ALGORITMA & PEMROGRAMAN

Oleh:

Tim Algoritma & Pemrograman IF

Linked List

PENGERTIAN LINKED LIST

- Salah satu bentuk struktur data, berisi kumpulan data (node) yang **tersusun** secara sekuensial, **saling sambung-menyambung, dinamis** dan **tidak terbatas.**
- Linked List sering disebut juga Senarai Berantai
- Linked List saling terhubung dengan bantuan variabel pointer
- Masing-masing data dalam Linked List disebut dengan node (simpul) yang menempati alokasi memori secara dinamis dan biasanya berupa record

Array vs Linked List

ARRAY	LINKED LIST
Statis	Dinamis
Penambahan / penghapusan data terbatas	Penambahan / penghapusan data tidak terbatas
Random access	Sequential access
Penghapusan array tidak mungkin	Penghapusan linked list mudah

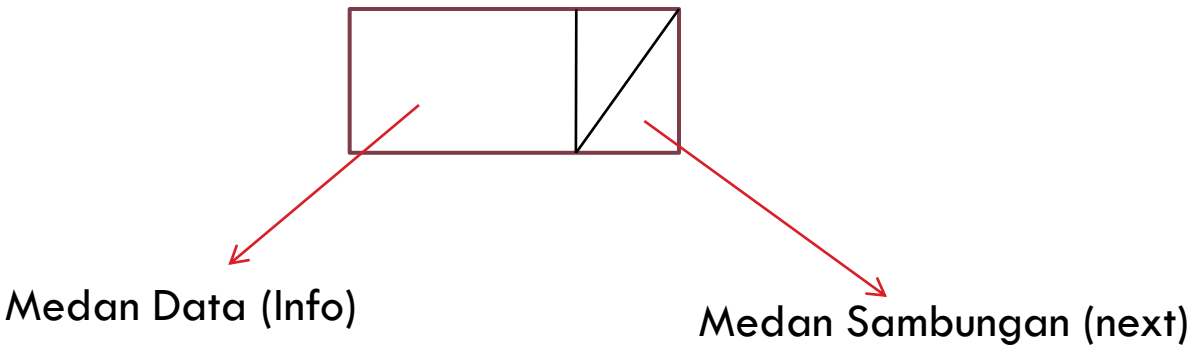
Bentuk Linked List

- Single Linked List
- Double Linked List
- Circular Linked List

Single Linked List

Linked list dengan simpul berisi satu link / pointer yang mengacu ke simpul berikutnya.

Simpul Single Linked List :



Deklarasi Linked List


```
Type
        nama_pointer = ↑Simpul
        Simpul  = Record
                medan_data  : tipedata,
                medan_sambungan : Namapointer
        EndRecord

nama_var_pointer : nama_pointer
```

Contoh Deklarasi Linked List


```
Type
   Point  =  ↑Data
   Data  = Record
           Info   : char ,
           Next  : Point
   Endrecord
awal, akhir : Point
```

Operasi – operasi Single Linked List

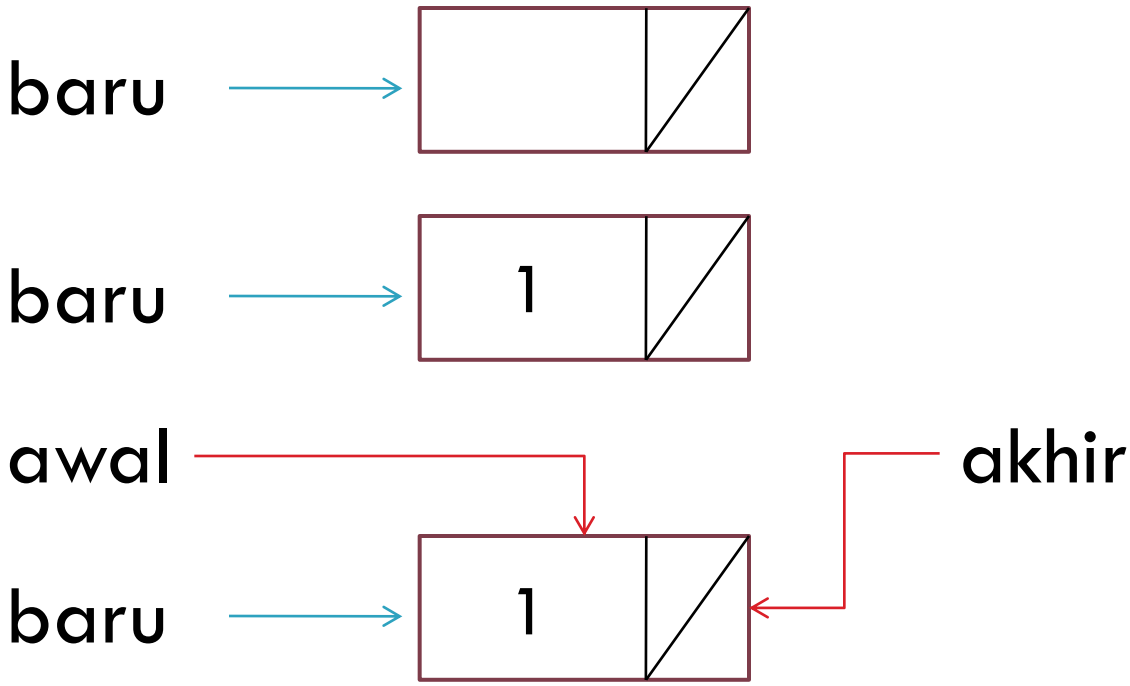
1. Penciptaan (create)
2. Penyisipan
3. Penghapusan
4. Traversal
5. Pencarian (Searching)
6. Pengurutan (Sorting)
7. Penghancuran (destroy)

Penciptaan

Pointer awal dan akhir diberi harga nil.

