tipe bentukan, operasi fungsional primitif

2) Representasi Logik

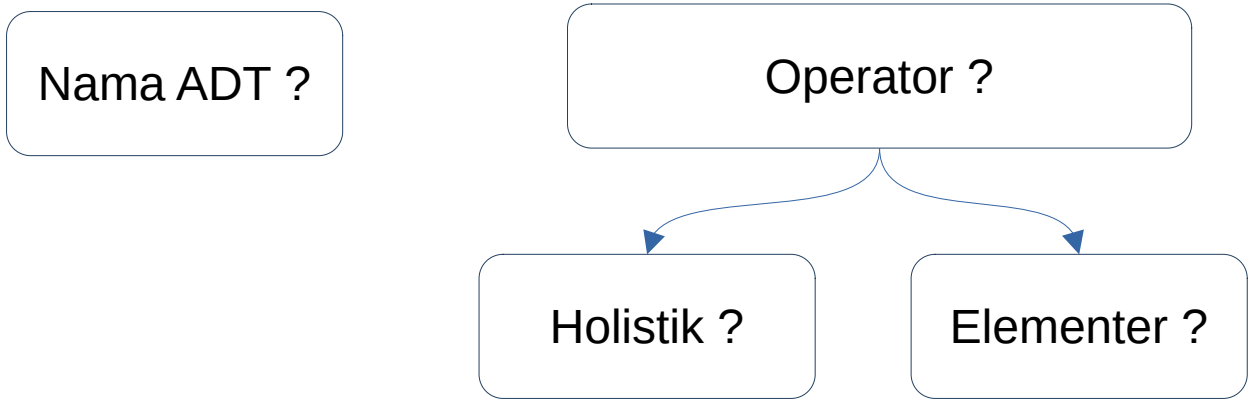
   struktur tipe bentukan, spesifikasi fungsi/prosedur

3) Representasi/implementasi Fisik

   a) representasi kontigu, struktur bersifat statis

   b) representasi berkait, struktur bersifat dinamis

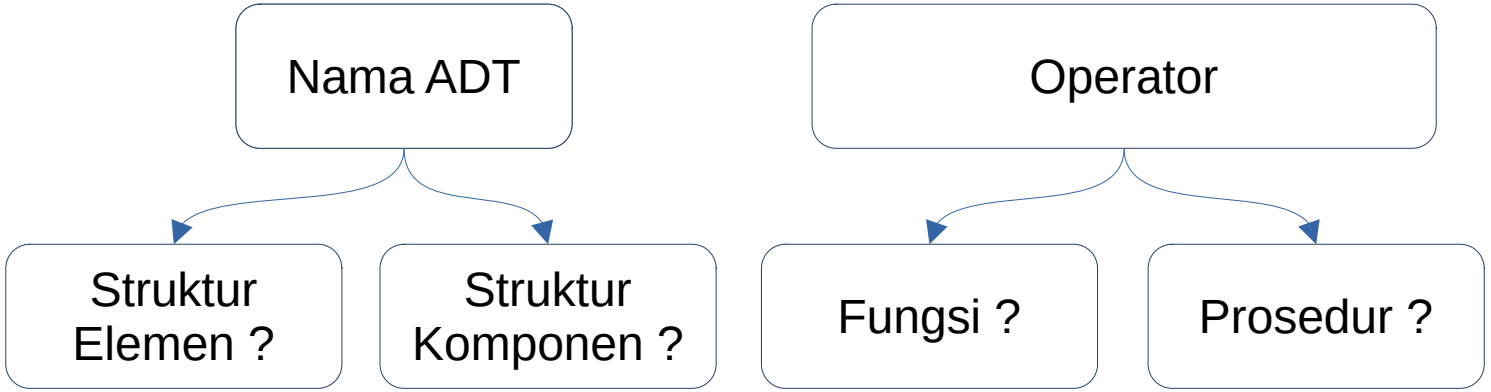
Abstraksi Level 1
Definisi Fungsional
List Linier Kait Tunggal



Definisi Fungsional List Linier

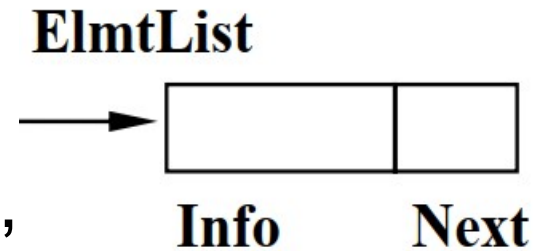
- L, L1 dan L2 adalah list linier dengan elemen ElmList
- **Create** : → L { Membentuk sebuah list linier kosong }
- **IsEmpty** : L → boolean { Tes apakah list kosong }
- **Insert** : ElmList x L → L { Menambah 1 elemen ke dalam list }
- **Delete** : L → L x ElmList { Menghapus sebuah elemen list }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

1.Penciptaan, terdapat 2 pilihan :

```
Procedure CreateList(output L:List1)
{I.S.: - ; F.S.: L list kosong}
Function NewList() --> List1
{mengembalikan list kosong}
```

