Michael Johanes Johansyah

L0123081

Informatika C

TUGAS PRAKTIKUM KP 9

LINKED LIST
(dan Stack)

Linked list adalah struktur data yang digunakan dalam pemrograman komputer untuk menyimpan dan mengorganisir data. Ini terdiri dari sejumlah simpul (node) yang terhubung satu sama lain. Setiap simpul memiliki dua komponen utama: data dan tautan ke simpul berikutnya dalam urutan.

Karakteristik Utama Linked List:

- Node: Setiap elemen dalam linked list disebut node. Node memiliki dua bagian, yaitu data (informasi yang ingin disimpan) dan tautan ke node berikutnya dalam linked list.
- Tautan (Pointer): Setiap node memiliki tautan yang menghubungkannya ke node berikutnya dalam urutan. Tautan terakhir diarahkan ke NULL, menunjukkan akhir dari linked list.

Gambaran Singkat Implementasi Linked List

Variabel yang Dibutuhkan:

- Head: Variabel yang menyimpan alamat node pertama dalam linked list.
- Node Structure: Struktur data yang mendefinisikan sebuah node, termasuk data dan tautan.

Contoh pengaplikasian (dalam bahasa C) :

```c
struct Node {
    int data;
    struct Node* next;
};

struct Node* head = NULL;  // Deklarasi head
```

Fungsi yang Digunakan:

- Insertion (Penyisipan): Menambahkan node baru ke linked list.
- Deletion (Penghapusan): Menghapus node tertentu dari linked list.
- Traversal (Penelusuran): Menampilkan atau memanipulasi setiap node dalam linked list.

Contoh pengaplikasian (dalam bahasa C) :

```c
void insertNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = head;
    head = newNode;
}

void deleteNode(int key) {
    struct Node *current = head, *prev = NULL;

    while (current != NULL && current->data != key) {
        prev = current;
        current = current->next;
    }

    if (current == NULL)
        return;

    if (prev != NULL)
        prev->next = current->next;
    else
        head = current->next;

    free(current);
}

void displayList() {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}
```

Selain itu, untuk mempermudah pemahaman, saya mempelajari materi Linked List melalui Youtube, khususnya melalui Youtube Channel Neso Academy dan juga melalui Website Log2Base2. Berikut ini beberapa potongan video yang dapat membantu pemahaman dengan gambar.

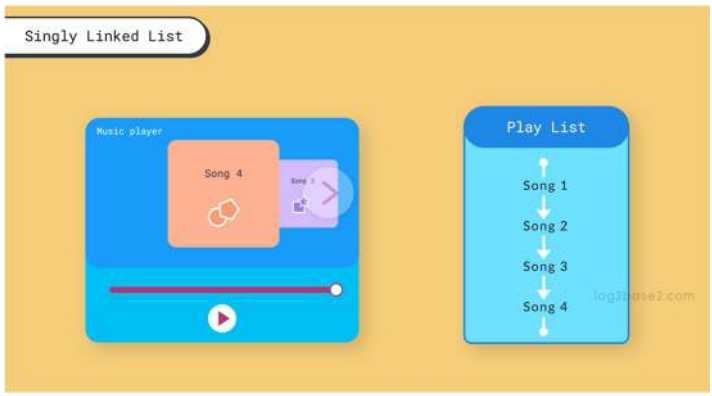
Sebelumnya, perlu diketahui bahwa pada materi Linked List, terdapat dua jenis Linked List, yaitu Singly Linked List dan Doubly Linked List. Berikut ini adalah penjelasan singkat untuk keduanya :

Singly Linked List

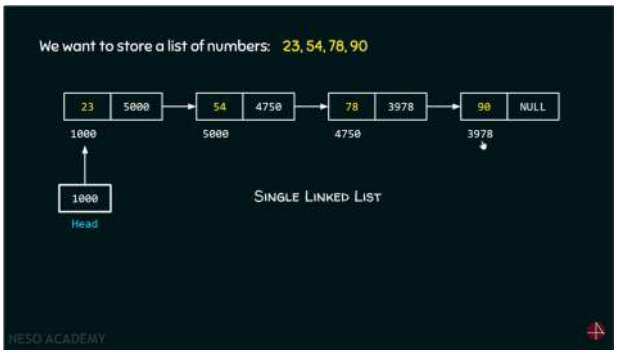
Definisi: Singly linked list adalah jenis linked list yang setiap node-nya hanya memiliki satu tautan yang mengarah ke node berikutnya dalam urutan.
Karakteristik Utama:

- Setiap node memiliki dua bagian: data dan tautan (next).
- Tautan pada node terakhir menunjuk ke NULL, menandakan akhir dari linked list.
- Struktur simpul (node) umumnya sederhana.