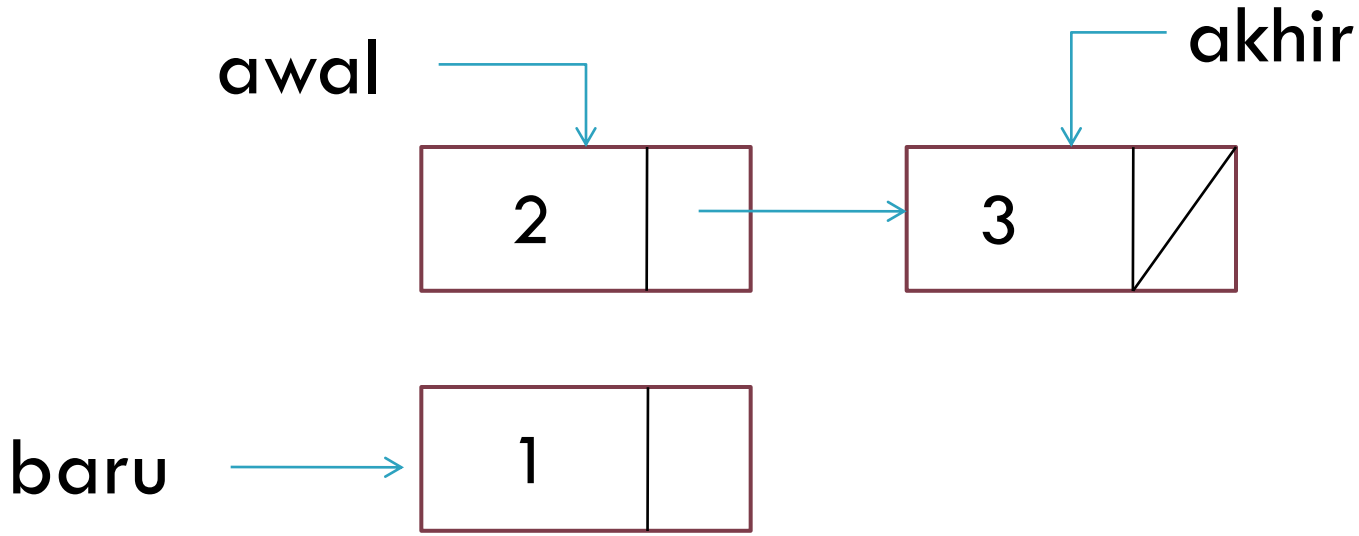
Penyisipan di Depan

- List kosong {awal = nil}

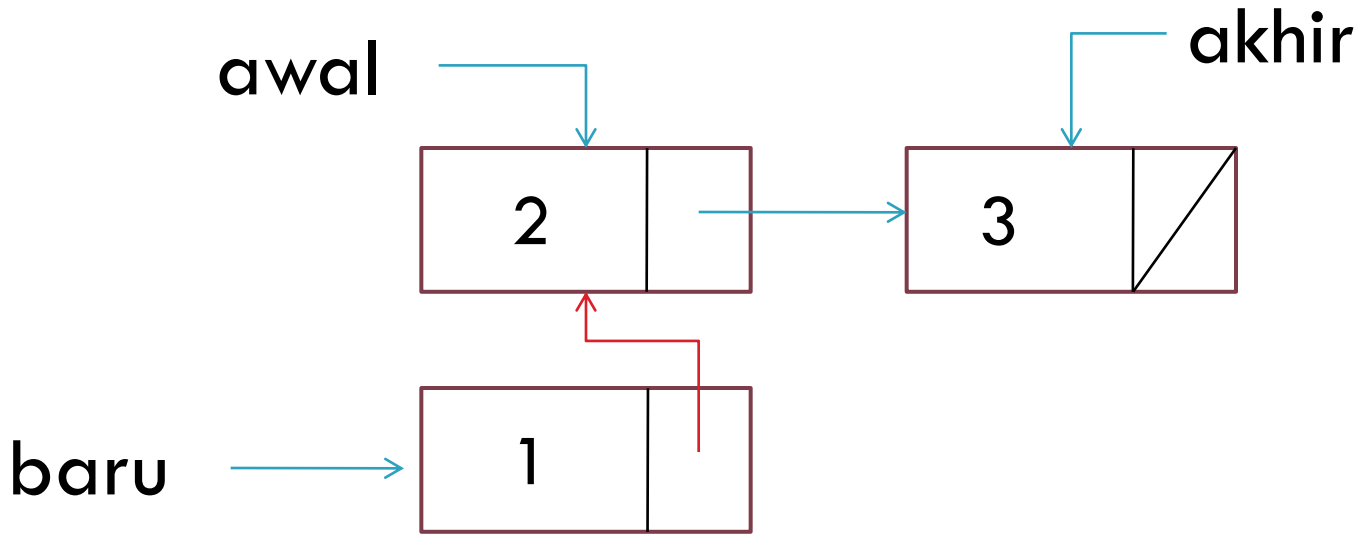


Penyisipan di Depan (lanjutan)

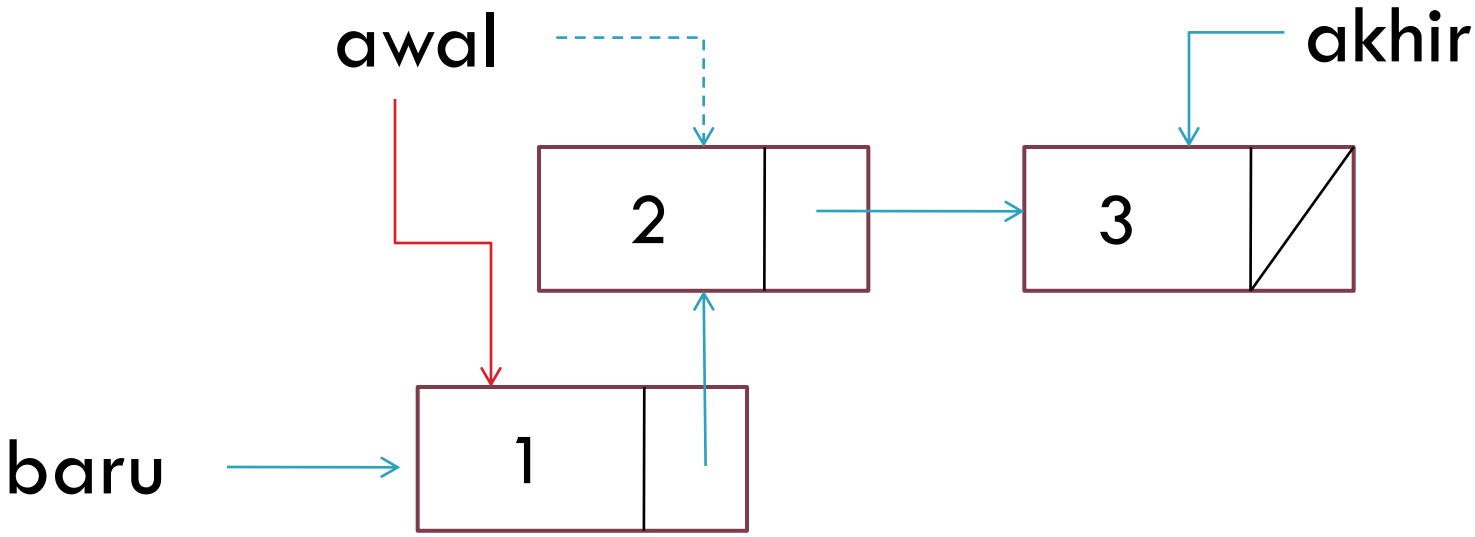
- **List tidak kosong {Awal ≠ Nil}**



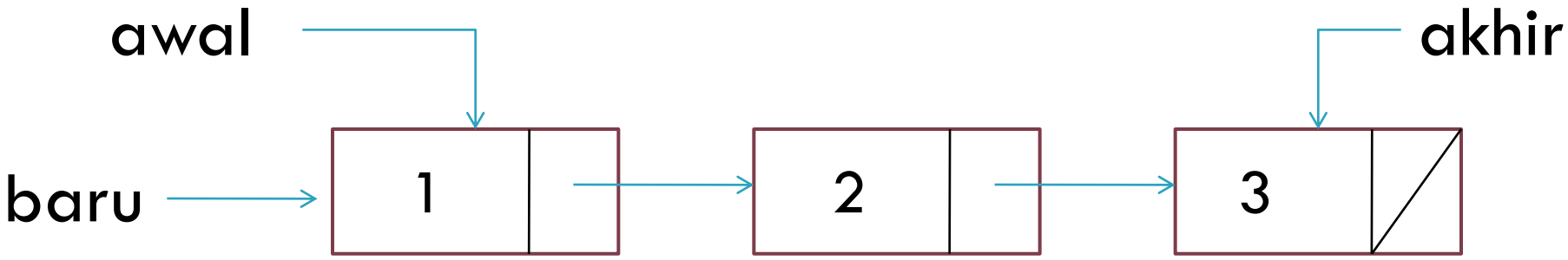
Penyisipan di Depan (lanjutan)



Penyisipan di Depan (lanjutan)



Penyisipan di Depan (lanjutan)



Algoritma Penyisipan di Depan

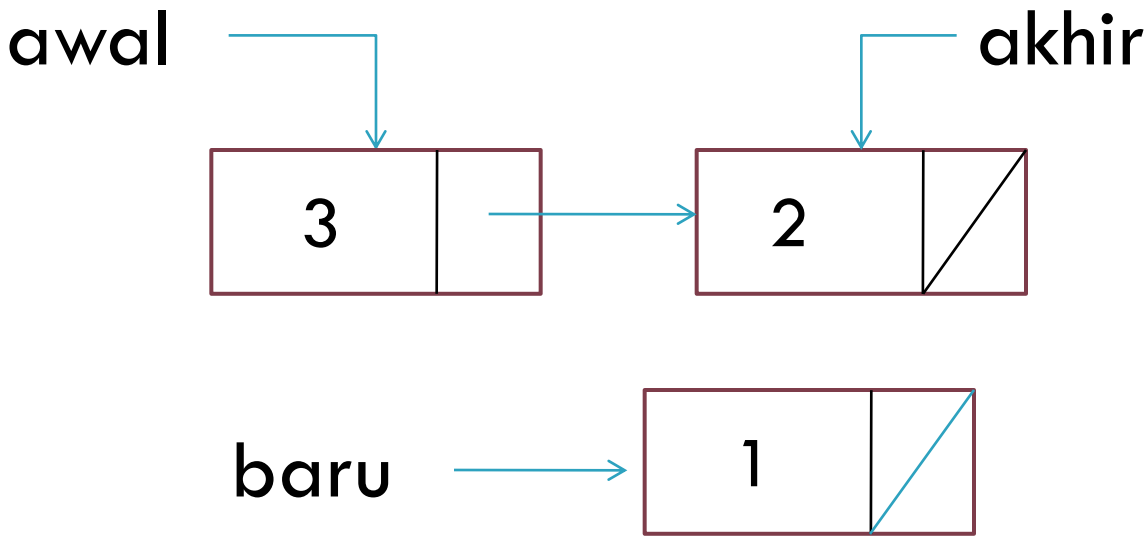

```
Procedure  SisipDepanSingle(Input  elemen : tipedata, I/O  awal, akhir : nama_pointer)
{I.S. : data yang akan disisipkan (elemen), pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}
{F.S. : menghasilkan satu simpul yang disisipkan di depan pada single linked list}
Kamus :
    baru : nama_pointer
Algoritma :
    Alloc(baru)
    baru↑.info ← elemen
    If (awal = nil)
     Then
            baru↑.next ← nil
            akhir ← baru
     Else
            baru↑.next ← awal
    EndIf
    awal ← baru
EndProcedure
```

