

# Skema Pemrosesan List Berkait

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

# Skema dasar pemrosesan List berkait

- Traversal
- Pencarian sekuensial (*search*)

# Skema traversal

Digunakan untuk memproses setiap elemen List dengan cara yang sama.

Mekanisme: mengunjungi setiap elemen List secara berurutan dimulai dari elemen pertama hingga elemen terakhir.

Jenis traversal:

- Dasar (tanpa perlakuan khusus untuk List kosong)
- Dengan perlakuan khusus untuk List kosong
- Untuk List yang tidak kosong

# Skema traversal dasar

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 tanpa pemrosesan khusus untuk 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 ← l

while (p ≠ NIL) do

proses(p)

    p ← p↑.next

terminasi

→ bebas,

mis. mau

jumlahin isi list

jd  $sum \leftarrow 0$

→ proses

$sum \leftarrow sum + p \uparrow info$

$p \leftarrow p \uparrow next$

# Skema traversal dengan penanganan list kosong

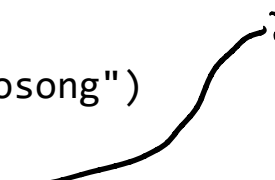
```
procedure SKEMAlistTraversal2(input l: List)
{ I.S. List l terdefinisi, mungkin kosong. }
{ F.S. Semua elemen List l "dikunjungi" dan telah diproses. }
{ Traversal sebuah List linier dengan pemrosesan khusus untuk List kosong. }
```

## KAMUS LOKAL

{ SAMA SEPERTI SKEMA 1, tidak ditulis untuk menghemat tempat }

## ALGORITMA

```
if l = NIL then
    output("List kosong")
else
    inisialisasi
    p ← l
    repeat
        proses(p)
        p ← p↑.next
    until (p = NIL)
terminasi
```



```
if l = NIL then
    output ("List kosong")
else
    sum ← 0
    count ← 0
    p ← l
    repeat
        sum ← sum + p↑.info
        count ← count + 1
        p ← p↑.next
    until (p = NIL)
    avg ← sum / count
    output (avg)
```

# Skema traversal untuk List tidak kosong

```
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. }
{ Traversal untuk List linier yang sudah dipastikan tidak kosong. }
```

## KAMUS LOKAL

```
{ SAMA SEPERTI SKEMA 1 dan 2, tidak ditulis untuk menghemat tempat }
```

## ALGORITMA

```
inisialisasi
p ← l
iterate
    proses(p)
stop (p↑.next = NIL)
    p ← p↑.next
terminasi
```

# Skema pencarian sekuensial

List linier tidak memungkinkan *binary search*.

Mekanisme: mengunjungi elemen-elemen List secara berurutan dimulai dari elemen pertama hingga ditemukan elemen yang memenuhi syarat pencarian, atau semua elemen telah dikunjungi.

Jenis skema pencarian:

- Dengan boolean
- Tanpa boolean

# Pencarian berdasarkan nilai elemen (1)

```
procedure SKEMAlistSearch1(input l: List, input x: ElType,  
                           output p: Address, output found: boolean)  
{ I.S. List linier l sudah terdefinisi dan siap dikonsultasi, x terdefinisi }  
{ F.S. p adalah address di mana x ditemukan, p = NIL jika tidak ketemu }  
{      found menyatakan apakah nilai x yang dicari ditemukan }  
{ Menggunakan skema search dengan boolean }
```

## KAMUS LOKAL

-

## ALGORITMA

```
p ← l  
found ← false  
while (p ≠ NIL) and (not found) do  
  if (x = p↑.info) then  
    found ← true  
  else  
    p ← p↑.next  
{ p = NIL or found }  
{ Jika found maka p = Address dari nilai yg dicari }  
{ p = NIL jika nilai tidak ditemukan }
```



# Pencarian berdasarkan nilai elemen (2)

```
procedure SKEMAlistSearch2(input l: List, input x: ElType,  
                           output p: Address, output found: boolean)  
{ I.S. List linier l sudah terdefinisi dan siap dikonsultasi, x terdefinisi }  
{ F.S. p adalah address di mana x ditemukan, p = NIL jika tidak ketemu }  
{      found menyatakan apakah nilai x yang dicari ditemukan }  
{ Menggunakan skema search tanpa boolean }
```

## KAMUS LOKAL

-

## ALGORITMA

```
p ← l  
if p = NIL then { List kosong }  
  found ← false  
else { List tidak kosong }  
  while (p↑.next ≠ NIL) and (p↑.info ≠ x) do  
    p ← p↑.next  
    { p↑.next = NIL or p↑.info = x }  
  found ← (p↑.info = x)  
  if (not found) then  
    p ← NIL
```

# Pencarian elemen yang memenuhi kondisi

```
procedure SKEMAlistSearchX(input l: List, input kondisi(p): boolean,  
                           output p: Address, output 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. p adalah address di mana kondisi(p) terpenuhi, p = NIL jika tidak ketemu }  
{   found menyatakan apakah ada p yang memenuhi kondisi(p) }
```

## KAMUS LOKAL

-

## ALGORITMA

```
p ← l  
found ← false  
while (p ≠ NIL) and (not found) do  
  if kondisi(p) then  
    found ← true  
  else  
    p ← p↑.next  
{ p = NIL or found }  
{ Jika found maka p adalah elemen List dengan kondisi(p) true }
```

## mencari minimum dan maksimum

if (p = NIL) then

output ("List kosong")

else

p ← 1

max ← p↑.info

min ← p↑.info

p ← p↑.next

while (p↑.next ≠ NIL) do

if p↑.info ≥ max then

max ← p↑.info

if p↑.info ≤ min then

min ← p↑.info

p ← p↑.next

output (max)

output (min)

function searchpos (L: list) → address

KAMUS LOKAL

found : boolean

p : address

ALGORITMA

found ← false

p ← L

while (p ≠ NIL) and (found = false) do

if p↑.info > 0 then

found ← true

else

p↑.next

if (found) then

→ p

else

→ NIL