Operator Holistik ADT List

2.Pemusnahan

```
Procedure DestroyList(input/output L:List1)
{I.S.: L terdefinisi ; F.S.: L musnah}
```

3.Pemeriksaan

```
Function IsEmptyList(L:List1) --> boolean
{mengembalikan true bila list kosong}
```

Operator Holistik ADT List

4.Penyambungan

```
Procedure ConcatList(input L1:List1,
 input L2:List1, output L:List1)

{I.S.: L1,L2 terdefinisi ;
 F.S.: L gabungan L1 dan L2}
```

5.Pemecahan

```
Procedure SplitList(input L:List1,
 output L1:List1, output L2:List1)

{I.S.: L terdefinisi ;
 F.S.: L1, L2 hasil pemecahan L}
```

Operator Holistik ADT List

6.Penggandaan

```
Procedure CopyList(input L1:List1,
output L2:List1)

{I.S.: L1 terdefinisi;
 F.S.: L2 salinan L1}
```

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

Operator Elementer ADT List

0.Alokasi Memori:

```
Function Alokasi(E:infotype)->address
{mengembalikan alamat elemen E}
Procedure Dealokasi(input/output
P:address)
{I.S.: P terdefinisi; F.S.: P musnah}
```

Operator Elementer ADT List

1. Penambahan Elemen:
   a. Berdasarkan Nilai/Value:
      insertVFirst, insertVLast
   b. Berdasarkan Alamat:
      InsertFirst, insertAfter, insertLast

Operator Elementer ADT List

1.Penambahan Elemen Awal, berbasis Value

```
Procedure InsertVFirst(input/output
L:List1, input V:infotype )

{ I.S. List L mungkin kosong }
{ F.S. P dialokasi, Info(P)=V }
{ Proses: menyisipkan sebuah elemen
beralamat P dengan Info(P)=V sebagai
elemen pertama list linier L }
```

Operator Elementer ADT List

1.Penambahan Elemen Akhir, berbasis Value

```
Procedure InsertVLast(input/output
L:List1, input V:infotype )

{ I.S. List L mungkin kosong }
{ F.S. P dialokasi, Info(P)=V }
{ Proses menyisipkan sebuah elemen
beralamat P dengan Info(P)=V sebagai
elemen terakhir list linier L }
```

Operator Elementer ADT List

2. Penghapusan Elemen:
   a. Berdasarkan Nilai/Value:
      deleteVFirst, deleteVLast
   b. Berdasarkan Alamat:
      deleteFirst, deleteAfter, deleteLast

Operator Elementer ADT List

2.Penghapusan Elemen Awal, berbasis Value

```
Procedure DeleteVFirst(input/output L:List1,
output V:infotype )

{ I.S. List L mungkin kosong }
{ F.S. Elemen pertama list L dihapus, dan
didealokasi. Hasil penghapusan disimpan
nilainya dalam V.

List mungkin menjadi kosong. Jika tidak
kosong, elemen pertama yang baru adalah
elemen sesudah elemen pertama yang lama }
```

Operator Elementer ADT List

2.Penghapusan Elemen Akhir, berbasis Value

```
Procedure DeleteVLast(input/output L:List1,
output V:infotype )

{ I.S. List L mungkin kosong }
{ F.S. Elemen terakhir list L dihapus, dan
didealokasi. Hasil penghapusan disimpan
nilainya dalam V.

List mungkin menjadi kosong. Jika tidak
kosong, elemen terakhir yang baru adalah
elemen sebelum elemen terakhir yang lama}
```

Operator Elementer ADT List

5. Penjelajahan Elemen:

   NbElm, getMin, getMax, getSum, printList

4. Pencarian Elemen:

   SearchX, CountX, getPrecX

3. Pengubahan Nilai Elemen:

   UpdateX, UpdateAllX

Operator Elementer ADT List

5.Penjelajahan Elemen

```
function NbElm(L:List1) --> integer
{ menghitung banyaknya elemen list L}
procedure PrintList(input L:List1)
{I.S. L terdefinisi; F.S. :-}
{ menampilkan semua elemen list L}
```

Operator Elementer ADT List

4.Pencarian Elemen, berbasis Value

```
Procedure SearchX(input L:List1, input
X:infotype, output A:address )

{ I.S. L, X terdefinisi }
{ F.S. A = alamat elemen yang bernilai X.

Mencari apakah ada elemen list dengan
info(P)= X. Jika ada, mengisi A dengan
address elemen tersebut. Jika tidak ada,
A=Nil }
```

Operator Elementer ADT List

3.Pengubahan Elemen, berbasis Value

```
Procedure UpdateX(input/output L:List1,
input X:infotype, input Y:infotype)

{ I.S. L, X, Y terdefinisi }
{ F.S. L tetap, atau elemen bernilai X
berubah menjadi bernilai Y.

Proses: mengganti elemen bernilai X
menjadi bernilai Y}
```

Operator Elementer ADT List

6.Operator lain: Invers, finvers, sort

```
procedure Invers(input/output L:List1)
{ I.S. L terdefinisi,
  F.S. semua elemen L terbalik
  urutannya dari posisi awal}
{ Proses: membalik urutan posisi
  elemen list L }
```

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