Penyisipan di Belakang

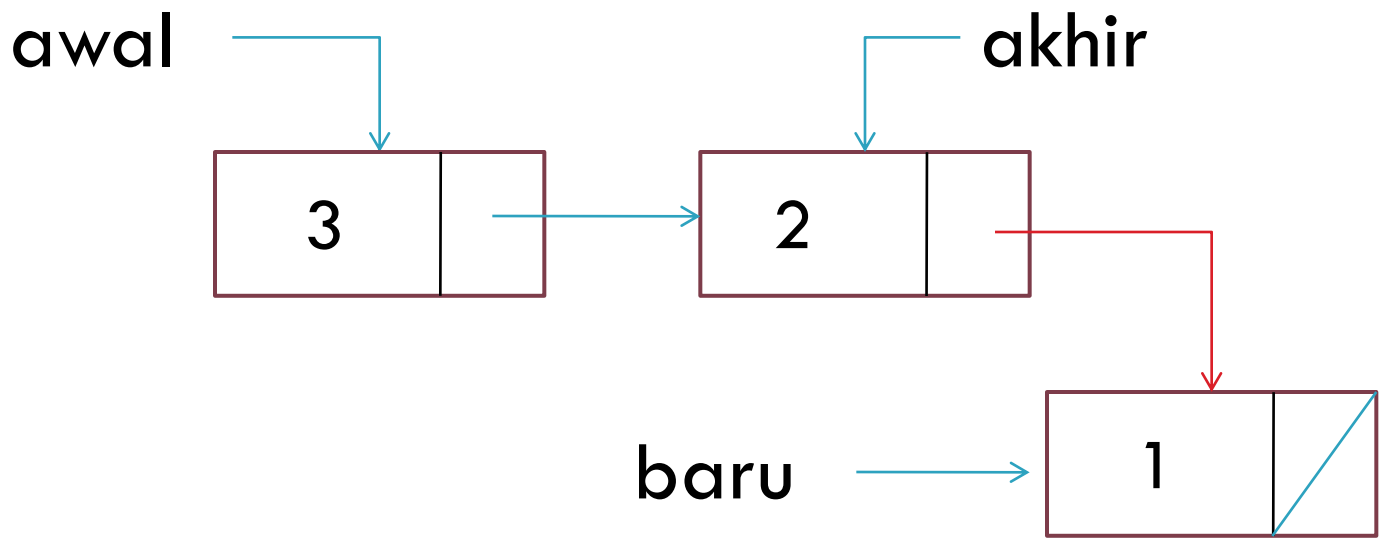
- List kosong {awal = nil}

{sama seperti pada penyisipan di depan}

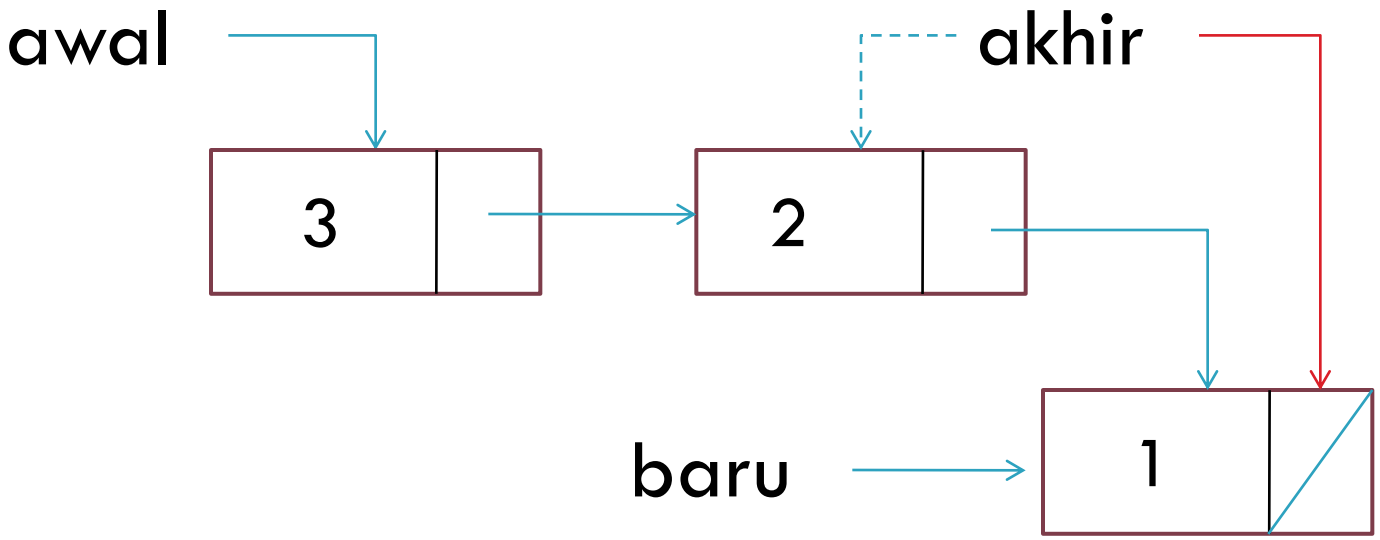
- List tidak kosong {awal ≠ Nil}



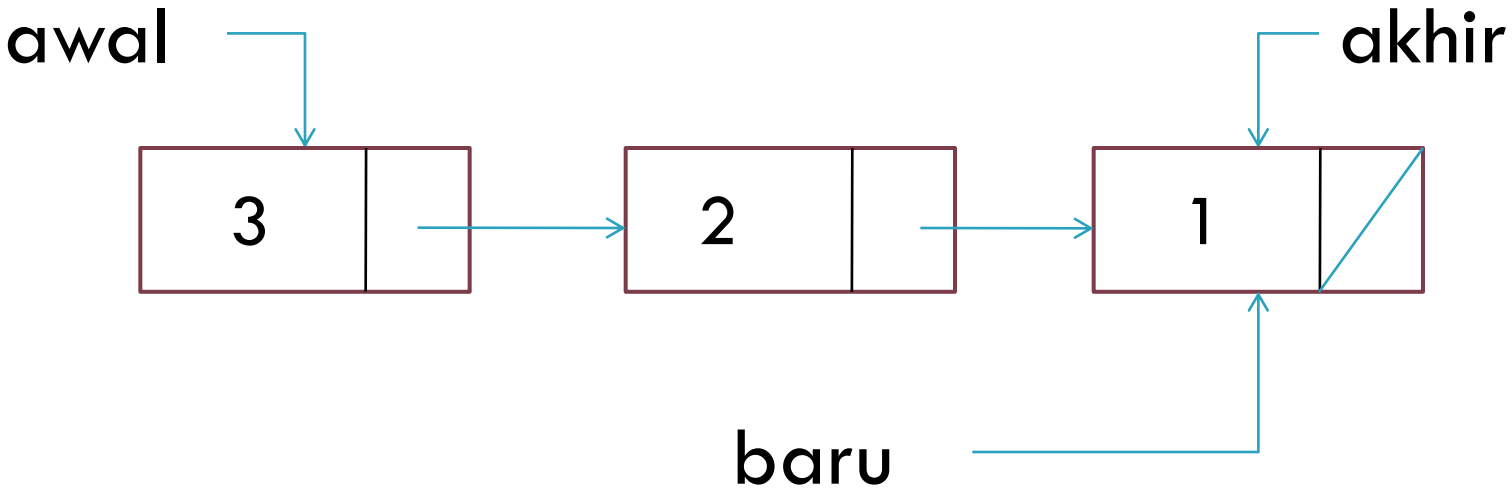
Penyisipan di Belakang (lanjutan)



Penyisipan di Belakang (lanjutan)



Penyisipan di Belakang (lanjutan)



Algoritma Penyisipan di Belakang

Procedure SisipBelakangSingle(Input elemen : tipedata, I/O awal, akhir : nama_pointer)

{I.S. : data yang akan disisipkan (elemen), pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}

{F.S. : menghasilkan satu simpul yang disisipkan di belakang pada single linked list}

