

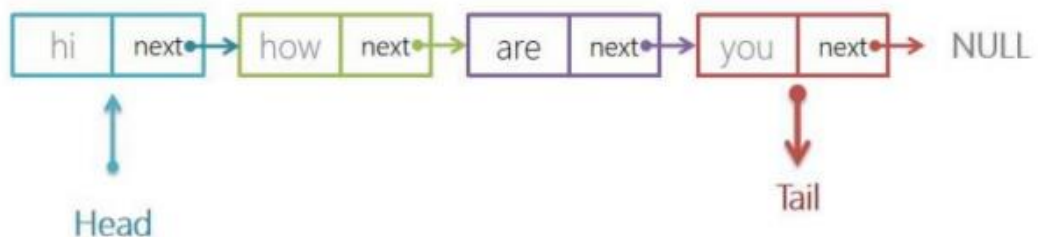
MODUL 4

LINKED LIST CIRCULAR DAN NON CIRCULAR

A. DASAR TEORI

1. Linked List Non Circular

Linked list non circular merupakan linked list dimana antara kepala (head) dan node terakhir (tail) tidak memiliki hubungan. Pada Linked List ini maka pointer terakhir selalu menunjuk 'NULL' sebagai pertanda data terakhir dalam list-nya. Linked list non circular dapat digambarkan sebagai berikut.



Gambar 1 Single Linked List Non Circular

OPERASI LINKED LIST NON CIRCULAR

1. Deklarasi Node/Simpul

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

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
node *head, *tail;
void init()
{
    head = NULL;
    tail = NULL;
};
```

3. Pengecekan Kondisi Linked List

```
bool isEmpty()
{
    if (head == NULL && tail == NULL)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

4. Operasi Penambahan Simpul

```
void insertBelakang(string dataUser)
{
    if (isEmpty() == true)
    {
        node *baru = new node;
        baru->data = dataUser;
        head = baru;
        tail = baru;
        baru->next = NULL;
    }

    else
    {
        node *baru = new node;
        baru->data = dataUser;
        baru->next = NULL;
        tail->next = baru;
        tail = baru;
    }
}
```

```
};
```

5. Operasi Penghapusan Simpul

```
void hapusDepan()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        node *helper;
        helper = head;
        if (head == tail)
        {
            head = NULL;
            tail = NULL;
            delete helper;
        }
        else
            head = head->next;
        helper->next = NULL;
        delete helper;
    }
}
```

6. Tampil Data Linked List

```
void tampil()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
}
```

```

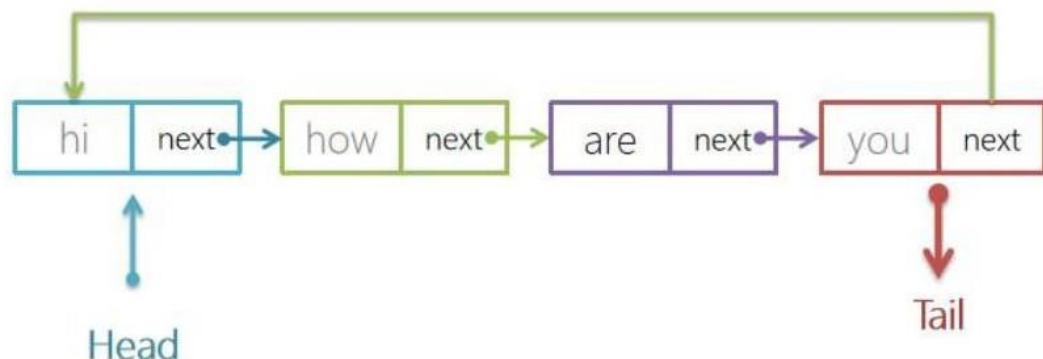
    }
    else
    {
        node *helper;
        helper = head;
        while (helper != NULL)
        {
            cout << helper->data << ends;
            helper = helper->next;
        }
    }
}

```

2. Linked List Circular

Linked list circular adalah struktur data linked list yang memiliki sifat berputar atau mengulang dari node head ke node tail. Dalam linked list circular, node tail tidak menunjuk ke '**NULL**', melainkan mengarah kembali ke node head, sehingga membentuk lingkaran. Pada node tail dalam linked list circular, pointer menunjuk kembali ke node head.

Linked list circular dapat digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu dalam pemutar musik, daftar pesan dalam antrian, atau penggunaan memori yang berulang dalam suatu aplikasi.



