STRUKTUR DATA
12 Double Linear Linked List

Semester Gasal 2024/2025

S1 Informatika FSM UNDIP

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
   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

Operator ADT Koleksi

Holistik:

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

Elementer:

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

Definisi Fungsional List Kait Ganda

- Diberikan L, L1 dan L2 adalah list linier dengan elemen Elm
- **ListEmpty** : L → boolean { Tes apakah list kosong }
- **CreateList** : → L { Membentuk sebuah list kosong }
- **Insert** : Elm x L → L { Menyisipkan 1 elemen ke dalam list }
- **Delete** : L → L x Elm { Menghapus sebuah elemen list }
- **UpdateList** : Elm x L → L { Mengubah informasi elemen list }
- **Concat** : L1 x L2 → L { Menyambung L1 dengan L2 }

Siklus Pemrograman

1) Requirement
2) Analisis
3) Desain
4) Implementasi
5) Testing

Definisi Logik + Spesifikasi

Mengurai kasus yang mungkin

Realisasi Algoritmik

Coding ke bahasa pemrograman

Aplikasi fungsi/prosedur+Debugging

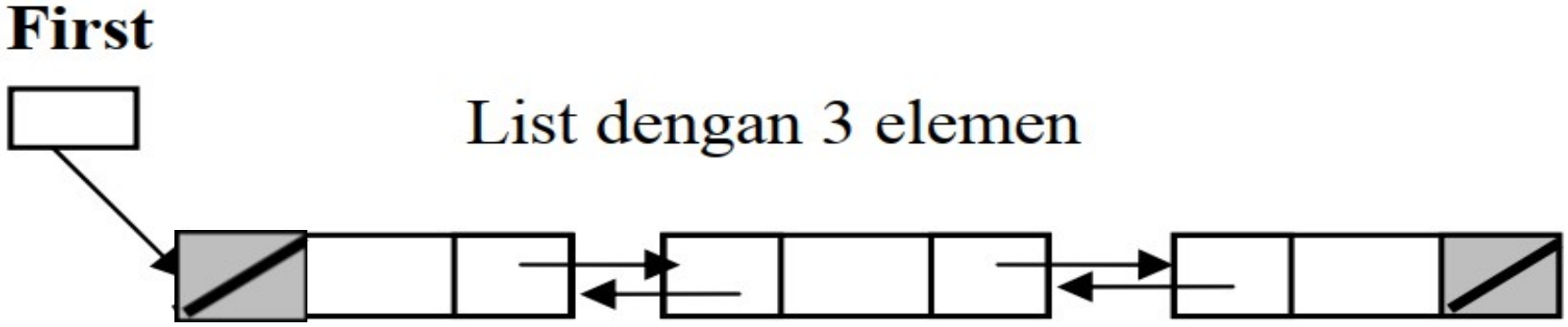
Selektor dalam List

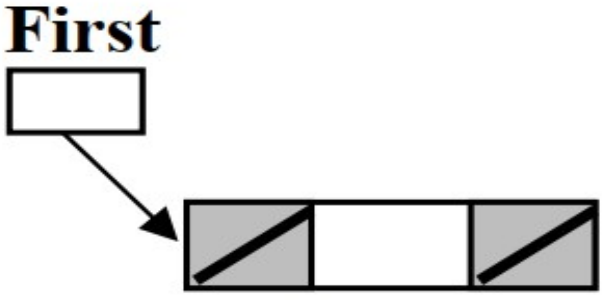
- Jika L adalah list, dan P adalah address:
- Alamat elemen pertama list L diacu dengan notasi :
  - **First(L)**
- Elemen yang diacu oleh P dikonsultasi/dibaca informasinya dengan notasi Selektor :
  - **Prev(P)**
  - **Info(P)**
  - **Next(P)**

Selektor dalam List

- Jika L adalah list, dan last adalah address:
- **Prev(First(L)) = NIL**
- **Next(last) = NIL,** jika last adalah elemen terakhir
- List kosong memiliki ciri:
  First(L) = NIL
- List dengan 1 elemen tunggal memiliki ciri:
  Prev(First(L)) = NIL and Next(First(L)) = NIL

Ilustrasi List Kait Ganda

- 



Representasi Logik Elemen List

```
type Elm = < Prev:address,
             Info:InfoType,
             Next:address >

type address = pointer to Elm
```

InfoType adalah sebuah type terdefinisi yang menyimpan informasi sebuah elemen, bisa tipe primitif, atomik, ataupun majemuk

Prev adalah alamat elemen sebelumnya (predesesor)

Next adalah alamat elemen berikutnya (suksesor).