Kamus :

```
    baru : nama_pointer
```

Algoritma :

```
    Alloc(baru)
    baru↑.info ← elemen
    baru↑.next ← nil
    If (awal = nil)
      Then
            awal ← baru
      Else
            akhir↑.next ← baru
    EndIf
    akhir ← baru
```

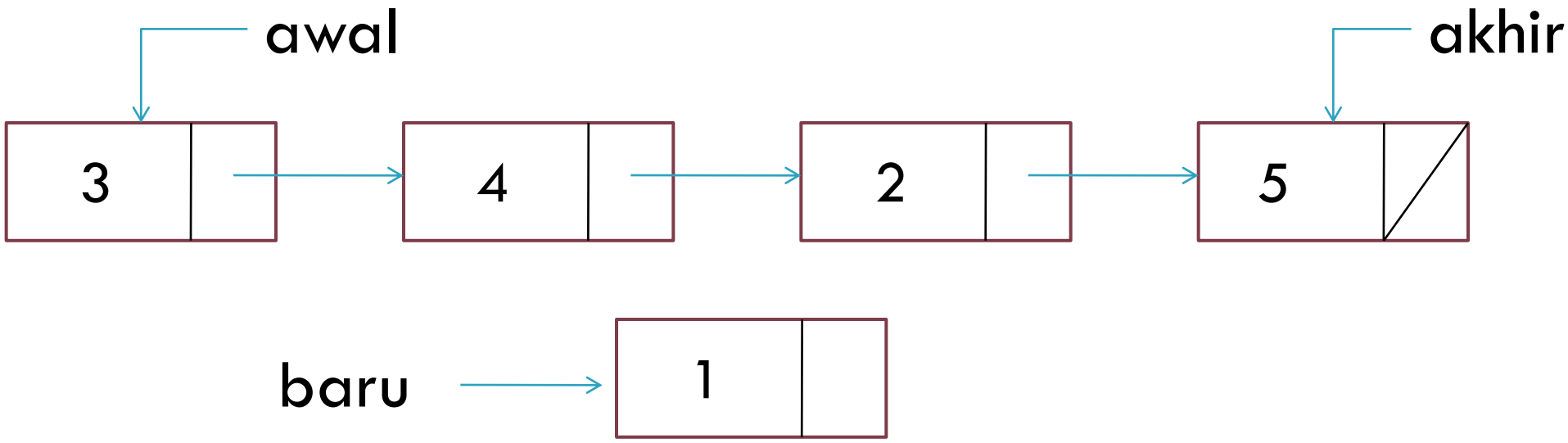
EndProcedure

Penyisipan di Tengah

- List kosong {awal = nil}

{sama seperti pada penyisipan di depan}

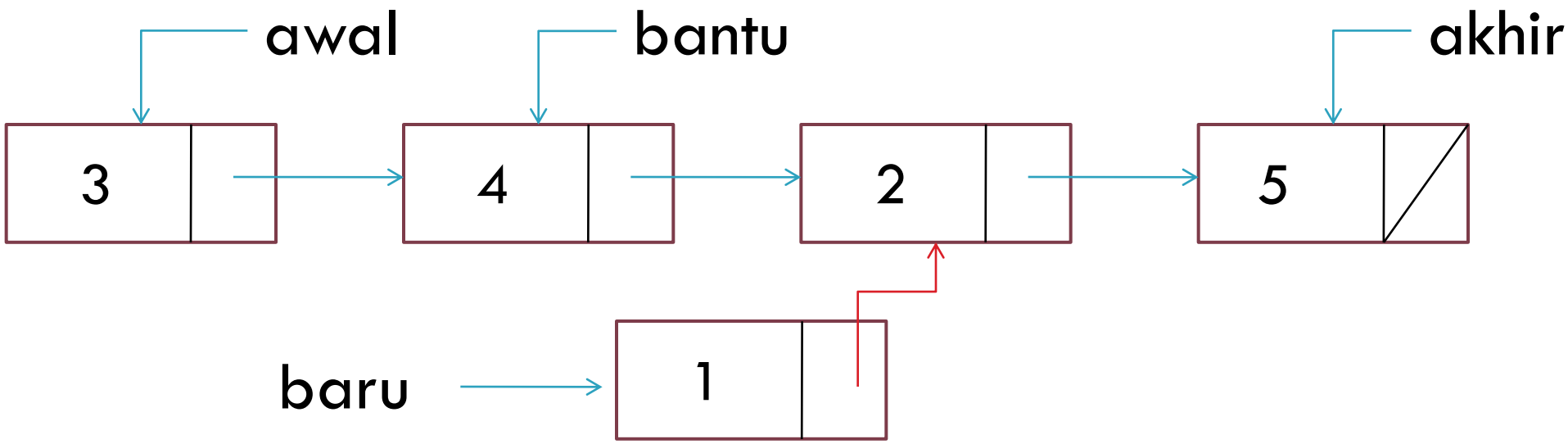
- List tidak kosong {Awal ≠ Nil}



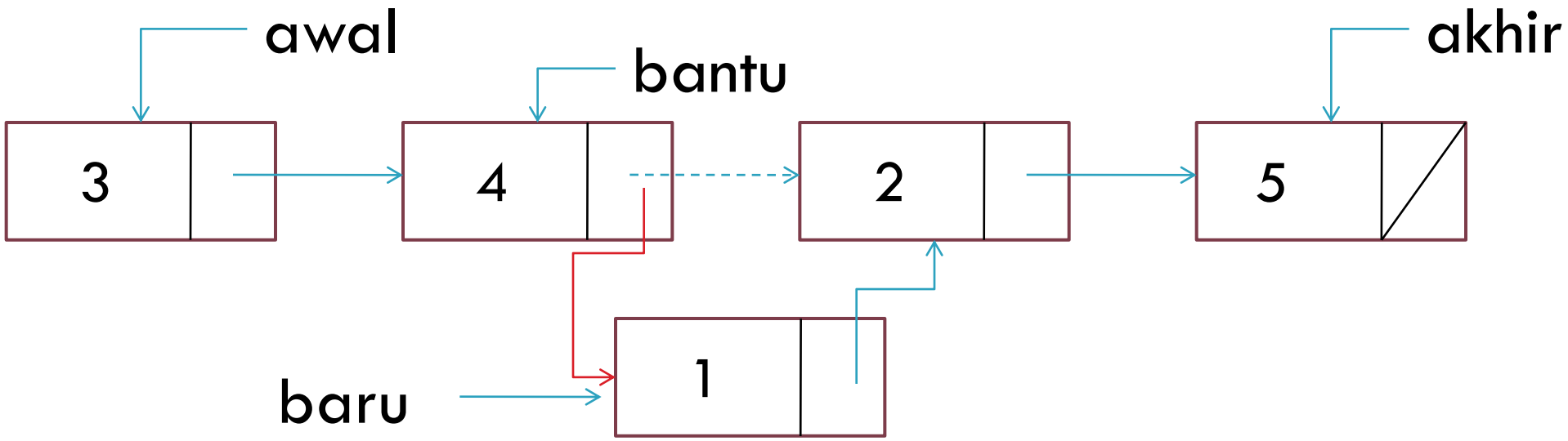
Misal akan menyisipkan angka 1 setelah angka 4

Penyisipan di Tengah (lanjutan)

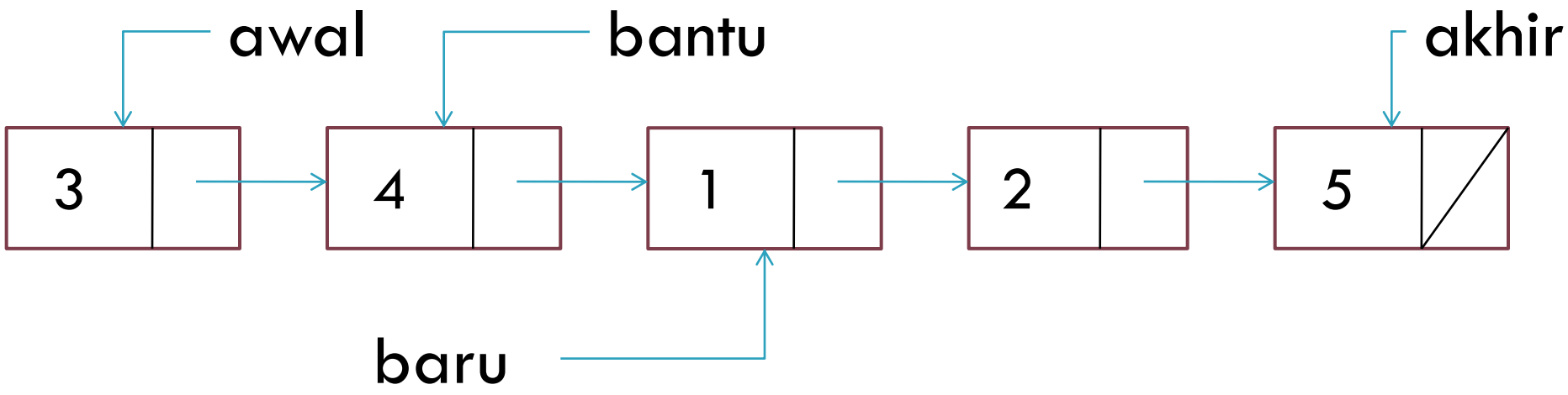
Angka 4 ditemukan dengan cara mencari mulai dari simpul pertama sampai angka 4 ditemukan (**metode sequential search**)



