

**LAPORAN PRAKTIKUM**  
**MODUL VI**  
**LINKED LIST CIRCULAR DAN NON CIRCULAR**



**Disusun Oleh:**

Muhammad Raafi Al Hafiidh

2311102070

**Dosen:**

Wahyu Andi Saputra, S.Pd., M.Eng.

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**  
**FAKULTAS INFORMATIKA**  
**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**  
**2024**

## **BAB I**

### **TUJUAN PRAKTIKUM**

- Mahasiswa memahami perbedaan konsep Linked list secara Sircular dan Non Sircular.
- Mahasiswa mampu menerapkan Linked list secara Sircular dan Non Sircular ke dalam pemrogramman.

## **BAB II**

### **DASAR TEORI**

**Linked List** adalah struktur data yang terdiri dari urutan elemen data yang disebut node. Setiap node terhubung satu sama lain dengan pointer yang menunjukkan alamat node selanjutnya.

#### **1. Linked List Non Circular**

- **Struktur:**
  - Setiap node memiliki dua field: data dan pointer next.
  - Pointer next pada node terakhir menunjuk ke NULL.
  - Akses data hanya bisa dilakukan dari node pertama (head).
- **Operasi:**
  - Insertion:
    - Di awal (head)
    - Di tengah
    - Di akhir
  - Deletion:
    - Di awal (head)
    - Di tengah
    - Di akhir
  - Traversal:
    - Dari awal (head) sampai akhir
- **Kelebihan:**
  - Implementasi mudah.
  - Memori yang dialokasikan dinamis.
  - Efisien untuk penyisipan dan penghapusan di tengah.
- **Kekurangan:**
  - Akses data hanya bisa dilakukan dari awal (head).
  - Penghapusan node di tengah membutuhkan traversal.

#### **2. Linked List Circular**

- **Struktur:**

- Setiap node memiliki dua field: data dan pointer next.
- Pointer next pada node terakhir menunjuk ke node pertama (head).
- Akses data bisa dilakukan dari node manapun.
- **Operasi:**
  - Insertion:
    - Di awal (head)
    - Di tengah
    - Di akhir
  - Deletion:
    - Di awal (head)
    - Di tengah
    - Di akhir
  - Traversal:
    - Bisa dimulai dari node manapun
    - Berhenti ketika kembali ke node awal
- **Kelebihan:**
  - Akses data bisa dilakukan dari node manapun.
  - Efisien untuk operasi insert dan delete di tengah.
- **Kekurangan:**
  - Implementasi sedikit lebih rumit.
  - Penghapusan node di tengah membutuhkan traversal.

## BAB III

### LATIHAN DAN TUGAS

#### A. GUIDED

##### 1. Guided 1

###### *Source Code*

```
#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 *hapus, *bantu, *bantu2;
    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)
            {
                bantu2 = bantu;
            }
            if (nomor == posisi)
            {
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
        }
        bantu2->next = bantu;
        delete hapus;
    }
}

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

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

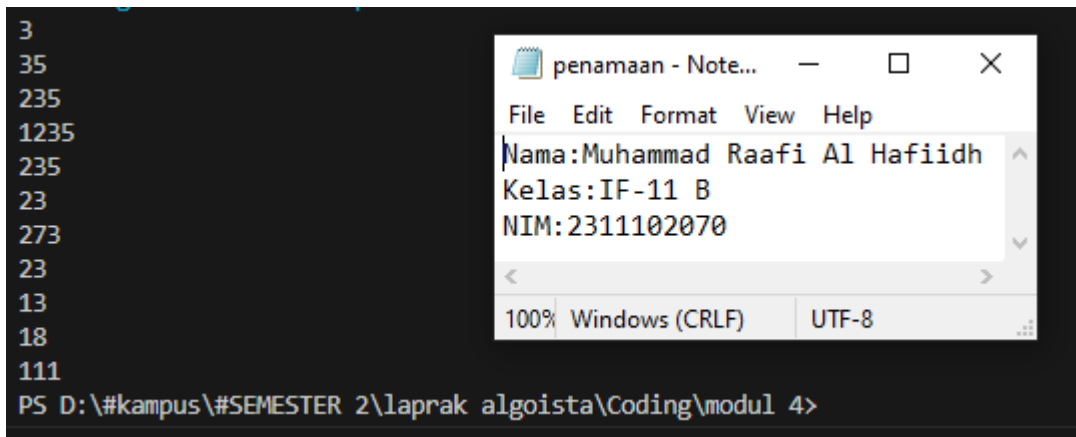
```

```

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;
}