Contoh gambar untuk menjelaskan Singly Linked List :

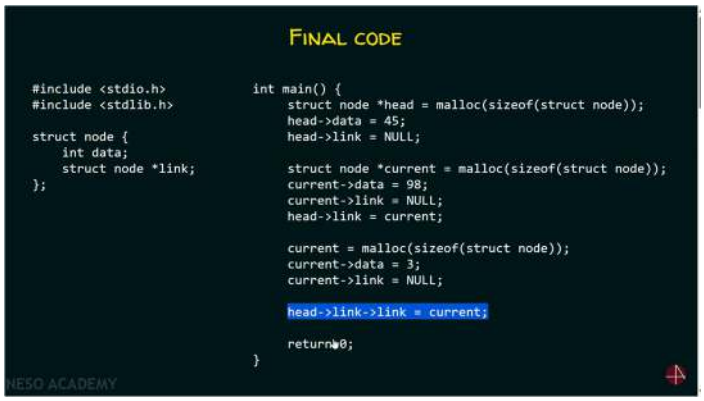


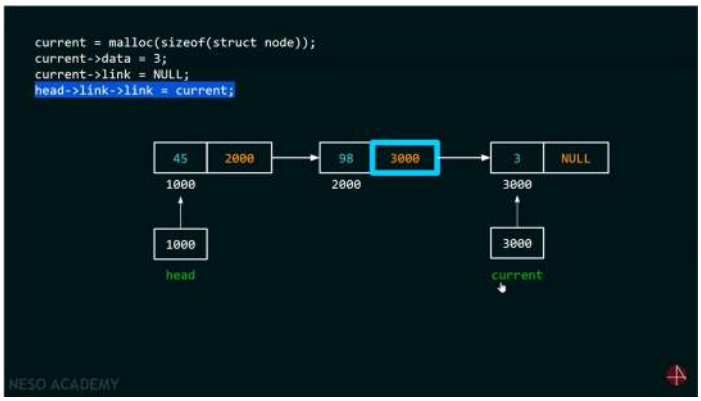
Pada gambar tersebut, kita dapat memahami pengaplikasian Singly Linked List dengan mudah, yaitu ketika kita sedang memutar musik, dimana kita mulai dengan lagu 1, dilanjutkan dengan lagu 2, dan seterusnya hingga akhir lagu, tapi kita tidak bisa kembali ke lagu sebelumnya, inilah yang membedakan Singly Linked List dengan Doubly Linked list yang akan dibahas setelah ini.



Pada gambar tersebut, dapat dilihat bahwa pada Singly Linked List, data yang tersimpan dimulai dari variabel Head, lalu terhubung (linked) dengan angka 23 dengan cara menyimpan alamat dari node yang memiliki angka 23 tersebut, yaitu 1000. kemudian pada angka 23 terdapat alamat 5000, yang merupakan alamat dari angka 54 agar dapat saling terhubung, struktur tersebut berulang hingga pada akhir bagian , yaitu angka 90, sudah tidak menyimpan alamat dari node selanjutnya yang berarti angka 90 adalah akhir dari list tersebut dan dapat dikatakan sebagai Tail dari list tersebut.

Berikut ini juga contoh code dan hasil yang didapatkan untuk menggunakan Singly Linked List :





Doubly Linked List

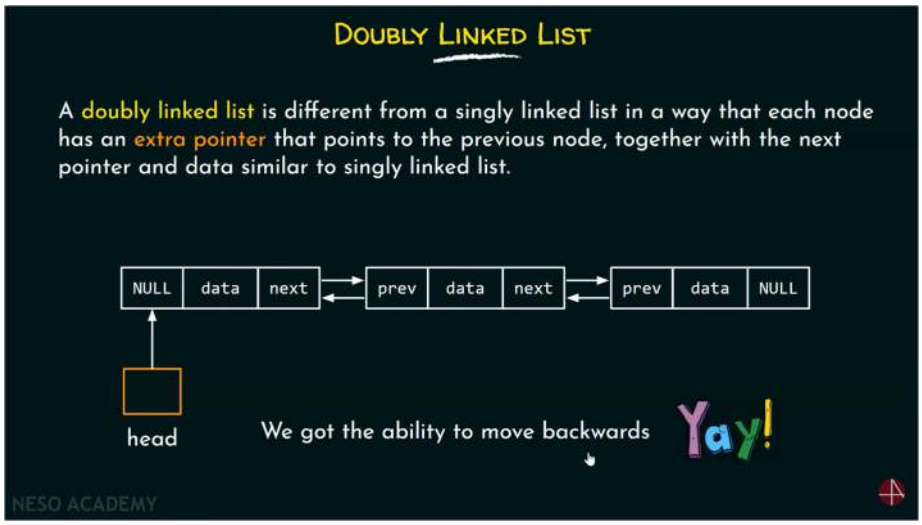
Definisi: Doubly linked list adalah jenis linked list yang setiap node-nya memiliki dua tautan, satu yang mengarah ke node sebelumnya dan satu yang mengarah ke node berikutnya dalam urutan.