Penyisipan di Tengah (lanjutan)



Penyisipan di Tengah (lanjutan)



Algoritma Penyisipan di Tengah

Procedure SisipTengahSingle(Input elemen : tipedata, I/O awal, akhir : nama_pointer)

{I.S. : data yang akan disisipkan (elemen), pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}

{F.S. : menghasilkan satu simpul yang disisipkan di tengah pada single linked list}

Kamus :

baru,bantu : nama_pointer

ketemu : boolean

datasisip : tipedata

Algoritma :

If (awal = nil)

Then

Alloc(baru)

baru↑.info ← elemen

baru↑.next ← nil

Algoritma Penyisipan di Tengah (lanjutan)

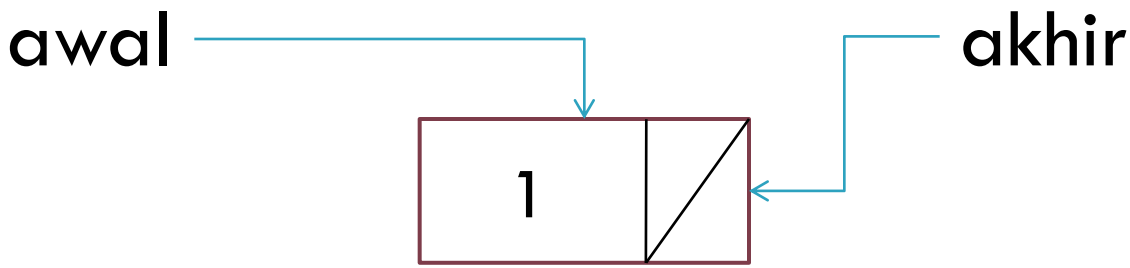
```
        awal ← baru
        akhir ← baru
Else
        Input(datasisip)
        bantu ← awal
        ketemu ← false
        While (not ketemu and bantu ≠ nil) do
            If (datasisip = bantu↑.info)
            Then
                    ketemu ← true
            Else
                    bantu ← bantu↑.next
            EndIf
        EndWhile
```

Algoritma Penyisipan di Tengah (lanjutan)

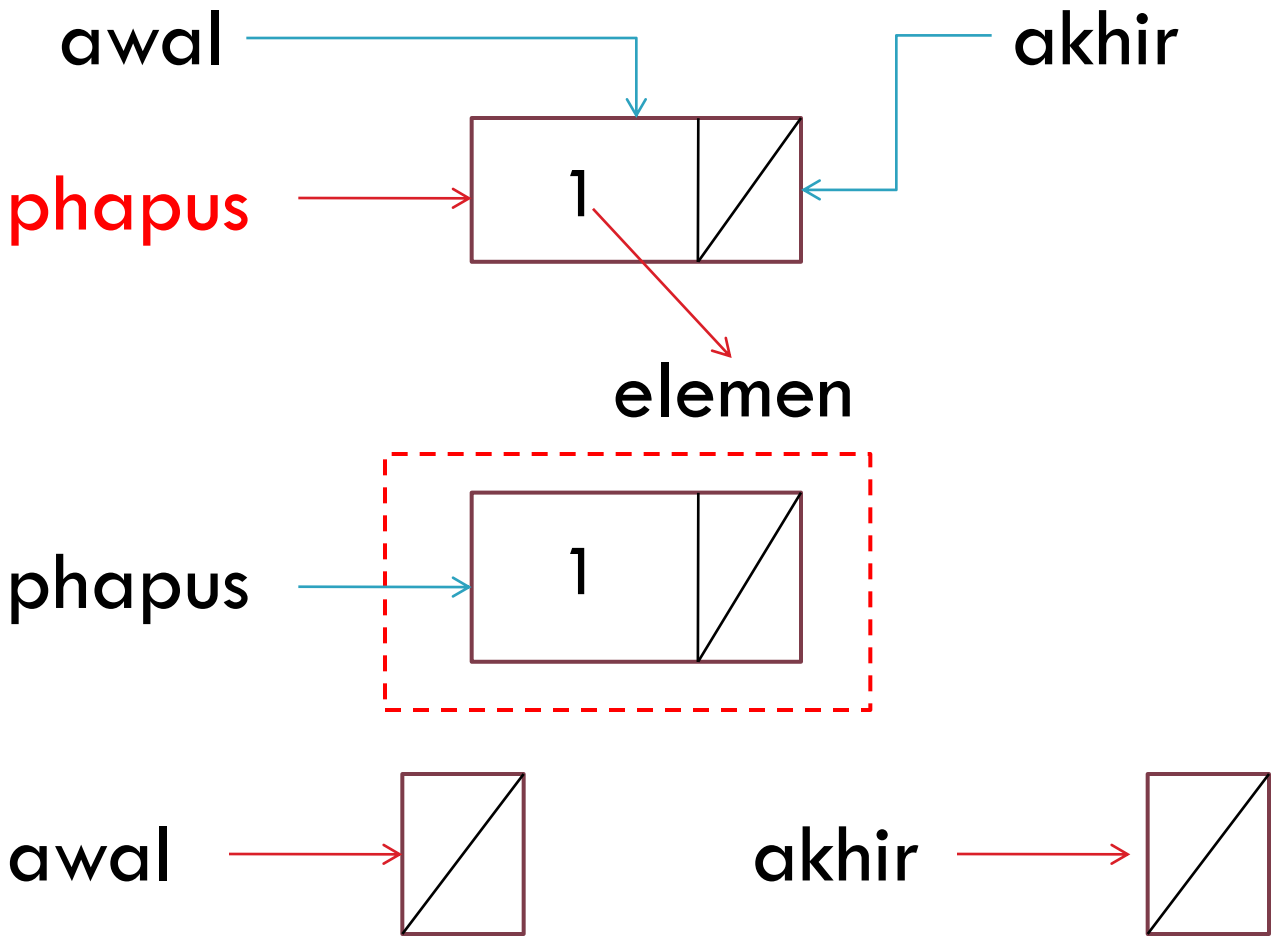
```
        If (ketemu)
        Then
            Alloc(baru)
            baru↑.info ← elemen
            If (bantu = akhir)
            Then
                sisip_belakang_single(elemen,awal,akhir)
            Else
                baru↑.next ← bantu↑.next
                bantu↑.next ← baru
            EndIf
        Else
            Output("Data yang akan disisipkan tidak ada");
        EndIf
    EndIf
EndProcedure
```

Penghapusan di Depan

- **Satu Simpul {awal = akhir}**

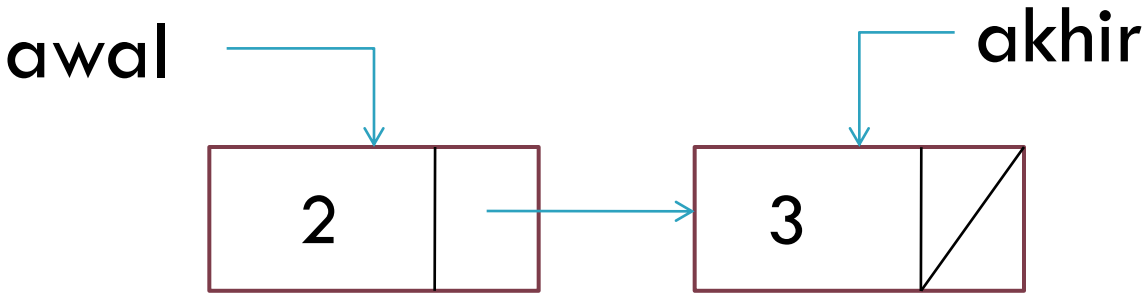


Penghapusan di Depan (lanjutan)