Gambar 2 Single Linked List Circular

OPERASI LINKED LIST CIRCULAR

1. Deklarasi Node/Simpul

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

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
Node *head, *tail, *baru, *bantu, *hapus;

void init()
{
    head = NULL;
    tail = head;
}
```

3. Pengecekan Kondisi Linked List

```
int isEmpty()
{
    if (head == NULL)
        return 1; // true
    else
        return 0; // false
}
```

4. Pembuatan Node

```
void buatNode(string data)
{
    baru = new Node;
    baru->data = data;
    baru->next = NULL;
}
```

5. Penambahan Node

```
// Tambah Depan
void insertDepan(string data)
{
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        while (tail->next != head)
        {
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}
```

6. Penghapusan Node

```
void hapusBelakang()
{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
```

```

        head = NULL;
        tail = NULL;

        delete hapus;
    }
    else
    {
        while (hapus->next != head)
        {
            hapus = hapus->next;
        }
        while (tail->next != hapus)
        {
            tail = tail->next;
        }
        tail->next = head;
        hapus->next = NULL;

        delete hapus;
    }
}

```

7. Penampilan Data Linked List Circular

```

void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
    }
}

```

```
        cout << endl;
    }
}
```

B. GUIDED

1. Linked List Non Circular

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    int data;
    Node *next;
};
Node *head;
Node *tail;

// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}

// Pengecekan
bool isEmpty()
{
    if (head == NULL)
        return true;
    else
        return false;
}
```



```
}

// Tambah Depan
void insertDepan(int nilai)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

// Tambah Belakang
void insertBelakang(int nilai)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
}
```

```

else
{
    tail->next = baru;
    tail = baru;
}
}

// Hitung Jumlah List
int hitungList()
{
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

// Tambah Tengah
void insertTengah(int data, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else

```

```

{
    Node *baru, *bantu;
    baru = new Node();
    baru
        ->data = data;
    // tranversing
    bantu = head;
    int nomor = 1;
    while (nomor < posisi - 1)
    {
        bantu = bantu
            ->next;

        nomor++;
    }
    baru
        ->next = bantu
            ->next;

    bantu
        ->next = baru;
    }
}

```

// Hapus Depan

```

void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head
            ->next != NULL)
        {
            hapus = head;
            head = head

```

```

        ->next;
        delete hapus;
    }
    else
    {
        head = tail = NULL;
    }
}
else
{
    cout << "List kosong!" << endl;
}
}

```

// Hapus Belakang

```

void hapusBelakang()
{
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false)
    {
        if (head != tail)
        {
            hapus = tail;
            bantu = head;
            while (bantu
                ->next != tail)
            {
                bantu = bantu
                    ->next;
            }
            tail = bantu;
            tail->next = NULL;
        }
    }
}

```

```

        delete hapus;
    }
    else
    {
        head = tail = NULL;
    }
}
else
{
    cout << "List kosong!" << endl;
}
}

// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *bantu, *hapus, *sebelum;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        int nomor = 1;
        bantu = head;
        while (nomor <= posisi)
        {
            if (nomor == posisi - 1)
            {

```

```

        sebelum = bantu;
    }
    if (nomor == posisi)
    {
        hapus = bantu;
    }
    bantu = bantu->next;
    nomor++;
}
sebelum->next = bantu;
delete hapus;
}
}

// Ubah Depan
void ubahDepan(int data)
{
    if (isEmpty() == 0)
    {
        head->data = data;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah Tengah
void ubahTengah(int data, int posisi)
{
    Node *bantu;
    if (isEmpty() == 0)
    {

```

```

        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {
            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

```

// Ubah Belakang

```

void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }
}

```

```

else
{
    cout << "List masih kosong!" << endl;
}
}

// Hapus List
void clearList()
{
    Node *bantu, *hapus;
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

// Tampilkan List
void tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << ends;
            bantu = bantu->next;
        }
    }
}

```



```
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();
}
```

```
    return 0;

}
```

2. Linked List Circular

```
#include <iostream>

using namespace std;

/// PROGRAM SINGLE LINKED LIST CIRCULAR

// Deklarasi Struct Node
struct Node
{
    string data;
    Node *next;
};

Node *head, *tail, *baru, *bantu, *hapus;

void init()
{
    head = NULL;
    tail = head;
}

// Pengecekan
int isEmpty()
{
    if (head == NULL)
        return 1; // true
    else
        return 0; // false
}
```

```
}

// Buat Node Baru
void buatNode(string data)
{
    baru = new Node;
    baru->data = data;
    baru->next = NULL;
}

// Hitung List
int hitungList()
{
    bantu = head;
    int jumlah = 0;

    while (bantu != NULL)
    {
        jumlah++;
        bantu = bantu->next;
    }

    return jumlah;
}

// Tambah Depan
void insertDepan(string data)
{
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1)
    {
```

```

        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        while (tail->next != head)
        {
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}