```

### *Screenshoot Program*



### Deskripsi Program

Program di atas merupakan sebuah program yang menggunakan fungsi Single Linked list. Program tersebut menjalankan beberapa fungsi diantaranya insertDepan(3) yang artinya program diperintahkan untuk melakukan fungsi insertDepan() dengan memasukkan angka 3 sebagai inputan di depan single linked list, lalu ada insertBelakang(5) yang artinya program diperintahkan untuk melakukan fungsi insertBelakang() dengan memasukkan angka 5 sebagai inputan di belakang single linked list. Lalu ada insertTengah(7, 2) artinya program akan diperintahkan memasukkan angka 7 sebagai inputan di posisi ke 2 didalam single linked list. Selanjutnya ada hapusDepan() dan hapusBelakang() artinya program akan diperintahkan untuk menghapus angka yang ada di depan dan di belakang single linked list. Selanjutnya ada fungsi ubahDepan(1) dimana program akan diperintah untuk mengubah data single linked list paling depan dengan angka 1, berikutnya ubahBelakang(8) yang sama seperti fungsi ubahDepan() tetapi perbedaannya fungsi tersebut akan merubah data paling belakang atau akhir dari single linked list. ubahTengah(11, 2) berarti program akan merubah data pada single linked list di posisi ke 2 menjadi angka 11. Lalu yang terakhir setiap fungsi dijalankan maka hasil dari fungsi tersebut akan ditampilkan menggunakan fungsi tampil().

## 2. Guided 2

### Source Code

```
#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;
    }
}

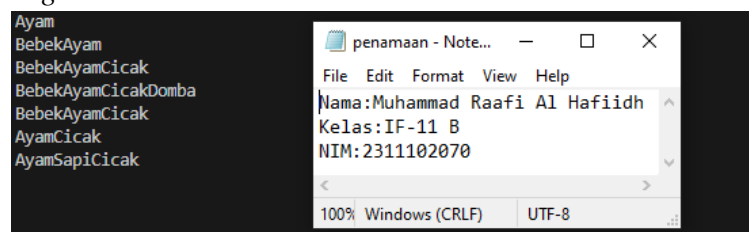
```

```

    }
}
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;
}

```

### *Screenshoot Program*



### **Deskripsi Program**

Program tersebut merupakan program yang menggunakan double linked list dalam proses nya. Pertama program akan menampilkan 6 menu dalam mengelola double linked list nantinya. Ada Add data, Delete data, Update data, Clear data, Display data, dan Exit. Untuk menambahkan data user harus memilih 1 untuk masuk ke menu Add data lalu memasukkan angka berapa yang ingin dimasukkan, setelah itu user akan kembali lagi ke menu awal. Apabila user ingin menghapus data maka user harus memilih 2 untuk ke menu Delete data dan memilih angka mana yang ingin dihapus, apabila user ingin mengubah data yang sebelumnya sudah dimasukkan maka user bisa memilih Update data dengan memasukkan 3 sebagai pilihan, nantinya user akan diminta memasukkan angka sebelumnya yang ingin diubah dan memasukkan angka baru. Lalu apabila user ingin menghapus semua data yang ada, user bisa memasukkan angka 4 untuk menghapus semua data yang ada sekaligus. Apabila user ingin melihat data yang sudah diinputkan user bisa memasukkan angka 5 untuk ke menu Display data, nantinya semua data yang ada akan ditampilkan secara berderet. Terakhir apabila user ingin keluar atau menyelesaikan program user harus memasukkan angka 6 dan menuju ke menu Exit.



## B. 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 di-*input* 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 diawal :  
**Dimas       22200010**
6. Tambahkan data berikut diantara data Dimas dan Anda :  
**Vina         22200022**
7. Ubah data Gabriel menjadi data berikut :  
**Jamal        22200001**

Ubah data Niken menjadi data berikut :

**April        22200017**

Tambahkan data berikut diakhir :

**Budi         22200000**

Ubah data NIM Candra menjadi :

**22200055**

Tampilkan seluruh data

Tampilkan *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!

-Tambah Depan-

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.