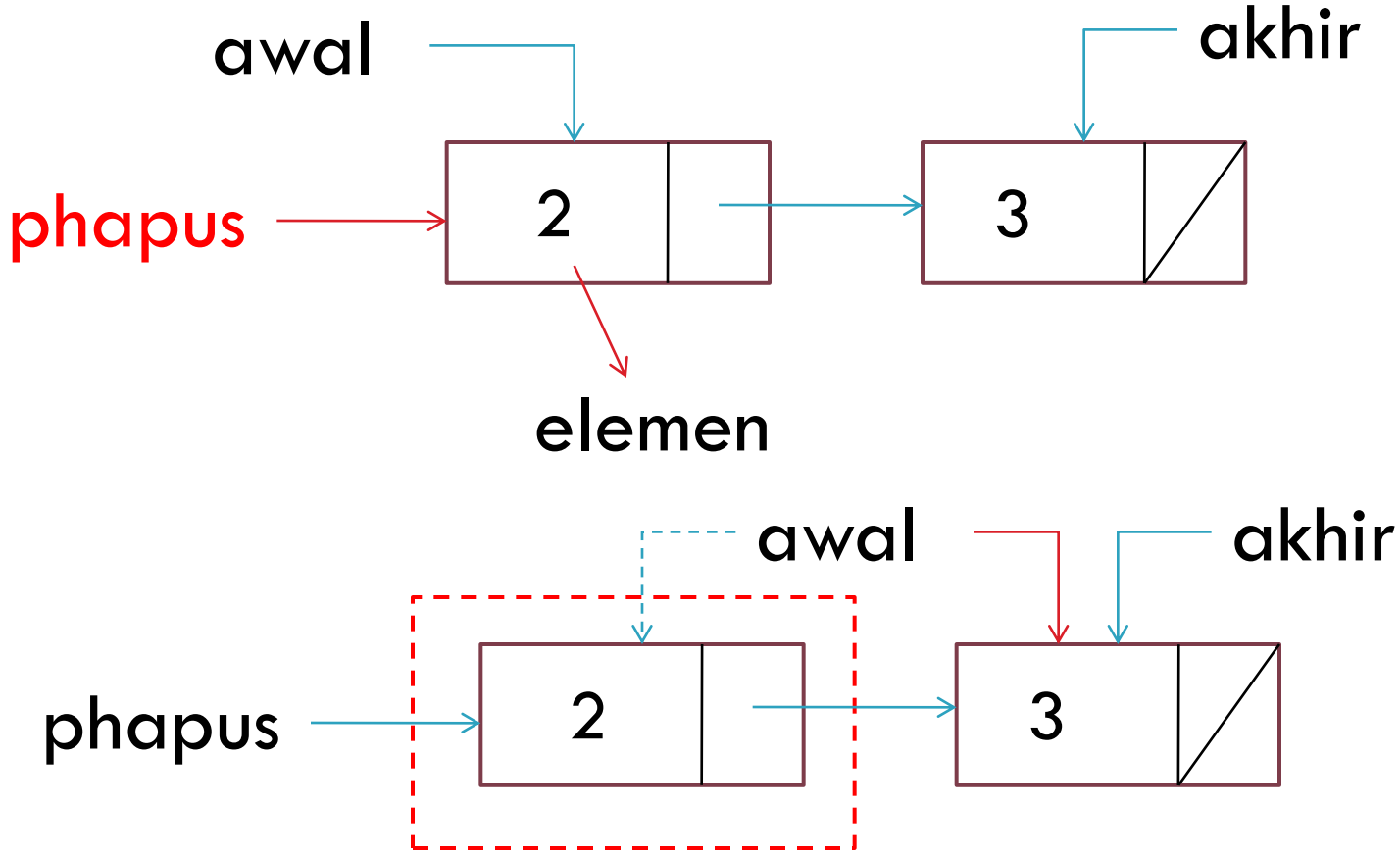


Penghapusan di Depan (lanjutan)

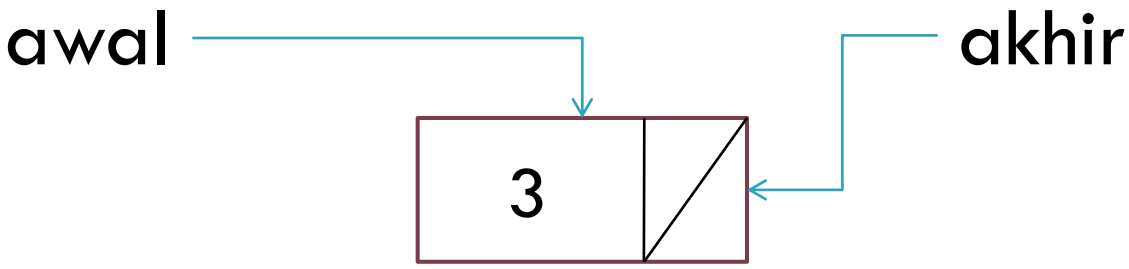
- **Lebih dari satu simpul {awal ≠ akhir}**



Penghapusan di Depan (lanjutan)



Penghapusan di Depan (lanjutan)



Algoritma Penghapusan di Depan

Procedure HapusDepanSingle(Output elemen : tipedata, I/O awal, akhir : nama_pointer)

{I.S. : pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}

{F.S. : menghasilkan single linked list yang sudah dihapus satu simpul di depan}

Kamus :

    phapus : nama_pointer

Algoritma :

    phapus ← awal

    elemen ← baru↑.info

    If (awal = akhir)

      Then

            awal ← nil

            akhir ← nil

Algoritma Penghapusan di Depan

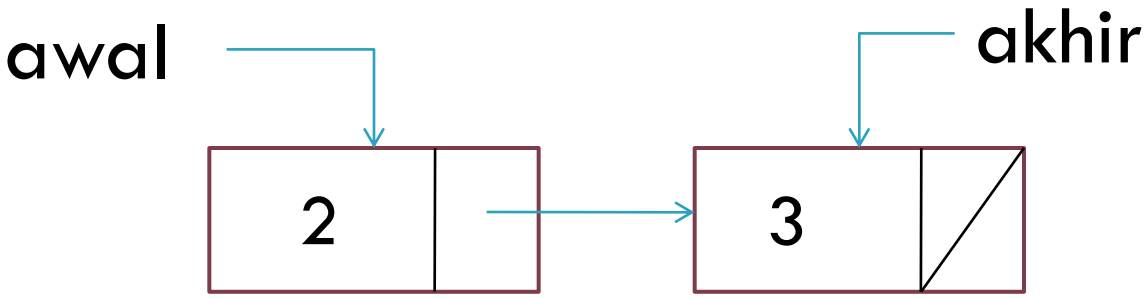

```
    Else
        awal← awal ↑.next
    EndIf
    Dealloc(phapus)

EndProcedure
```

