

STRUKTUR DATA LINKED LIST/SENARAI BERANTAI

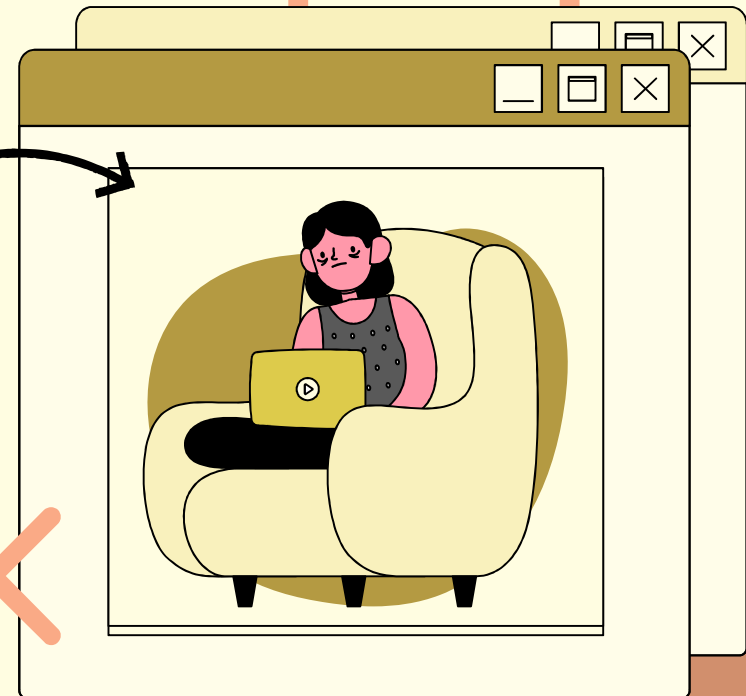
Nama Kelompok

Nama : Dani Fahdlu Rohman Silaen

NPM : 232310027

Nama : Hery Tua Sigalingging

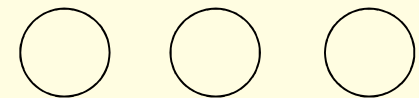
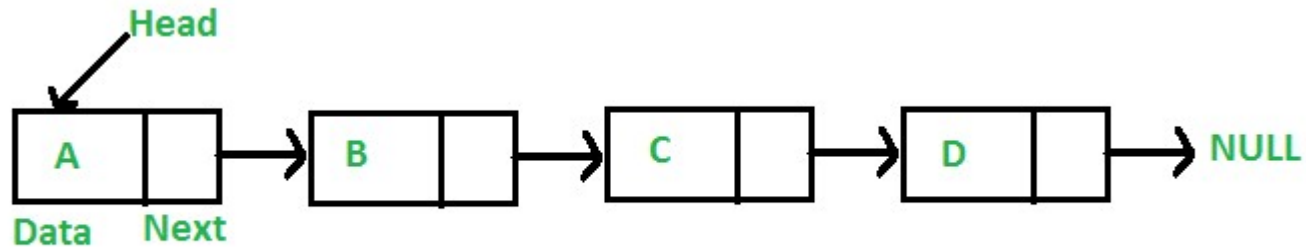
NPM : 232310030



LAB ALGORITMA
Kak Shania Oktaviani

PENGERTIAN LINKED LIST

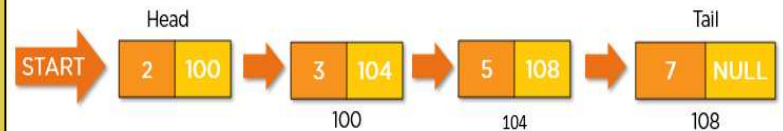
Linked list adalah struktur data linier berbentuk rantai simpul di mana setiap simpul menyimpan 2 item, yaitu **nilai data** dan **pointer ke simpul elemen berikutnya**. Berbeda dengan array, elemen linked list tidak ditempatkan dalam alamat memori yang berdekatan melainkan elemen ditautkan menggunakan pointer.



Jenis Jenis Linked List

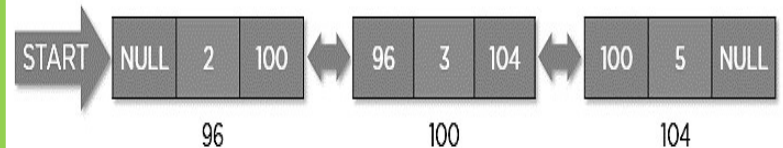
1. Singly linked list

Singly linked list adalah linked list unidirectional. Jadi, kita hanya dapat melintasinya dalam satu arah, yaitu dari simpul kepala ke simpul ekor.