Karakteristik Utama:

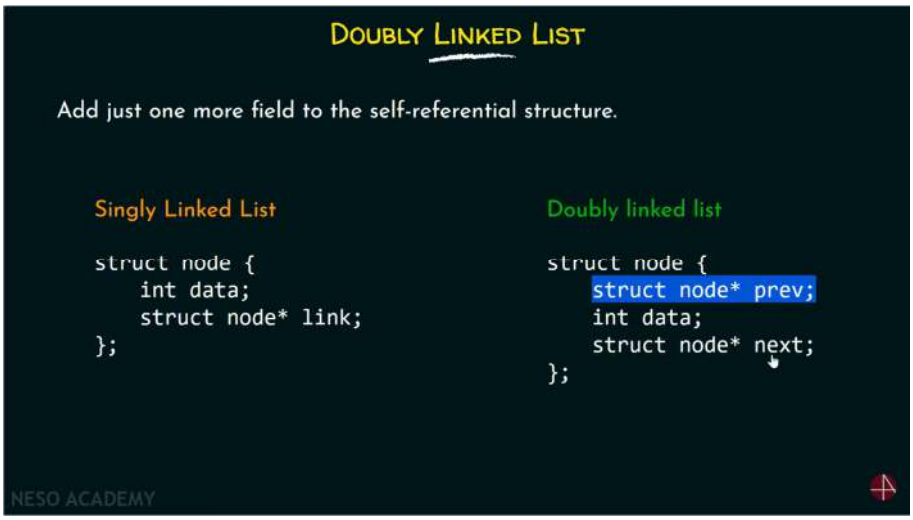
- Setiap node memiliki tiga bagian: data, tautan ke node berikutnya (next), dan tautan ke node sebelumnya (previous).
- Tautan pada node pertama dan terakhir dapat menunjuk ke NULL atau mengelilingi linked list.
- Lebih kompleks daripada singly linked list karena memiliki tautan ke node sebelumnya.

Contoh gambar untuk menjelaskan Doubly Linked List :

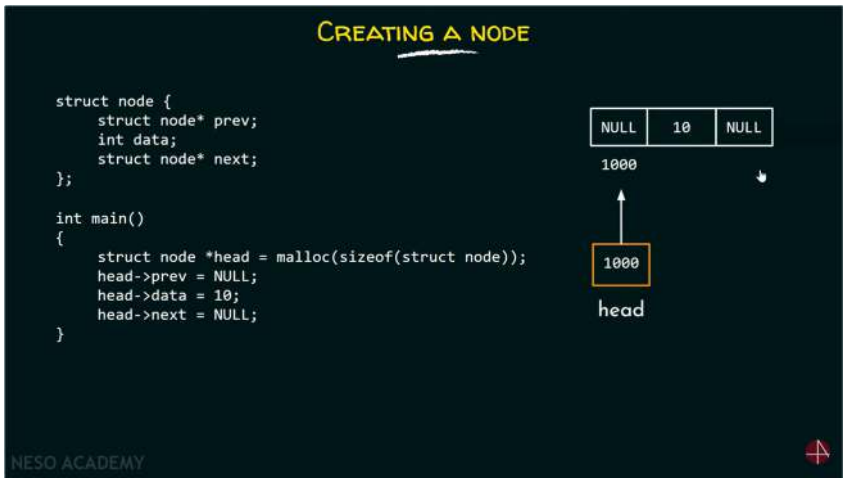
Pada gambar tersebut, kita dapat memahami pengaplikasian Singly Linked List dengan mudah, yaitu ketika kita sedang memutar musik, dimana kita mulai dengan lagu 1, dilanjutkan dengan lagu 2, dan seterusnya hingga akhir lagu, namun pada Double Linked List, kita bisa kembali dari lagu 2 ke lagu 1, tidak seperti Singly Link List yang hanya bisa maju tanpa kembali ke lagu sebelumnya.



