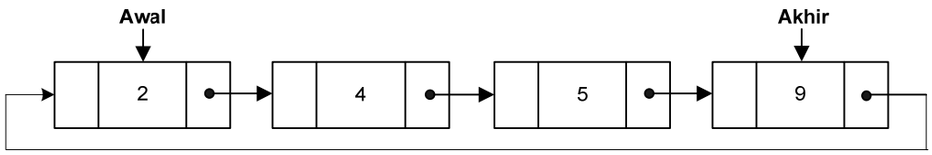
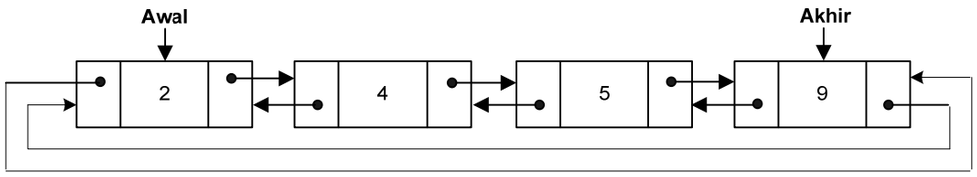
III. Circular Linked List

Circular Linked List adalah suatu linked list yang tidak memiliki nilai nil/NULL untuk medan sambungannya. Perhatikan Gambar 3.1 dan Gambar 3.2.



Gambar 3.1. Circular Single Linked List



Gambar 3.2. Circular Double Linked List

Deklarasi bisa dilihat kembali di Single Linked List atau Double Linked List.

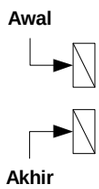
III.1 Circular Single Linked List

Operasi-operasi yang ada pada Circular Single Linked List hamper sama seperti pada Single Linked List yang telah dibahas sebelumnya.

Operasi-operasinya sebagai berikut:

1. Penciptaan

Penciptaan adalah memberikan nilai nil terhadap variabel pointer awal dan variabel pointer akhir.



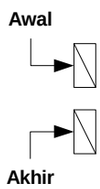
2. Penyisipan

2.a <u>Penyisipan di depan/awal</u>

Operasi ini berguna untuk menambahkan satu simpul baru di posisi pertama. Langkah pertama untuk penambahan data adalah pembuatan simpul baru dan mengisinya dengan data pada field info-nya. Pointer yang menunjuk ke simpul

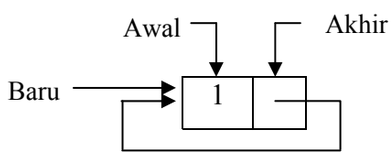
tersebut dipanggil dengan nama **baru**. Kondisi ketika akan menyisipkan simpul baru:

- **List kosong {Awal=nil}**
  - Kondisi sebelum disisipkan



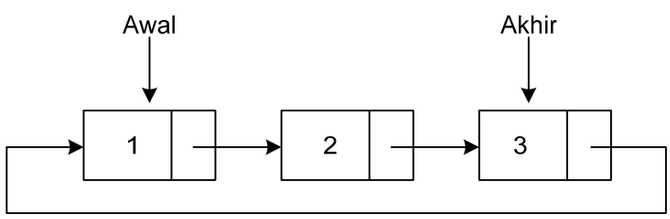
  - Kondisi setelah operasi penyisipan

    Operasi penyisipan pertama kali ketika linked list masih kosong adalah dengan mengisikan alamat pointer baru ke pointer awal dan pointer akhir. Dan medan sambungan kanan(next) menunjuk ke simpul itu sendiri, seperti gambar di bawah ini:

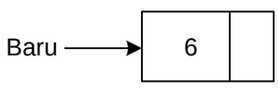


- **List tidak kosong {Awal ≠ Nil}**

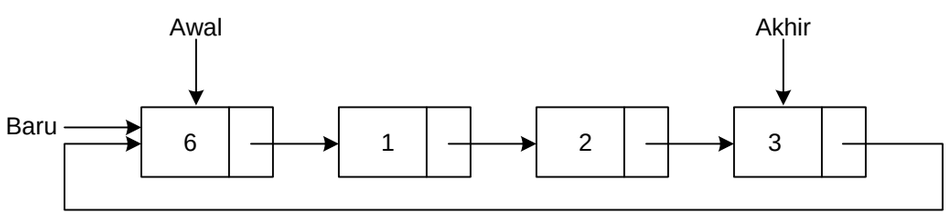
  Keadaan list mula-mula:



  Simpul yang akan disisipkan:



  Maka keadaan list setelah penyisipan di depan sebagai berikut:



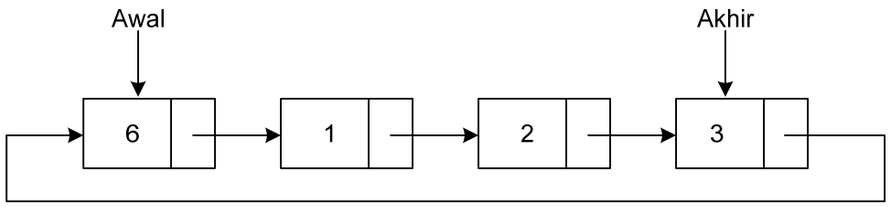
2.b. <u>Penyisipan di tengah</u>

Proses penyisipan di tengah sama seperti pada single linked list (lihat modul sebelumnya), karena tidak mengubah struktur circularnya

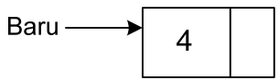
- List Kosong {Awal = Nil} (Sama dengan penyisipan di depan)

- List tidak kosong {Awal ≠ Nil}

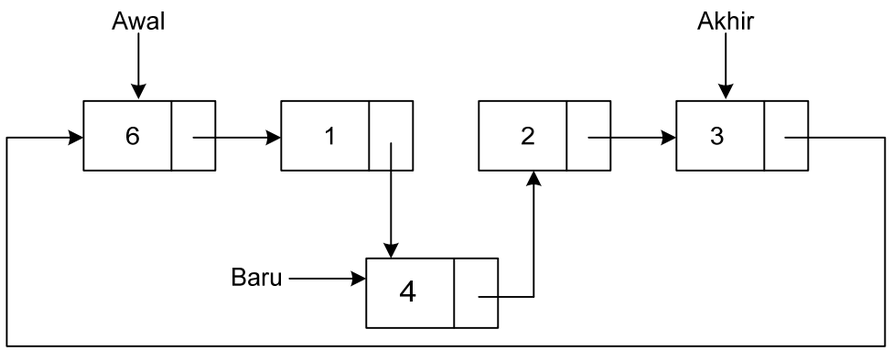
  Mula-mula keadaan list sebagai berikut:



Akan menyisipkan angka 4 setelah angka 1



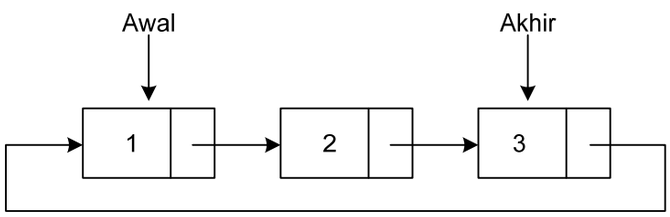
Keadaan list setelah penyisipan di tengah sebagai berikut:



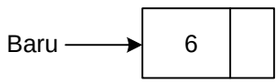
2.c <u>Penyisipan di akhir</u>

- List kosong {Awal = Nil} (Sama dengan penyisipan di depan)
- List tidak kosong (Awal ≠ Nil}

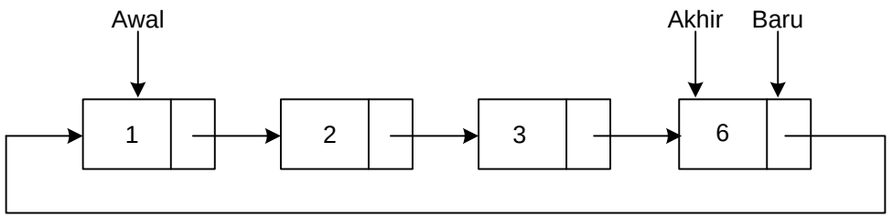
  Mula-mula keadaan list sebagai berikut:



Simpul yang akan disisipkan:



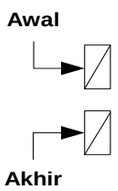
Maka keadaan list setelah penyisipan di depan sebagai berikut:



3. Penghapusan

Pada dasarnya proses penghapusan di circular sama seperti pada linear linked list. Ada 3 kondisi yang perlu diperhatikan yaitu kondisi linked list masih kosong, kondisi linked list hanya memiliki satu data, dan kondisi linked list yang memiliki lebih dari satu data (satu elemen)