*Source Code*

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    string nama;
    int nim;
    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(string nama, int nim)
    {
        // Buat Node baru
        Node *baru = new Node;
        baru->nama = nama;
        baru->nim = nim;
        baru->next = NULL;
        if (isEmpty() == true)
        {
            head = tail = baru;
            tail->next = NULL;
        }
        else
        {
            baru->next = head;
            head = baru;
        }
    }
    // Tambah Belakang
    void insertBelakang(string nama, int nim)
    {
        // Buat Node baru
        Node *baru = new Node;
        baru->nama = nama;
        baru->nim = nim;
        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(string nama, int nim, 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->nama = nama;
        baru->nim = nim;
        // 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(string nama, int nim)
{
    if (isEmpty() == 0)
    {
        head->nama = nama;
        head->nim = nim;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Tengah
void ubahTengah(string nama, int nim, 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->nama = nama;
            bantu->nim = nim;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

```

```

}
// Ubah Belakang
void ubahBelakang(string nama, int nim)
{
    if (isEmpty() == 0)
    {
        tail->nama = nama;
        tail->nim = nim;
    }
    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 << "Nama: " << bantu->nama << " Nim: " <<
bantu->nim << endl;
            bantu = bantu->next;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
        return;
    }
}
int main()
{
    string nama;
    int nim;

```



```

int pilihan;
int posisi;

do
{

    cout << "\n===Data Mahasiswa===" << endl;
    cout << "1. Tambah Depan" << endl;
    cout << "2. Tambah Belakang" << endl;
    cout << "3. Tambah Tengah" << endl;
    cout << "4. Ubah Depan" << endl;
    cout << "5. Ubah Belakang" << endl;
    cout << "6. Ubah Tengah" << endl;
    cout << "7. Hapus Depan" << endl;
    cout << "8. Hapus Belakang" << endl;
    cout << "9. Hapus Tengah" << endl;
    cout << "10. Hapus List" << endl;
    cout << "11. Tampilkan" << endl;
    cout << "12. Exit" << endl;
    cout << "Pilihan :" << endl;
    cin >> pilihan;
    switch (pilihan)
    {
    case 1:
    {
        cout << "--Tambah Depan--" << endl;
        cout << "Masukkan Nama :";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin.ignore();
        cin >> nim;
        insertDepan(nama, nim);
        cout << "Data " << nama << " berhasil diinput!";

        break;
    }
    case 2:
    {
        cout << "--Tambah Belakang--" << endl;
        int jumlah;
        cout << "Masukan jumlah data yang diinputkan: ";
        cin >> jumlah;
        for (int q = 1; q <= jumlah; q++)
        {
            cout << "Masukkan Nama :";
            cin.ignore();
            getline(cin, nama);
            cout << "Masukkan Nim: ";
            cin >> nim;
            insertBelakang(nama, nim);
        }
        break;
    }
    }
}

```

```

    }
    case 3:
    {
        cout << "--Tambah Tengah--" << endl;
        cout << "Masukkan Nama :";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin >> nim;
        cout << "Masukan Posisi: ";
        cin >> posisi;
        insertTengah(nama, nim, posisi);
        cout << "Data " << nama << " Berhasil diinput!" <<
endl;
        break;
    }
    case 4:
    {
        cout << "--Ubah Depan--" << endl;
        cout << "Masukkan Nama :";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin >> nim;
        ubahDepan(nama, nim);
        break;
    }
    case 5:
    {
        cout << "--Ubah Belakang--" << endl;
        cout << "Masukkan Nama :";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin >> nim;
        ubahBelakang(nama, nim);
        break;
    }
    case 6:
    {
        cout << "--Ubah Tengah--" << endl;
        cout << "Masukkan Nama :";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan Nim: ";
        cin >> nim;
        cout << "Masukkan Posisi: ";
        cin >> posisi;
        ubahTengah(nama, nim, posisi);
        break;
    }
}

```

```

case 7:
{
    cout << "--Hapus Depan--" << endl;
    hapusDepan();
    cout << "Data Depan berhasil terhapus!";
    break;
}
case 8:
{
    cout << "--Hapus Belakang--" << endl;
    hapusBelakang();
    cout << "Data Belakang berhasil terhapus!";
    break;
}
case 9:
{
    cout << "--Hapus Tengah--" << endl;
    cout << "Masukkan Posisi: ";
    cin >> posisi;
    hapusTengah(posisi);
    cout << "Data Tengah berhasil terhapus!";
    break;
}
case 10:
{
    clearList();
    break;
}

case 11:
{
    tampil();
    break;
}
case 12:
{
    cout << "Terima Kasih!" << endl;
}

default:
{
    cout << "Pilihan tidak Valid!" << endl;
    break;
}
}
} while (pilihan != 12);

return 0;
}

```

*Screenshoot Program*

```
--Tambah Depan--  
Masukkan Nama :Muhammad Raafi Al Hafiidh  
Masukkan Nim: 2311102070  
Data Muhammad Raafi Al Hafiidh berhasil diinput!
```