Dapat dilihat pada gambar tersebut, dijelaskan bahwa yang paling membedakan antara Doubly Linked List dan Singly Linked List adalah pada Doubly Linked List, kita dapat maju ke data selanjutnya, dan juga mundur kembali ke data sebelumnya, berbeda dengan Singly Linked List yang hanya bisa maju ke data selanjutnya.



Berikut ini struktur tambahan pada node untuk Doubly Linked List



Dan berikut ini gambaran sederhana cara membuat node untuk Doubly Linked List

Kedua jenis linked list memiliki kelebihan dan kekurangan tergantung pada kebutuhan aplikasi. Singly linked list cenderung lebih sederhana dan memerlukan sedikit ruang memori, sementara doubly linked list memberikan keleluasaan tambahan dalam navigasi maju dan mundur, tetapi memerlukan lebih banyak ruang memori untuk menyimpan tautan tambahan. Pemilihan antara keduanya tergantung pada persyaratan dan kinerja aplikasi yang spesifik.

Stacks

Stack adalah struktur data yang mengikuti prinsip "Last In, First Out" (LIFO). Artinya, elemen terakhir yang dimasukkan adalah elemen pertama yang diambil. Seperti tumpukan buku yang diletakkan satu per satu, buku yang terakhir diletakkan di atas menjadi buku yang pertama diambil.

Operasi Dasar pada Stack:

- Push: Menambahkan elemen ke atas stack.
- Pop: Menghapus elemen dari atas stack.
- Peek atau Top: Mendapatkan nilai elemen teratas tanpa menghapusnya.
- isEmpty: Memeriksa apakah stack kosong.
- isFull: Memeriksa apakah stack penuh (untuk stack dengan ukuran terbatas).

Contoh Penggunaan Stack:

- Undo dalam Aplikasi: Stack dapat digunakan untuk menyimpan aksi pengguna sehingga dapat dilakukan operasi undo dengan mengambil elemen teratas dari stack.
- Evaluasi Ekspresi Postfix: Stack dapat digunakan untuk mengimplementasikan evaluasi ekspresi postfix.

Hubungan dengan Linked List:

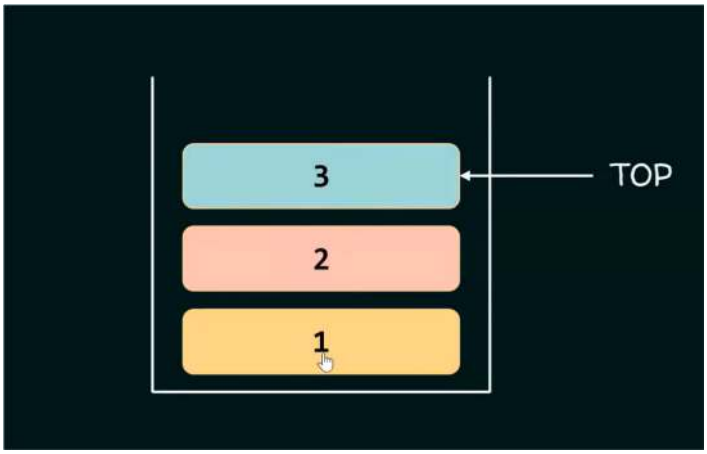
1. **Implementasi dengan Linked List:**

- Stack dapat diimplementasikan menggunakan linked list. Setiap elemen stack direpresentasikan sebagai node dalam linked list.
- Operasi push dan pop dilakukan dengan menambah atau menghapus elemen pada awal linked list (head).

2. **Node Struktur untuk Stack:**

```
struct Node {
    int data;
    struct Node* next;
};
```

3. **Contoh Visualisasi Linked List sebagai Stack:**



4. **Operasi Push (Menambah Elemen ke Stack):**

```
void push(struct Node** top, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = *top;
    *top = newNode;
}
```

5. **Operasi Pop (Mengambil Elemen dari Stack):**

```
int pop(struct Node** top) {
    if (*top == NULL) {
        printf("Stack underflow\n");
        return -1; // Nilai khusus untuk menunjukkan kesalahan
    }
    struct Node* temp = *top;
    int data = temp->data;
    *top = temp->next;
    free(temp);
    return data;
}
```

Stack dan linked list memiliki keterkaitan karena implementasi stack yang umum seringkali melibatkan struktur linked list. Oleh karena itu, pemahaman tentang linked list dapat mempermudah pemahaman tentang struktur stack dan sebaliknya.