Representasi Logik ADT List

- Cara 1 prosedural/sekuensial :

```
type List3 = address

L : List3 {maka First(L) = L }
```

- Cara 2 berbasis objek (rekomendasi) :

```
type List3 = < First : address >

L : List3 {maka First(L) = L.First }
```

Coding ADT ke Bahasa C

```
/* type infotype = character */
typedef char infotype; /* elemen bertipe character */
/* type address = pointer to Elm */
typedef struct tElm * address;
/* Representasi address dengan pointer */
/* type Elm =< info:infotype, next:address > */
typedef struct tElm { address  prev;
                      infotype info;
                      address  next;
                    } Elm;
/* typedef List3 =< First:address >*/
typedef struct { address First; } List3;
```

Operator Elementer ADT List

0.Alokasi Memori:

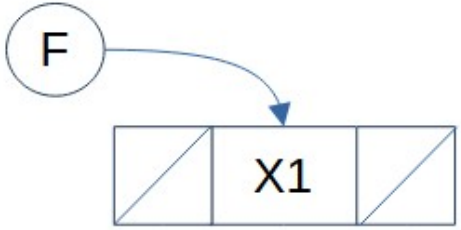
```
Function Alokasi(E:infotype)->address
{mengembalikan alamat elemen E}
Procedure Dealokasi(input/output
P:address)
{I.S.: P terdefinisi; F.S.: P musnah}
```

Operator PrintList

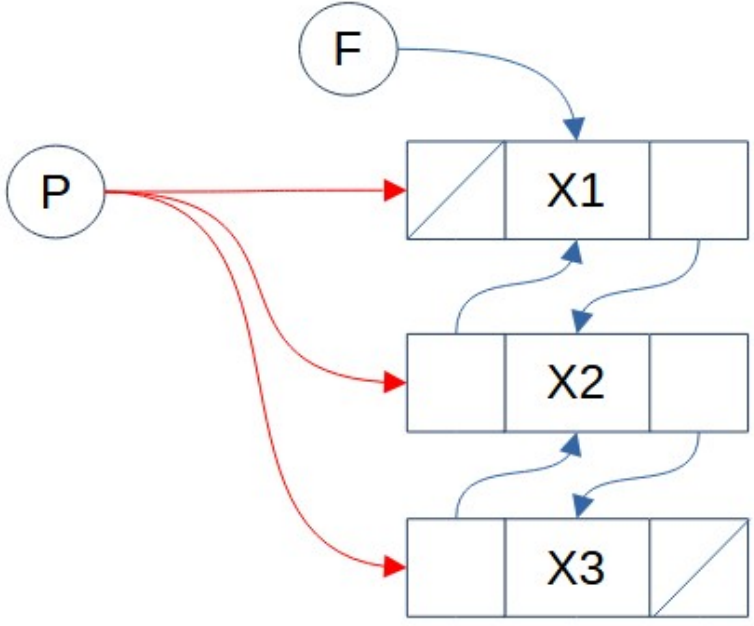
```
procedure PrintList(input L:List3)
{I.S. L terdefinisi; F.S. :-}
{ menampilkan semua elemen list L}
```

Kasus-kasus PrintList

1) Satu Elemen



2) Banyak Elemen



Algoritma PrintList

```
procedure PrintList(input L:List3)
{ I.S. L terdefinisi; F.S. :-}
{ Proses: menampilkan semua elemen list L}
kamus lokal
   P : address
algoritma
   P <- First(L)
   if P =/= NIL then
       repeat
            output info(P)
            P  <- next(P)
       until P = NIL
```

Coding PrintList di Bahasa C

```
/*procedure PrintList(input L:List3)
{I.S. L terdefinisi; F.S. :-}
{ menampilkan semua elemen list L}*/
void PrintList(List3 L) {
    //kamus lokal
    address P;
    //algoritma
    P = First(L);
    if (P != NIL) {
        printf("\nElemen:");
        do {
            printf("\t %c",info(P));
            P = next(P);
        } while( P != NIL );
    }
}
```

5.Aplikasi

```
//kamus
  List3 Senarai;
//algoritma
  PrintList( Senarai );
```

Operator Holistik ADT List

1.Penciptaan, terdapat 2 pilihan :

```
Procedure CreateList(output L:List3)
{I.S.: - ; F.S.: L list kosong}
Function GetNewList() --> List3
{mengembalikan list kosong}
```

Operator Holistik ADT List

2.Pemusnahan

```
Procedure DestroyList(input/output L:List3)
{I.S.: L terdefinisi ; F.S.: L musnah}
```

3.Pemeriksaan

```
Function IsEmptyList(L:List3) --> boolean
{mengembalikan true bila list kosong}
```