```
--Tambah Belakang--  
Masukan jumlah data yang diinputkan: 7  
Masukkan Nama :Alvin  
Masukkan Nim: 22200001  
Masukkan Nama :Candra  
Masukkan Nim: 22200002  
Masukkan Nama :Niken  
Masukkan Nim: 22200005  
Masukkan Nama :Joko  
Masukkan Nim: 22200008  
Masukkan Nama :Friska  
Masukkan Nim: 22200015  
Masukkan Nama :Gabriel  
Masukkan Nim: 22200040  
Masukkan Nama :Karin  
Masukkan Nim: 22200020
```

```
--Hapus Belakang--  
Data Belakang berhasil terhapus!
```

```
--Tambah Tengah--  
Masukkan Nama :Cika  
Masukkan Nim: 22200003  
Masukan Posisi: 6  
Data Cika Berhasil diinput!
```

```
--Hapus Tengah--  
Masukkan Posisi: 5  
Data Tengah berhasil terhapus!
```

```
--Tambah Depan--  
Masukkan Nama :Dimas  
Masukkan Nim: 22200010  
Data Dimas berhasil diinput!
```

```
--Tambah Tengah--  
Masukkan Nama :Vina  
Masukkan Nim: 22200022  
Masukan Posisi: 2  
Data Vina Berhasil diinput!
```

```
--Ubah Tengah--  
Masukkan Nama :April  
Masukkan Nim: 22200033  
Masukkan Posisi: 6
```

```
--Ubah Tengah--  
Masukkan Nama :Jamal  
Masukkan Nim: 22200033  
Masukkan Posisi: 8
```

```
--Tambah Belakang--  
Masukan jumlah data yang diinputkan: 1  
Masukkan Nama :Budi  
Masukkan Nim: 22200000
```

```
--Ubah Tengah--  
Masukkan Nama :Candra  
Masukkan Nim: 21200055  
Masukkan Posisi: 5
```

```
Nama: Dimas Nim: 2200010  
Nama: Vina Nim: 22200022  
Nama: Muhammad Raafi Al Hafiidh Nim: 311102070  
Nama: Alvin Nim: 22200001  
Nama: Candra Nim: 21200055  
Nama: April Nim: 22200017  
Nama: Cika Nim: 22200003  
Nama: Friska Nim: 22200015  
Nama: Jamal Nim: 22200033  
Nama: Budi Nim: 22200000
```

### Deskripsi Program

Program diatas merupakan program implementasi konsep Single Linked List Cicular pada data mahasiswa, user diminta menginputkan Nama dan NIM dan memiliki tampilan menu seperti :

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
12. KELUAR

Pilih Operasi :

Program diatas ini hampir sama dengan Single Linked List Circular yang membedakan adalah non circular dia Kembali ke NULL sedangkan circular dia ke head jadi datanya akan terus berputar.

## **BAB IV**

### **KESIMPULAN**

#### **Linked List Non Circular:**

- **Struktur:**
  - Setiap node memiliki dua field: data dan pointer next.
  - Pointer next pada node terakhir menunjuk ke NULL.
  - Akses data hanya bisa dilakukan dari node pertama (head).
- **Kelebihan:**
  - Implementasi mudah.
  - Memori yang dialokasikan dinamis.
  - Efisien untuk penyisipan dan penghapusan di tengah.
- **Kekurangan:**
  - Akses data hanya bisa dilakukan dari awal (head).
  - Penghapusan node di tengah membutuhkan traversal.

#### **Linked List Circular:**

- **Struktur:**
  - Setiap node memiliki dua field: data dan pointer next.
  - Pointer next pada node terakhir menunjuk ke node pertama (head).
  - Akses data bisa dilakukan dari node manapun.
- **Kelebihan:**
  - Akses data bisa dilakukan dari node manapun.
  - Efisien untuk operasi insert dan delete di tengah.
- **Kekurangan:**
  - Implementasi sedikit lebih rumit.
  - Penghapusan node di tengah membutuhkan traversal.

#### **Pilihan Jenis Linked List:**

Pilihan jenis linked list yang tepat tergantung pada kebutuhan aplikasi.

- **Linked list non circular:** Cocok untuk aplikasi yang membutuhkan akses data yang cepat dari awal dan operasi penyisipan dan penghapusan di tengah yang sering dilakukan.

- **Linked list circular:** Cocok untuk aplikasi yang membutuhkan akses data yang cepat dari node manapun dan operasi penyisipan dan penghapusan di tengah yang jarang dilakukan.

## **REFERENSI**

Asisten Praktikum. "Modul 1 Tipe Data". Learning Management System. 2024.

Introduction to Algorithms, 4th Edition (Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, 2022)