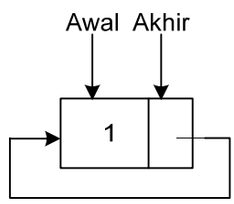
Kondisi linked list kosong (Pada kondisi ini proses penghapusan tidak bisa dilakukan, dan kondisi ini disimpan di luar dari prosedur ini)



3.a. Penghapusan di depan

Penghapusan data di awal adalah proses menghapus simpul pertama (yang ditunjuk oleh variabel pointer Awal), sehingga variabel pointer Awal akan berpindah ke simpul berikutnya, dan medan sambungan kanan (next) dari simpul yang ditunjuk oleh poiter akhir akan menunjuk ke simpul pertama yang baru (yang ditunjuk oleh pointer awal).

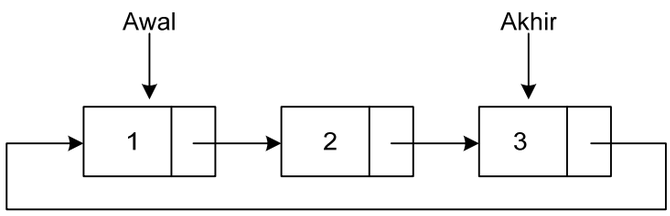
- **Kondisi linked list memiliki hanya satu data{Satu simpul}**



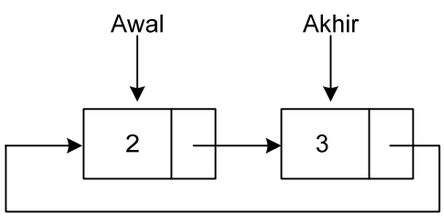
Ketika circular linked list hanya memiliki satu simpul, lalu ada proses penghapusan maka list akan kosong, dengan tahapan yang sama seperti di single linked list yang linear.

- **Kondisi linked list memiliki lebih dari satu data {satu simpul}**

  Keadaan mula-mula circular single linked list sebagai berikut:



  Jika simpul pertama (yang ditunjuk oleh pointer awal) dihapus, maka circular single linked list menjadi seperti gambar di bawah ini:



3.b. Penghapusan di tengah

Proses penghapusan di tengah sama dengan single linked list (lihat modul sebelumnya), karena tidak mengubah struktur dari list, tapi kalau simpul yang akan dihapus ketemu di akhir, maka akan memanggil modul (prosedur) hapus di akhir/belakang.

3.c. Penghapusan di belakang/akhir

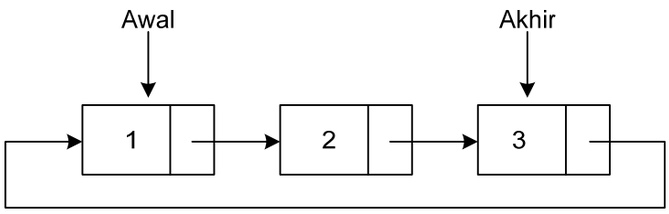
Penghapusan data di akhir adalah proses menghapus simpul terakhir (yang ditunjuk oleh variabel pointer Akhir), sehingga variabel pointer Akhir akan berpindah ke simpul sebelumnya, dan medan sambungan kanan (next) dari simpul yang ditunjuk oleh pointer Akhir yang baru akan menunjuk ke simpul pertama (yang ditunjuk pointer awal).

- **Kondisi linked list memiliki hanya satu data (satu simpul)**

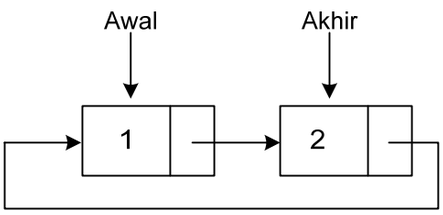
  Sama seperti proses penghapusan di depan/awal.

- **Kondisi linked list memiliki lebih dari satu data (satu simpul)**

  Keadaan mula-mula circular single linked list sebagai berikut:



Jika simpul terakhir (yang ditunjuk oleh pointer akhir) dihapus, maka circular single linked list menjadi seperti gambar di bawah ini:



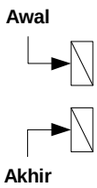
III.2. Circular Double Linked List

Operasi-operasi yang ada pada Circular Double Linked List hampir sama seperti pada Double Linked List yang telah dibahas sebelumnya.

