LINGKED LIST

Linked list dibagi menjadi 4 macam

1. Singly linked list
2. Doubly Linked list
3. Circular Linked LIst (circular = tidak ada ujungnya
4. Circular Doubly Linked List

Apa itu Linked List

* Link list adalah element yang berurutan yang berhubungan dengan pointer / penunjuk
* Elemen terakhir menunjukan ke null / kosong (untuk linked list non cicular)
* Elemen pada singel linked list dapat bertambah atau berkurang  (dinamis) selama program di jalankan
* Dapat di buat selama di perlukan (hingga memory sistem abis)
* Lingked list tidak membuang uang memoty (tetapi mebutuhkan memori ekstra yntuk pointer)
* Model Data senkuensial yang saling berkaitan
* Mirip dengan array, tetapi lebih fleksibel dan sedikit kompleks
* Simpul pertama dari linked list adalah Head dan yang terakhir adalah Nul

Di sini kelemahan array = kurang fleksibel dalam alokasi memory. Misalkan sudah mendeklarasi suatu array maka maksimal array hanya sampai angka yang di deklarasikan

 Dan sedangkan Linked List  secara model mirip seperti array, ibarat memiliki banyak kamar yang dapat menyimpan data yang banyak dengan variabel tunggal Dan bersifat fleksibel = dapat menambah alokasi penyimpanan linked list, apabila ada penambahan data bisa di lakukan di linked list dan dapat menambah kapasitas di data ini.

Lingked list  bersifat Sekuensial (bertahap / gabisa loncat), Lingked list bisa di artikan adalah sebuah List yang saling berkaitan (linked list)

Linked List l terdari sebuah element/node yang Tiap elemen terdiri dari 2 bagisan yaitu nilai dan address

Misalkan node yang pertama terkoneksi dengan node ke 2 yang terdapat nilai  dan address dan sampai seterusnya hingga node tersebut berhubungan dengan node terakhir (last) , element yang paling belakang biasanya di bagian adress di kasih garis yang biasa di sebut null

Variabel / nama array linked list data yang bisa menympan banyak nilai, biasanya setiap nilai di simpan dalam sebuah node yang tehubung dengan node lainnya

Itu lah mengapa linkedlist di sebut data sekuensial atau berurutan, Ini mirip sekali dengan array tapi disini setiap penyimpanan harus berhubung dengan penimpanan lainnya.

Linked List tidak memiliki indeks element / no urut, cara mengaksesnya itu harus di cek satu persatu dari node paling depan, dan apabila mau menambahkan data tinggal menambahkan node nya kembali.

Perbedaan linked list dan array

* Linked list tidak memiliki indeks element / no urut dan sedangkan array punya.
* Lingked List membutuhkan tipe data pointer dalam pengoprasiannya yang berfungsi pengganti indeks yang sifatnya sekuensial dan sedangkan array tidak membutuhkan pointer dikarenakan sudah mempunyai indeks element.

      - linkedlist sangat cocok untuk menyimpan data yang jumlahnya tidak menentu (dinamis)

      - Linked List Sangat menghemat Memory

Konsep pointer=

Pointer adalahh tipe data yang berfungsi  untuk membaca atau menyimpan alamat suatu variabel

Biasanya variabel pointer di beri tanda asterisk (\*)

Contoh =

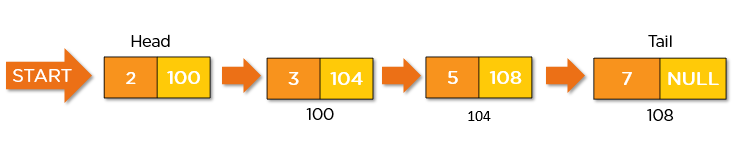
char\*ptr; —---- char pointer to carracter untuk penunjuk alamat memory

int\*indeks; —--------- pointer to integer

float\*data; —------ pointer to float

1. SINGLY LINGKED LIST

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



1. DOUBLY LINKED LIST

Doubly linked list adalah linked list bidirectional. Jadi, kita bisa melintasinya secara dua arah. Tidak seperti singly linked list, simpul doubly linked list berisi satu pointer tambahan yang disebut previous pointer. Pointer ini menunjuk ke simpul sebelumnya.

|  |
| --- |
| Ilustrasi Doubly linked list |

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. Jadi saat melintas, kita harus berhati-hati dan berhenti saat mengunjungi kembali simpul kepala.

|  |
| --- |
| Ilustrasi Circular linked list |

CIRCULAR DOUBLY LINKED LIST

Circular doubly linked list adalah gabungan dari Doubly linked list dan Circular linked list. Seperti Doubly linked list, linked list ini memiliki pointer tambahan yang disebut previous pointer, dan mirip dengan Circular linked list, simpul terakhirnya menunjuk pada simpul kepala. Jenis linked list ini adalah bidirectional. Jadi, kita bisa melintasinya dua arah.

