

Representasi Fisik List Linier: Struktur Berkait dengan Pointer

IF2110/IF2111 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Representasi implisit dan eksplisit

Sebelumnya kita telah menjumpai list berkait dengan representasi **implisit**

yaitu struktur data list berkait di mana *menunjuk ke sebuah list* adalah sama dengan *menunjuk ke elemen pertamanya*:

```
type ElType: ...  
type Address: pointer to Node  
type Node: < info: ElType,  
             next: Address >  
type List: Address
```

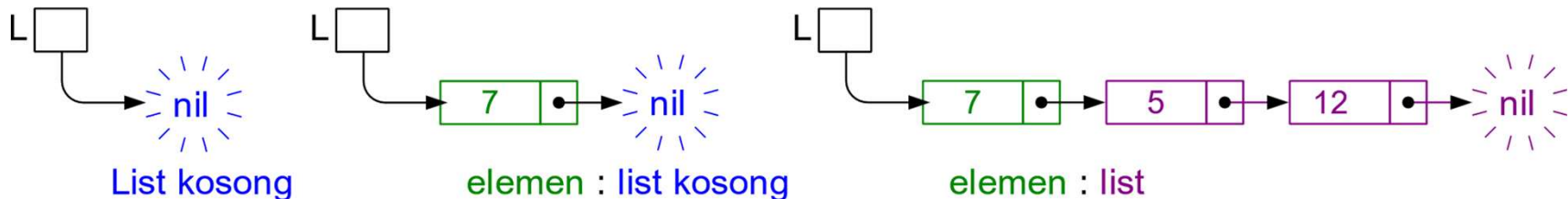
Terdapat representasi lain yaitu **eksplisit**, di mana elemen pertama list merupakan *bagian dari* struktur data list:

```
type List: < first: Address >
```

Representasi implisit

Representasi implisit mewakili definisi rekursif sebuah list linier seperti dalam paradigma fungsional:

- List kosong adalah list.
- List tidak kosong terdiri atas sebuah elemen yang diikuti list.



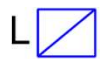
Elemen pertama list L, First = L.

Next dari First harus merupakan list juga, \therefore type List: Address

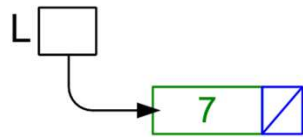
Representasi implisit

Representasi implisit mewakili definisi rekursif sebuah list linier seperti dalam paradigma fungsional:

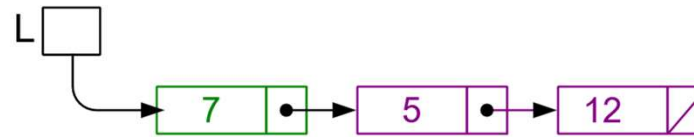
- List kosong adalah list.
- List tidak kosong terdiri atas sebuah elemen yang diikuti list.



List kosong



elemen : list kosong



elemen : list

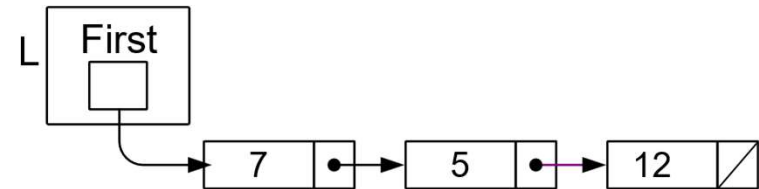
Elemen pertama list L, First = L.

Next dari First harus merupakan list juga, \therefore type List: Address

Representasi eksplisit

Dalam representasi eksplisit, First merupakan *bagian* dari struktur data list.

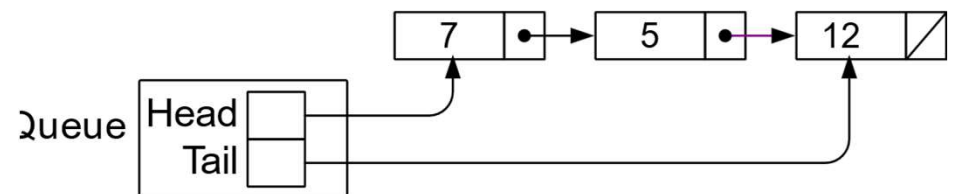
```
type List: < first: Address >
```



\therefore akses ke elemen pertama `l` adalah `l.first`.

Representasi ini berguna misalnya pada implementasi Queue memanfaatkan struktur list berkait:

```
type Queue: < head: Address,  
              tail: Address >
```



Proses rekursif

```
type ElType: ...  
type Address: pointer to Node  
type Node: < info: ElType,  
             next: Address >  
type List: Address
```

{ Representasi Implisit }



```
procedure displayList(input l: List)  
{ I.S. l terdefinisi  
  F.S. Setiap elemen l di-print }
```

KAMUS LOKAL

ALGORITMA

```
if (isEmpty(l)) then { Basis 0 }  
  { tidak melakukan apa-apa }  
else { Rekurens }  
  output(l↑.info)  
  PrintList(l↑.next)
```

...

displayList(l)

Proses rekursif

```
type ElType: ...  
type Address: pointer to Node  
type Node: < info: ElType,  
             next: Address >  
type List: < first: Address >
```

{ Representasi Eksplisit }



*{ PROBLEMS:
 l tidak memiliki info.
 l tidak memiliki next. }*

```
procedure displayList(input l: List)  
{ I.S. L terdefinisi  
  F.S. Setiap elemen list diprint }
```

KAMUS LOKAL

ALGORITMA

```
if (isEmpty(l)) then { Basis 0 }  
  { tidak melakukan apa-apa }  
else { Rekurens }  
  output(l↑.info)  
  PrintList(l↑.next)
```

...

displayList(l)

Proses rekursif

```
type ElType: ...  
type Address: pointer to Node  
type Node: < info: ElType,  
             next: Address >  
type List: < first: Address >
```

{ Representasi Eksplisit }



*{ PROBLEM:
l.first↑.next merupakan Address,
tidak bisa di-pass ke displayList yang
menerima sebuah List. }*

```
procedure displayList(input l: List)  
{ I.S. L terdefinisi  
  F.S. Setiap elemen list diprint }
```

KAMUS LOKAL

ALGORITMA

```
if (isEmpty(l)) then { Basis 0 }  
  { tidak melakukan apa-apa }  
else { Rekurens }  
  output(l.first↑.info)  
  PrintList(l.first↑.next)
```

...

displayList(l)

Proses rekursif

```
type ElType: ...  
type Address: pointer to Node  
type Node: < info: ElType,  
             next: Address >  
type List: < first: Address >
```

{ Representasi Eksplisit }



*{ PROBLEM:
p merupakan Address, tidak bisa di-pass
ke isEmpty yang menerima sebuah List. }*

```
procedure displayList (input p: Address)  
{ I.S. p terdefinisi  
  F.S. Setiap elemen list diprint }
```

KAMUS LOKAL

ALGORITMA

```
if (isEmpty(p)) then { Basis 0 }  
  { tidak melakukan apa-apa }  
else { Rekurens }  
  output(p↑.info)  
  displayList(p↑.next)
```

...

```
displayList(l.first)
```