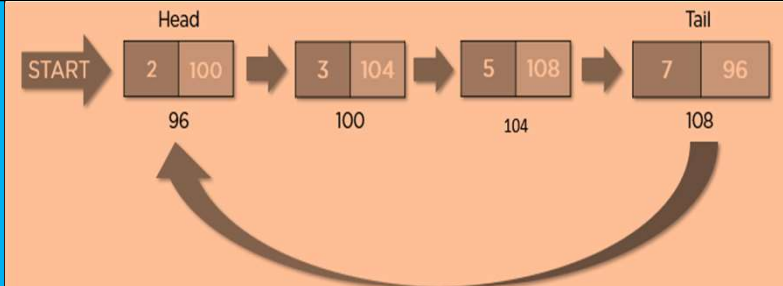
2. Doubly linked list

Doubly linked list adalah linked list bidirectional. Jadi, kita bisa melintasinya secara dua arah.



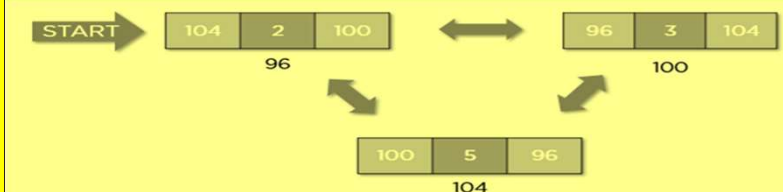
3. Circular linked list

Circular linked list adalah linked list unidirectional. Kita hanya dapat melintasinya dalam satu arah. Tetapi jenis linked list ini memiliki simpul terakhir yang menunjuk ke simpul kepala.



4. Circular doubly linked list

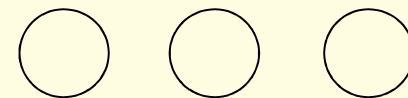
Circular doubly linked list adalah gabungan dari Doubly linked list dan Circular linked list.



Karakteristik Linked List








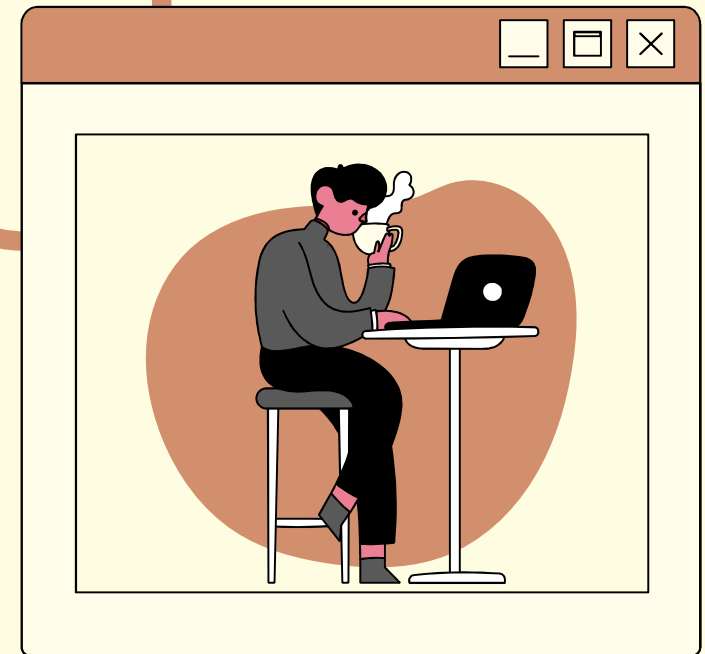
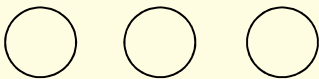
- Linked list menggunakan memori tambahan untuk menyimpan link (tautan)
- Untuk inisialisasi awal linked list, kita tidak perlu tahu ukuran dari elemen.
- Linked list umumnya dapat digunakan untuk mengimplementasikan struktur data lain seperti stack, queue, ataupun graf
- Simpul pertama dari linked list disebut sebagai Head.
- Pointer setelah simpul terakhir selalu bernilai NULL
- Dalam struktur data linked list, operasi penyisipan dan penghapusan dapat dilakukan dengan mudah
- Tiap-tiap simpul dari linked list berisi pointer atau tautan yang menjadi alamat dari simpul berikutnya
- Linked list bisa menyusut atau bertambah kapan saja dengan mudah.