|  |
| --- |
| Ilustrasi Circular doubly linked list |

**Operasi-operasi pada Linked List**

Ada beberapa operasi yang bisa kita lakukan pada struktur data linked list. Misalnya, operasi insertion yaitu tindakan menambahkan elemen baru ke 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

**Fungsi dan Kegunaan Linked List**

Adapun fungsi dan kegunaan linked list adalah sebagai berikut:

* Linked list dapat digunakan untuk mengimplementasikan struktur data lain seperti [stack](https://www.trivusi.web.id/2022/07/struktur-data-stack.html), [queue](https://www.trivusi.web.id/2022/07/struktur-data-queue.html), graf, dll.
* Digunakan untuk melakukan operasi aritmatika pada bilangan long integer
* Dipakai untuk representasi matriks rongga.
* Digunakan dalam alokasi file yang ditautkan.
* Membantu dalam manajemen memori.

Penerapan linked list banyak ditemui dalam beberapa kasus berikut:

* Linked list digunakan dalam penjadwalan Round-Robin untuk melacak giliran dalam permainan multi-pemain.
* Digunakan dalam aplikasi penampil gambar. Gambar sebelumnya dan berikutnya ditautkan, sehingga dapat diakses oleh tombol prev dan next.
* Dalam playlist musik, lagu yang sedang diputar ditautkan ke lagu sebelumnya dan berikutnya.

**Kelebihan Linked List**

Berikut ini dalah keunggulan menggunakan linked list:

* **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.
* **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.
* **Implementasi**: Struktur data linier seperti stack dan queue seringkali mudah diimplementasikan menggunakan linked list.
* **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**

Adapun kelemahan menggunakan linked list adalah sebagai berikut:

* **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.

Program ini akan mencakup pembuatan linked list, penambahan elemen, penghapusan elemen, dan pencarian elemen.

**1. Membuat Struktur Node**

Struktur node adalah unit dasar dari linked list. Setiap node memiliki dua komponen utama: data dan pointer yang menunjuk ke node berikutnya. Berikut adalah contoh kode untuk membuat struktur node:

struct Node {

int data;

Node\* next;

};

**2. Membuat Linked List Kosong**

Selanjutnya, kita perlu membuat linked list kosong dengan menggunakan pointer kepala (head). Pointer ini akan menunjuk ke elemen pertama dalam linked list. Berikut adalah contoh kode untuk membuat linked list kosong:

Node\* head = NULL;

**3. Menambahkan Elemen ke Linked List**

Untuk menambahkan elemen baru ke linked list, kita perlu melakukan alokasi memori untuk node baru dan mengatur nilainya. Selanjutnya, kita mengatur pointer node sebelumnya (jika ada) agar menunjuk ke node baru, dan node baru menunjuk ke node berikutnya. Berikut adalah contoh kode untuk menambahkan elemen ke linked list:

void tambahElemen(int newData) {

Node\* newNode = new Node;

newNode->data = newData;

newNode->next = head;

head = newNode;

}

**4. Menghapus Elemen dari Linked List**

Untuk menghapus elemen dari linked list, kita perlu mencari elemen yang ingin dihapus dan menyesuaikan pointer node sebelumnya agar menunjuk ke node berikutnya. Selanjutnya, kita membebaskan memori yang dialokasikan untuk node yang dihapus. Berikut adalah contoh kode untuk menghapus elemen dari linked list:

void hapusElemen(int deleteData) {

Node\* current = head;

Node\* previous = NULL;

while (current != NULL && current->data != deleteData) {

previous = current;

current = current->next;

}

if (current == NULL) {

return; // Elemen tidak ditemukan

}

if (previous == NULL) {

head = current->next; // Menghapus elemen pertama

} else {

previous->next = current->next; // Menghapus elemen tengah atau terakhir

}

delete current; // Menghapus node

}

**5. Mencari Elemen dalam Linked List**

Untuk mencari elemen dalam linked list, kita perlu membandingkan nilai setiap node dengan nilai yang dicari

. Jika ditemukan, kita dapat mengembalikan node yang sesuai atau memberikan indikasi bahwa elemen tidak ditemukan. Berikut adalah contoh kode untuk mencari elemen dalam linked list:

Node\* cariElemen(int searchData) {

Node\* current = head;

while (current != NULL) {

if (current->data == searchData) {

return current; // Elemen ditemukan

}

current = current->next;

}

return NULL; // Elemen tidak ditemukan

REFERENSI

<https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html>

<https://pemburukode.com/contoh-program-linked-list-cpp-sederhana/>

<https://youtu.be/VunTar7dl-0?si=HLGm6-JXTDzRGWqT>