Operasi-operasinya sebagai berikut:

1. Penciptaan

Penciptaan adalah memberikan nilai nil terhadap variabel pointer awal dan variabel pointer akhir.



2. Penyisipan

2.a <u>Penyisipan di depan/awal</u>

Operasi ini berguna untuk menambahkan satu simpul baru di posisi pertama. Langkah pertama untuk penambahan data adalah pembuatan simpul baru dan mengisinya dengan data pada field info-nya. Pointer yang menunjuk ke simpul tersebut dipanggil dengan nama **baru**.

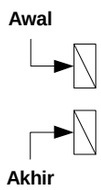
Ada 2 kondisi yang harus diperhatikan dalam penambahan data di awal yaitu :

a. Ketika linked list masih kosong

Kalau kondisi linked list masih kosong, maka simpul baru akan menjadi simpul awal dan sekaligus simpul akhir dari circular double linked list.

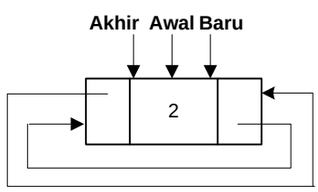
Perhatikan gambar di bawah ini :

- Kondisi sebelum disisipkan



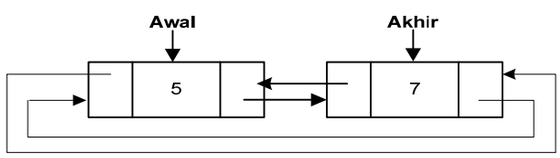
- Kondisi setelah operasi penyisipan

  Operasi penyisipan pertama kali ketika linked list masih kosong adalah dengan mengisikan alamat pointer baru ke pointer awal dan pointer akhir. Dan memberikan nilai nil pada medan sambungan kiri (prev) dan medan sambungan kanan(next). Lihat gambar di bawah ini:

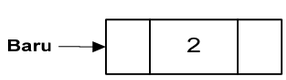


b. Ketika linked list tidak kosong

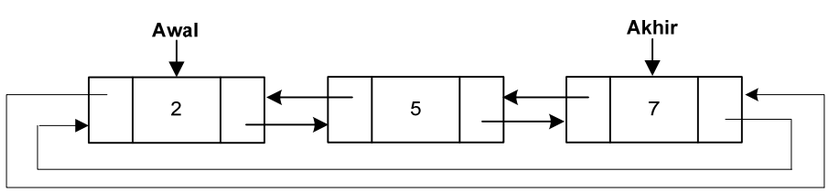
Misalnya mula-mula keadaan list sebagai berikut:



- Misalkan akan menyisipkan data 2



- Setelah simpul yang ada data 2 disisipkan di depan/awal, maka circular double linked list sebagai berikut:



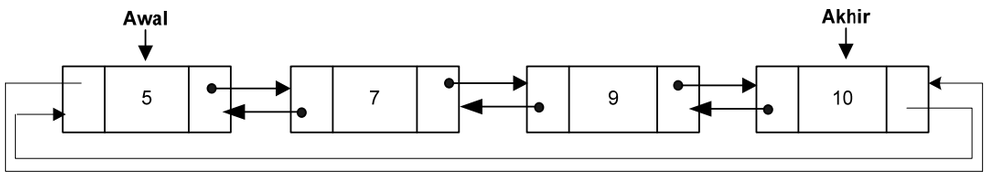
2.b. <u>Penyisipan di tengah</u>

Proses penyisipan di tengah sama seperti pada double linked list (lihat modul sebelumnya), karena tidak mengubah struktur circularnya.

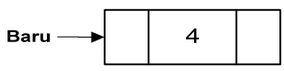
Untuk proses tersebut ada 2 hal yang harus diperhatikan yaitu :

- **Kondisi linked list masih kosong** prosesnya sama seperti penyisipan di depan/awal.
- **Kondisi linked list sudah mempunyai data (tidak kosong)**

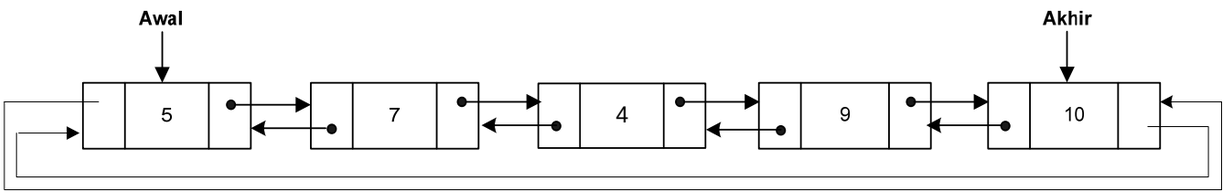
  Mula-mula keadaan linked list sebagai berikut:



Misalnya akan menyisipkan data 4 sebelum data 9 (untuk menyisipkan data setelahnya lihat kembali pada single linked list)



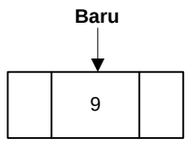
Maka bentuk circular double linked list setelah mengalami penyisipan di tengah seperti gambar berikut:



2.c. <u>Penyisipan di belakang/akhir</u>

Operasi ini berguna untuk menambahkan elemen baru di posisi akhir. Langkah pertama untuk penambahan data adalah pembuatan elemen baru dan pengisian nilai infonya. Pointer yang menunjuk ke data tersebut dipanggil dengan nama **baru**.

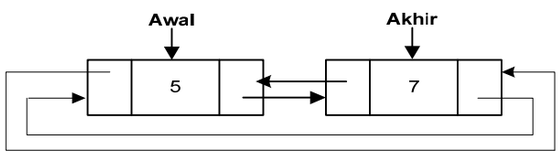
Misalkan akan menyisipkan data 9 di akhir.



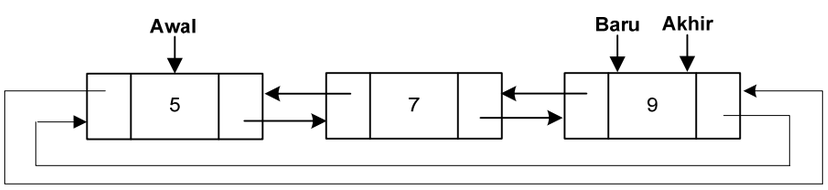
Ada 2 kondisi yang harus diperhatikan dalam penambahan data di akhir yaitu :

- **Kondisi linked list masih kosong** prosesnya sama seperti penyisipan di depan/awal.
- **Ketika linked list sudah mempunyai data**

  Mula-mula list sebagai berikut:



  Setelah data (simpul) baru disisipkan di akhir, maka gambar dari circular double linked list sebagai berikut:



3. Penghapusan

a. <u>Penghapusan di awal</u>

Operasi ini berguna untuk menghapus data pada posisi pertama. Ada 3 keadaan yang mungkin terjadi ketika akan melakukan proses hapus yaitu :

- **Kondisi linked list masih kosong**

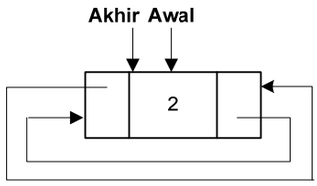
  Jika kondisi ini terjadi, maka proses penghapusan data tidak bisa dilakukan karena linked list masih kosong.

- **Kondisi linked list hanya memiliki 1 data/simpul**

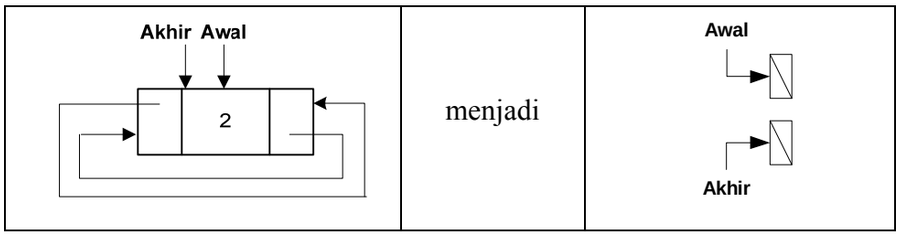
  Langkah yang perlu dilakukan ketika ingin melakukan proses penghapusan linked list yang memiliki hanya 1 data adalah dengan cara menempatkan dahulu satu pointer bantuan (phapus), kemudian medan data dari simpul yang akan dihapus disimpan ke dalam sebuah variabel (elemen), kemudian pointer awal dan akhir diberi harga nil dan simpul langsung dihapus.