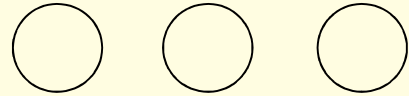


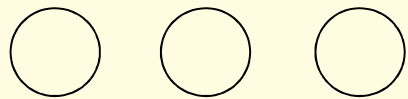
Operasi-operasi pada Linked List

Berikut adalah daftar operasi dasar pada linked list :

-  **Traversal** - mengakses setiap elemen dari linked list
-  **Insertion** - menambahkan elemen baru ke linked list
-  **Deletion** - menghapus elemen yang ada
-  **Searching** - menemukan simpul pada linked list
-  **Sorting** - mengurutkan simpul dari struktur linked list

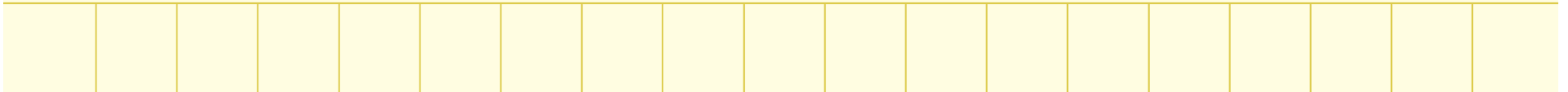






Kelebihan Linked List

- 01 Struktur data dinamis:** Linked list adalah himpunan dinamis sehingga dapat bertambah dan menyusut saat runtime dengan mengalokasikan dan membatalkan alokasi memori. Jadi kita tidak perlu memberikan ukuran awal dari linked list.
- 02 Tidak boros memori:** Dalam linked list, pemanfaatan memori yang efisien dapat dicapai karena ukuran linked list bertambah atau berkurang pada runtime sehingga tidak ada pemborosan memori dan tidak perlu mengalokasikan memori sebelumnya.
- 03 Implementasi:** Struktur data linier seperti stack dan queue seringkali mudah diimplementasikan menggunakan linked list.
- 04 Operasi penyisipan dan penghapusan:** Operasi penyisipan dan penghapusan cukup mudah dalam linked list. Kita tidak perlu menggeser elemen setelah operasi penyisipan atau penghapusan elemen, hanya alamat yang ada di pointer berikutnya saja yang perlu diperbarui.



Kelemahan Linked List

- **Penggunaan memori:** Linked list memerlukan lebih banyak memori dibandingkan dengan array. Karena dalam linked list, pointer juga perlu menyimpan alamat elemen berikutnya dan membutuhkan memori tambahan untuk dirinya sendiri.
- **Traversal:** Dalam traversal, linked list lebih banyak memakan waktu dibandingkan dengan array. Akses langsung ke elemen tidak bisa dilakukan pada linked list seperti array yang dapat akses elemen berdasarkan indeks. Untuk mengakses sebuah simpul pada posisi n dari linked list, kita harus melintasi semua simpul sebelumnya.
- **Reverse Traversing:** Dalam single linked list, reverse traversing tidak dimungkinkan, tetapi dalam kasus double-linked list, ini dapat dimungkinkan karena berisi pointer ke node yang terhubung sebelumnya dengan setiap node. Untuk melakukannya, diperlukan memori tambahan untuk pointer sebelumnya sehingga ada pemborosan memori.
- **Akses Acak:** Akses acak tidak bisa dilakukan dalam linked list karena alokasi memorinya yang dinamis.

Linked Lists in 4

THANKS

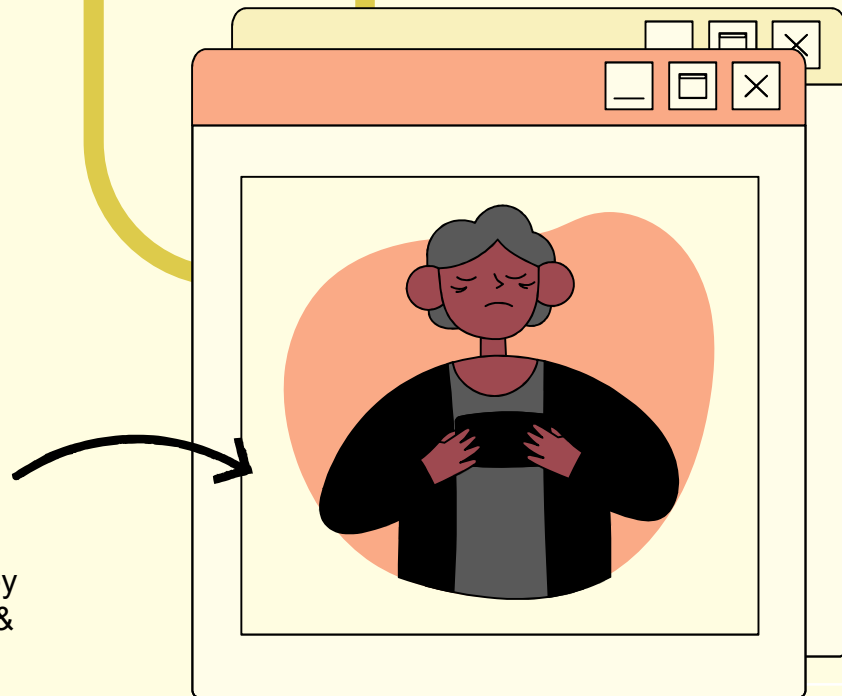
Do you have any questions?

725silaen@gmail.com

083811870999



CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik** and illustrations by **Storyset**



Pak kata gua teh