Penghapusan di Belakang

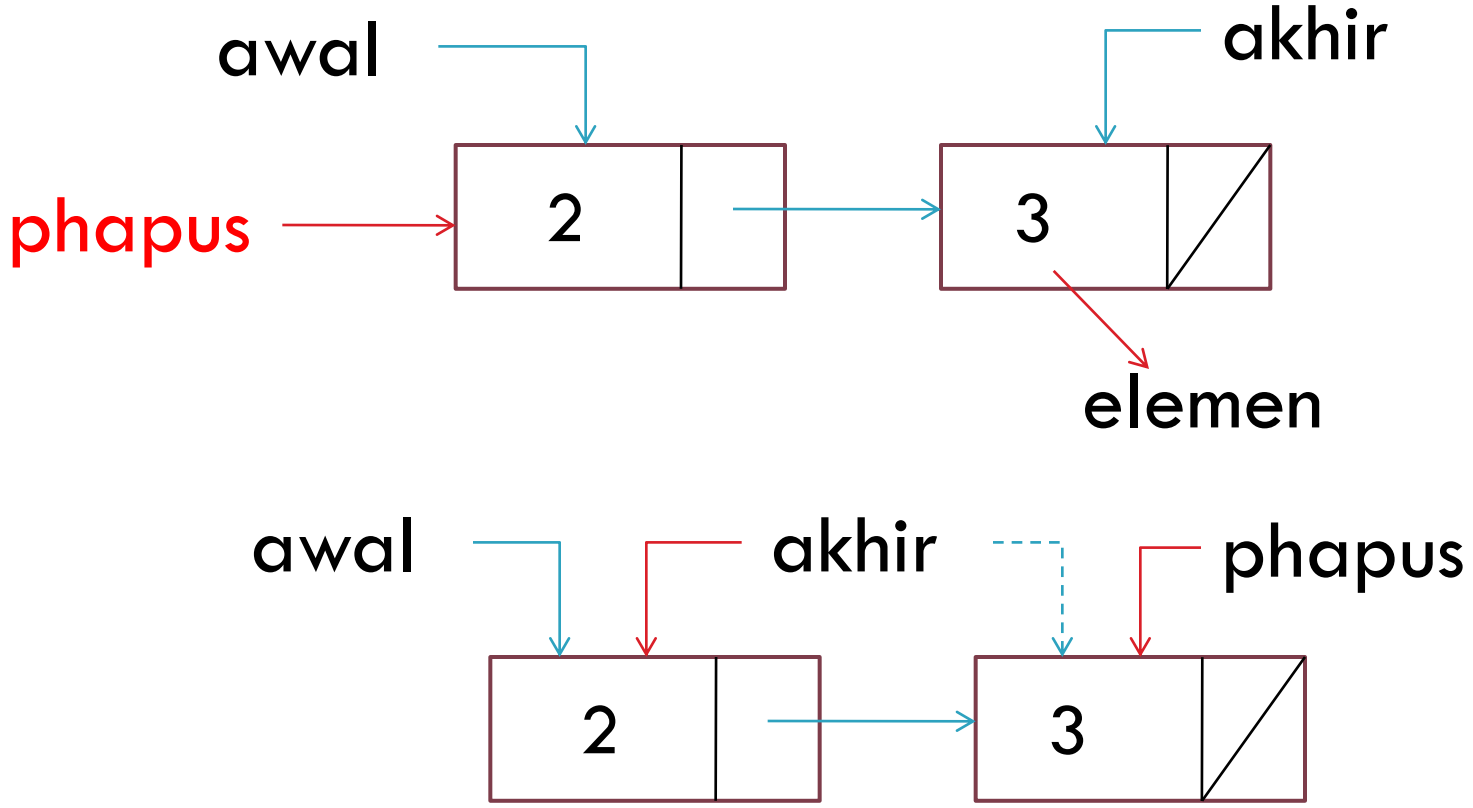
- Satu Simpul {awal = akhir}

{sama seperti penghapusan di depan}

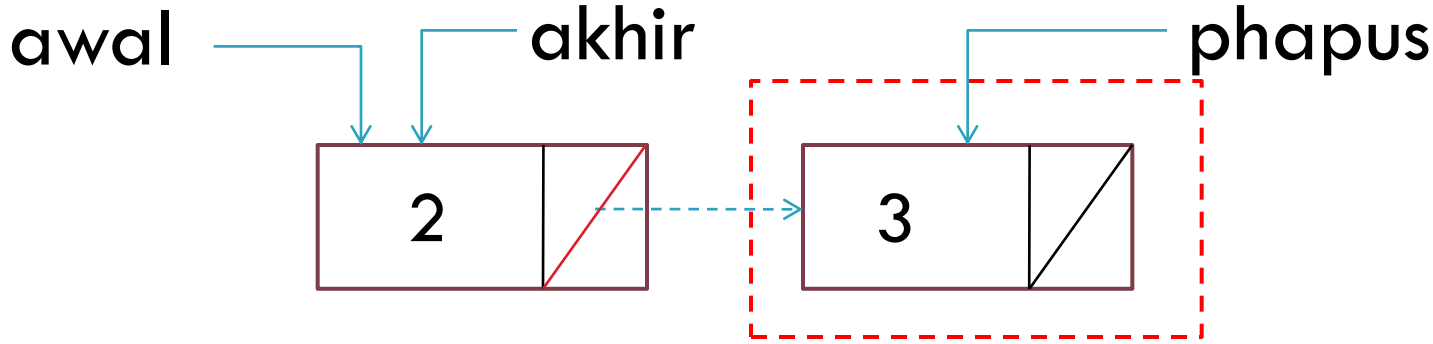
- Lebih dari satu simpul {awal ≠ akhir}

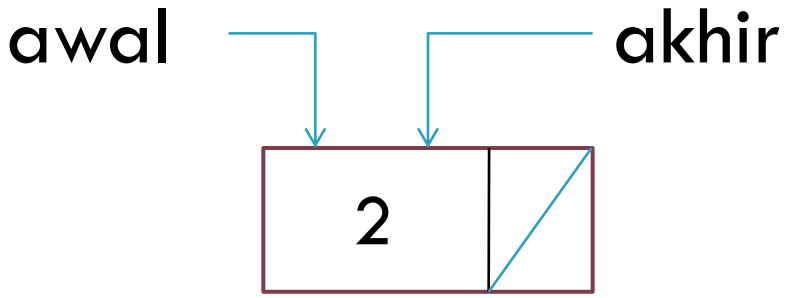


Penghapusan di Belakang (lanjutan)



Penghapusan di Belakang (lanjutan)





Algoritma Penghapusan di Belakang


```
Procedure  HapusBelakangSingle(Output elemen : tipedata, I/O  awal, akhir : nama_pointer)
{I.S. :  pointer penunjuk awal dan pointer penunjuk akhir sudah terdifinisi}
{F.S. : menghasilkan single linked list yang sudah dihapus satu simpul di belakang}
Kamus :
    phapus : nama_pointer
Algoritma :
    phapus ← awal
    elemen ← baru↑.info
    If (awal = akhir)
     Then
          awal ← nil
          akhir ← nil
```

Algoritma Penghapusan di Belakang

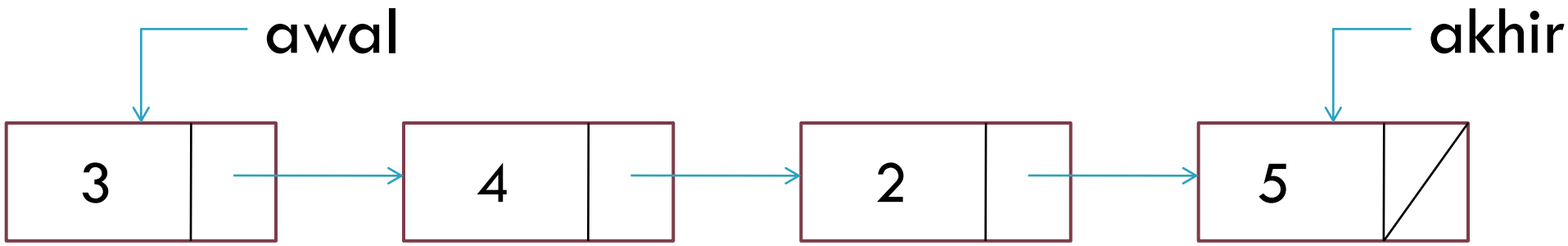
```
    Else
        while (phapus↑.next ≠ akhir) do
            phapus ← phapus↑.next
        endwhile
        akhir ← phapus
        phapus ← phapus↑.next
        akhir↑.next ← NULL
    EndIf
    Dealloc(phapus)

EndProcedure
```