Untuk lebih jelas perhatikan urutan penghapusannya di bawah ini :

- Kondisi list sebelum dihapus



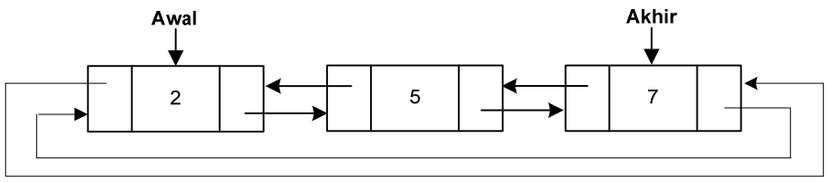
- Kondisi list setelah ada proses penghapusan menjadi:



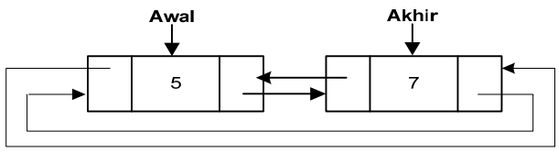
- **Kondisi linked list memiliki lebih dari satu data atau satu simpul**

  Untuk operasi penghapusan data di posisi pertama pada circular double linked list yang mempunyai data lebih dari satu buah adalah :

  Misalkan keadaan list mula-mula:



  Setelah terjadi penghapusan di awal/depan, maka keadaan list menjadi:



b. <u>Penghapusan di tengah</u>

Proses penghapusan di tengah sama dengan single linked list (lihat modul sebelumnya), karena tidak mengubah struktur dari list, tapi kalau simpul yang akan dihapus ketemu di akhir, maka akan memanggil modul (prosedur) hapus di akhir/belakang.

c. <u>Penghapusan di akhir</u>

Penghapusan data di akhir adalah proses menghapus simpul terakhir (yang ditunjuk oleh variabel pointer Akhir), sehingga variabel pointer Akhir akan berpindah ke simpul sebelumnyanya, dan medan sambungan kanan (next) dari simpul yang ditunjuk oleh pointer Akhir yang baru akan menunjuk ke simpul pertama (yang ditunjuk oleh pointer awal), dan medan sambungan kiri (prev) dari simpul yang ditunjuk oleh pointer awal menunjuk ke simpul terakhir (yang ditunjuk oleh pointer akhir).

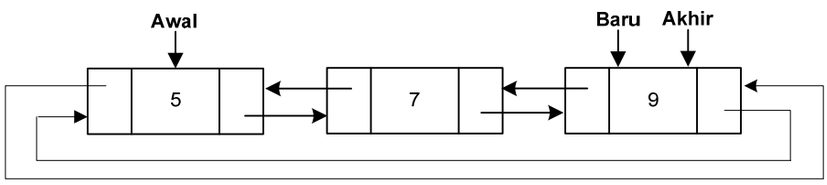
Ada 3 keadaan yang mungkin terjadi ketika akan melakukan proses hapus yaitu :

i. **Kondisi linked list hanya memiliki satu data atau satu simpul**

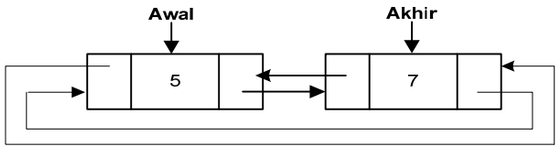
   Penghapusan di akhir prosesnya sama seperti penghapusan di depan

ii. **Kondisi linked list memiliki lebih dari satu data atau lebih dari 1 simpul**

   Misalkan mula-mula keadaan list sebagai berikut:



Setelah proses penghapusan di akhir, maka keadaan circular double linked list menjadi:



4. Penelusuran/traversal

Prosesnya secara umum sama seperti penelusuran pada single liked list atau double linked list (lihat modul sebelumnya).

5. Pencarian /Seaching

Prosesnya secara umum sama seperti penelusuran pada single liked list atau double linked list (lihat modul sebelumnya).

6. **Pengurutan/sorting**

   Prosesnya secara umum sama seperti penelusuran pada single liked list atau double linked list (lihat modul sebelumnya).

7. **Penghancuran/destroy**

   Proses penghancuran bisa dengan cara memanggil modul/subrutin penghapusan di awal atau penghapusan di akhir secara terus menerus sampai list kosong, atau dengan cara:

```
phapus ← awal
While (awal ≠ nil) do
    If (awal = akhir)
      Then
          awal ← nil
          akhir ← nil
      Else
          awal ← awal↑.next
          awal↑.prev ← akhir
          akhir↑.next ← awal
    EndIf
    dealloc(phapus)
    phapus ← awal
EndWhile
```