```

// Tambah Belakang

```

void insertBelakang(string data)
{
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        while (tail->next != head)
        {
            tail = tail->next;

```

```

    }

    tail->next = baru;
    baru->next = head;
}
}

// Tambah Tengah
void insertTengah(string data, int posisi)
{
    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        baru->data = data;
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
}

```

```

// Hapus Depan

```

```
void hapusDepan()
{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;

        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;

            delete hapus;
        }
        else
        {
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            head = head->next;
            tail->next = head;
            hapus->next = NULL;

            delete hapus;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
```

```
// Hapus Belakang
void hapusBelakang()
{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;

            delete hapus;
        }
        else
        {
            while (hapus->next != head)
            {
                hapus = hapus->next;
            }
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;

            delete hapus;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
```

```

    }
}

// Hapus Tengah
void hapusTengah(int posisi)
{
    if (isEmpty() == 0)
    {
        // transversing
        int nomor = 1;
        bantu = head;

        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;

        delete hapus;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus List
void clearList()
{
    if (head != NULL)
    {

```



```

        hapus = head->next;

        while (hapus != head)
        {
            bantu = hapus->next;
            delete hapus;
            hapus = bantu;
        }
        delete head;
        head = NULL;
    }
    cout << "List berhasil terhapus!" << endl;
}

// Tampilkan List
void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

```

```

int main()
{
    init();
    insertDepan("Ayam");
    tampil();
    insertDepan("Bebek");
    tampil();
    insertBelakang("Cicak");
    tampil();
    insertBelakang("Domba");
    tampil();
    hapusBelakang();
    tampil();
    hapusDepan();
    tampil();
    insertTengah("Sapi", 2);
    tampil();
    hapusTengah(2);
    tampil();

    return 0;
}

```

C. UNGUIDED

Buatlah program menu Linked List Non Circular untuk menyimpan **Nama** dan **NIM mahasiswa**, dengan menggunakan *inputan* dari *user*.

Lakukan operasi berikut:

1. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah)

[Nama_Anda] [NIM_Anda] → data pertama yang diinput adalah Nama & NIM Anda

Nama	NIM
------	-----

Alvin	22200001
Candra	22200002
Niken	22200005
Joko	22200008
Friska	22200015
Gabriel	22200040
Karin	22200020

2. Hapus data Karin

3. Tambahkan data berikut diantara data Joko dan Friska:

Cika 22200003

4. Hapus data Joko

5. Tambahkan data berikut di awal:

Dimas 22200010

6. Tambahkan data berikut diantara data Dimas dan Anda:

Vina 22200022

7. Ubah data Gabriel menjadi data berikut:

Jamal 22200033

Ubah data Niken menjadi data berikut:

April 22200017

Tambahkan data berikut di akhir:

Budi 22200000

Ubah data NIM Candra menjadi:

21200055

Tampilkan seluruh data

Tampilan *output* program sebagai berikut:

- **Tampilan menu:**

PROGRAM SINGLE LINKED LIST NON-CIRCULAR

1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan

```
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. TAMPILKAN
0. KELUAR
```

Pilih Operasi :

- **Tampilan operasi tambah:**

-Tambah Depan-

Masukkan Nama : Aleksander

Masukkan NIM : 22212121

Data Aleksander berhasil *diinput*!

-Tambah Tengah-

Masukkan Nama : Cika

Masukkan NIM : 22200003

Masukkan posisi : 6

Data Cika berhasil *diinput*!

- **Tampilan operasi hapus:**

-Hapus belakang-

Data Karin berhasil dihapus

-Hapus Tengah-

Masukkan posisi: 5

Data Joko berhasil dihapus!

- **Tampilan operasi ubah:**

-Ubah Belakang-

Masukkan Nama : Jamal Masukkan

NIM : 22200033

Data Gabriel telah diganti dengan data Jamal !

-Ubah Tengah-

Masukkan Nama : April

Masukkan NIM : 22200017

Masukkan Posisi : 6

Data Niken telah diganti dengan data April !

- **Tampilan operasi Tampilkan list**

DATA MAHASISWA

NAMA	NIM
Dimas	22200010
Vina	22200022
Aleksander	22212121
Alvin	22200001
Candra	22200055
April	22200017
Cika	22200003
Friska	22200015

Jamal	22200033
Budi	22200000

*Catatan: Desain tampilan program dapat disesuaikan dengan kreativitas