Penghapusan di Tengah

- **Satu Simpul {awal = akhir}**

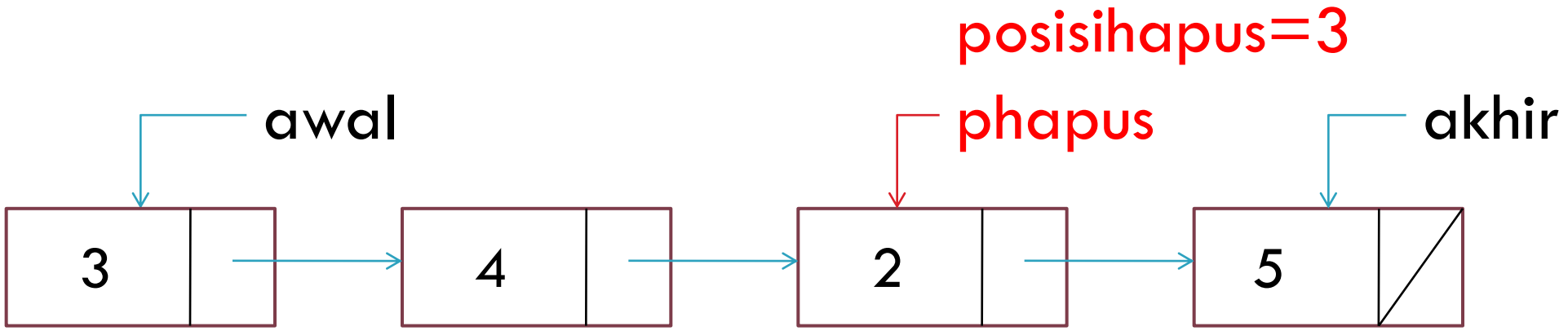
{sama seperti penghapusan di depan}

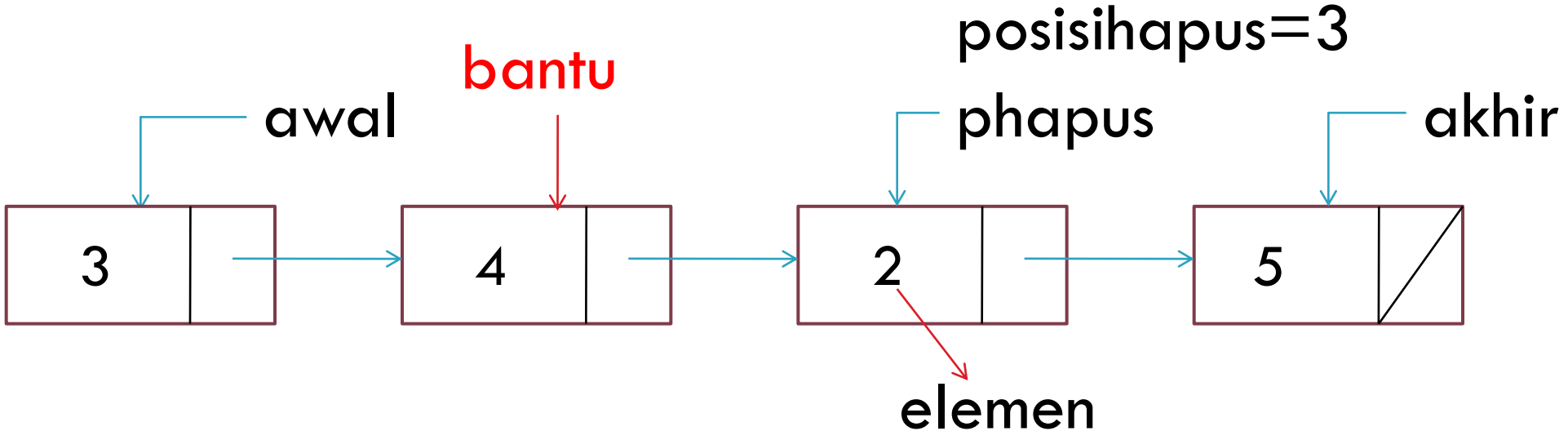
- **Lebih dari satu simpul {awal ≠ akhir}**



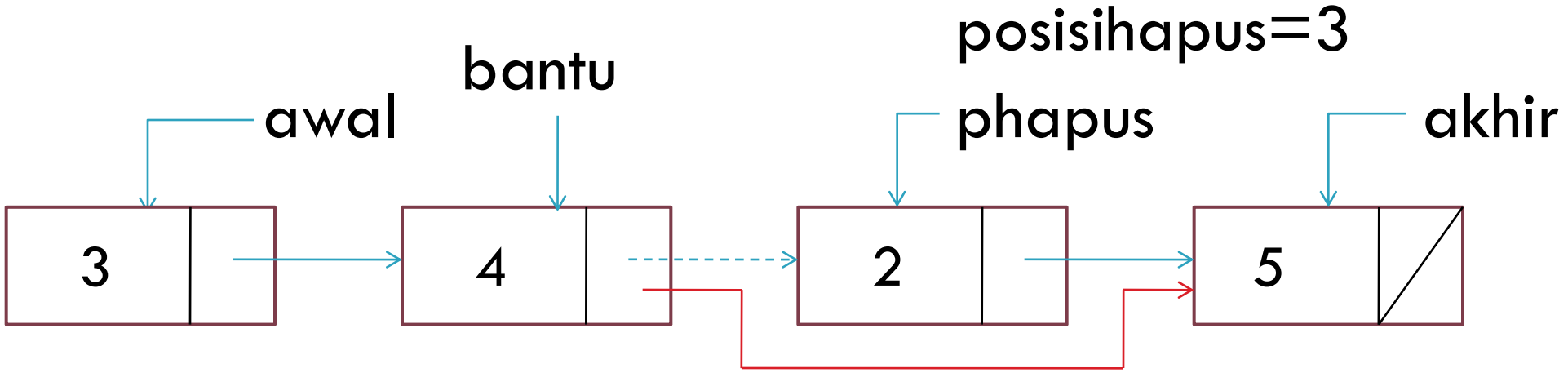
Misalkan simpul yang akan dihapus simpul ke 2

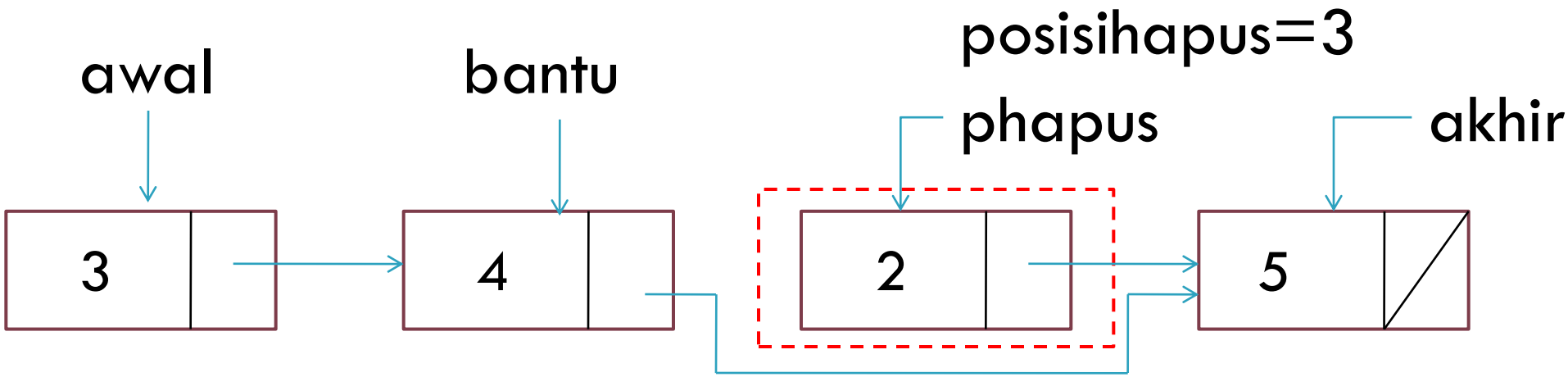
Penghapusan di Tengah (lanjutan)



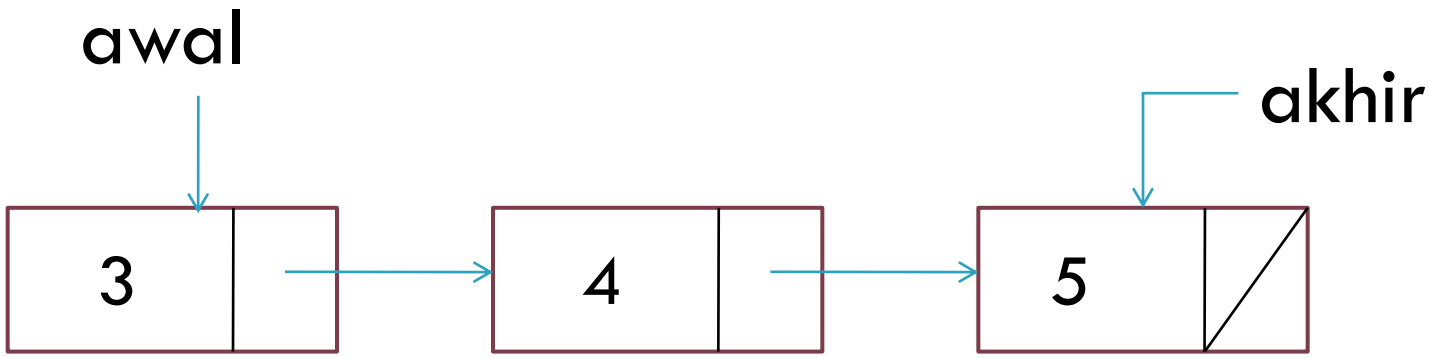


Penghapusan di Tengah (lanjutan)





Penghapusan di Tengah (lanjutan)



Algoritma Penghapusan di Tengah

Tugas buat algoritma dan programnya !