Operator Holistik ADT List

4.Penyambungan

```
Procedure ConcatList(input L1:List3,
 input L2:List3, output L:List3)
{I.S.: L1,L2 terdefinisi ;
 F.S.: L gabungan L1 dan L2}
```

5.Pemecahan

```
Procedure SplitList(input L:List3,
 output L1:List3, output L2:List3)
{I.S.: L terdefinisi ; F.S.: L1, L2 hasil
pemecahan L1}
```

Operator Holistik ADT List

6.Penggandaan

```
Procedure CopyList(input L1:List3,
output L2:List3)

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

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:List3, input V:infotype )

{ I.S. List L mungkin kosong }

{ F.S. P dialokasi, Info(P)=V }

{ Insert sebuah elemen beralamat P
dengan Info(P)=V sebagai elemen pertama
list linier L yg mungkin kosong }
```

Operator Elementer ADT List

2.Penambahan Elemen Akhir, berbasis Value

```
Procedure InsertVLast(input/output
L:List3, input V:infotype )

{ I.S. List L mungkin kosong }

{ F.S. P dialokasi, Info(P)=V }

{ Insert sebuah elemen beralamat P
dengan Info(P)=V sebagai elemen akhir
list linier L yg mungkin kosong }
```

Operator Elementer ADT List

3.Penambahan Elemen tertentu, berbasis Value

```
Procedure InsertVAfterX(input/output
L:List3, input X:infotype, input
V:infotype )
{ I.S. List L mungkin kosong }
{ F.S. P dialokasi, Info(P)=V }
{ Insert sebuah elemen beralamat P
dengan Info(P)=V sebagai elemen dengan
posisi setelah elemen bernilai X }
```

Operator Elementer ADT List

4.Penambahan Elemen tertentu, berbasis Value

```
Procedure InsertVBeforeX(input/output
L:List3, input X:infotype, input
V:infotype )
{ I.S. List L mungkin kosong }
{ F.S. P dialokasi, Info(P)=V }
{ Insert sebuah elemen beralamat P
dengan Info(P)=V sebagai elemen dengan
posisi sebelum elemen bernilai X }
```

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:List3,
output V:infotype )
{ I.S. List L tidak 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 list yang baru adalah
elemen sesudah elemen pertama yang lama. }
```

Operator Elementer ADT List

2.Penghapusan Elemen Akhir, berbasis Value

```
Procedure DeleteVLast(input/output L:List3,
output V:infotype )
{ I.S. List L tidak 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 list yang baru adalah
elemen sebelum elemen terakhir yang lama. }
```

Operator Elementer ADT List

3.Penghapusan Elemen Tertentu, berbasis Value

```
Procedure DeleteX(input/output
L:List3, input X:infotype)

{ I.S. List L tidak kosong }

{ F.S. List mungkin menjadi kosong,
atau berkurang 1 elemen. }

{ Proses: Elemen bernilai X dihapus,
dan didealokasi. }
```

Operator Elementer ADT List

3.Penghapusan Elemen Tertentu, berbasis Value

```
Procedure DeleteVAfterX(input/output
L:List3, input X:infotype, output
V:infotype )

{ I.S. List L tidak kosong }

{ F.S. Elemen setelah X dihapus, dan
didealokasi. Hasil penghapusan disimpan
nilainya dalam V.

List mungkin menjadi kosong. }
```

Operator Elementer ADT List

3.Penghapusan Elemen Tertentu, berbasis Value

```
Procedure DeleteVBeforeX(input/output
L:List3, input X:infotype, output
V:infotype )

{ I.S. List L tidak kosong }

{ F.S. Elemen sebelum X dihapus, dan
didealokasi. Hasil penghapusan disimpan
nilainya dalam V.

List mungkin menjadi kosong. }
```

Operator Elementer ADT List

4. Pencarian Elemen:

   SearchX, searchPrec

3. Pengubahan Nilai Elemen:

   UpdateX, updateAllX

5. Penjelajahan Elemen:

   NbElm, getMin, getMax, getSum, printList

Operator Elementer ADT List

5.Penjelajahan Elemen

```
function NbElm(L:List3) --> integer
{ menghitung banyaknya elemen list L}
```

Operator Elementer ADT List

4.Pencarian Elemen, berbasis Value

```
Procedure SearchX(input L:List3, input
X:infotype, output A:address )

{ I.S. L, X terdefinisi }

{ F.S. A berisi alamat elemen yang nilainya 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:List3,
input X:infotype, input Y:infotype)
{ I.S. L, X, Y terdefinisi }
{ F.S. L tetap, atau elemen bernilai X
berubah menjadi Y.
Mengganti elemen bernilai X menjadi Y}
```

Operator Elementer ADT List

6.Operator lain:

Invers, finvers, sort

```
procedure Invers(input/output L:List3)
{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