

LAPORAN PRAKTIKUM
MODUL 4
LINKED LIST CIRCULAR DAN NON CIRCULAR



Disusun oleh :
Diva Octaviani
NIM : 2311102006

Dosen Pengampu:
Wahyu Andi Saputra, S. Pd., M. Eng.

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

BAB I

TUJUAN PRAKTIKUM

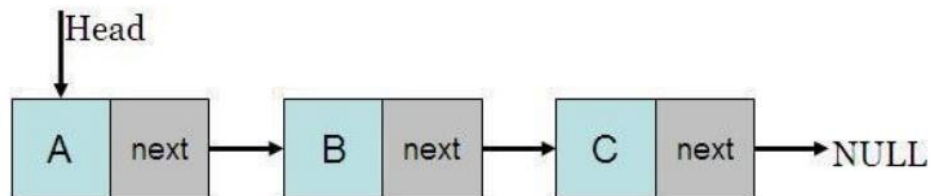
1. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
2. Praktikan dapat membuat linked list circular dan non circular.
3. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

BAB II

DASAR TEORI

1. Linked List Non Circular

Linked list non circular merupakan linked list dengan node pertama (head) dan node terakhir (tail) yang tidak saling terhubung. Pointer terakhir (tail) pada Linked List ini selalu bernilai 'NULL' sebagai pertanda data terakhir dalam list-nya. Linked list non circular dapat digambarkan sebagai berikut.



Asumsikan kita memiliki sejumlah node yang selalu menoleh ke sebelah dalam arah yang sama. Atau, sebagai alat bantu, kita bisa mengasumsikan beberapa orang yang bermain kereta api. Yang belakang akan memegang yang depan, dan semuanya menghadap arah yang sama. Setiap pemain adalah node. Dan tangan pemain yang digunakan untuk memegang bahu pemain depan adalah variabel next. Sampai di sini, kita baru saja mendeklarasikan tipe data dasar sebuah node. Selanjutnya, kita akan mendeklarasikan beberapa variabel pointer bertipe struct tnode. Beberapa variabel tersebut akan kita gunakan sebagai awal dari linked list, node aktif dalam linked list, dan node sementara yang akan digunakan dalam pembuatan node di linked list. Berikan nilai awal NULL kepada mereka. Deklarasi node untuk beberapa keperluan, seperti berikut ini:

```
struct tnode *head=NULL, *curr=NULL, *node=NULL;
```

Dengan demikian, sampai saat ini, telah dimiliki tiga node. Satu sebagai kepala (head), satu sebagai node aktif dalam linked list (curr) dan satu lagi node sementara (node). Untuk mempermudah pengingatan, ingatlah gambar anak panah yang mengarah ke kanan. Head akan berada pada pangkal anak panah, dan node-node berikutnya akan berbaris ke arah bagian anak panah yang tajam.

Operasi pada Linked List Non Circular

1) Deklarasi simpul (node)

```
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) Penambahan simpul (node)

```
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) Penghapusan simpul (node)

```
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

```

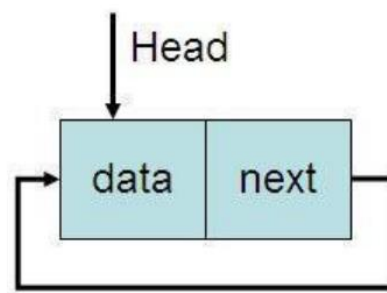
void tampil()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        node *helper;
        helper = head;
        while (helper != NULL)
        {
            cout << helper->data << ends;
            helper = helper->next;
        }
    }
}

```

2. Linked List Circular

Linked list circular merupakan linked list yang tidak memiliki akhir karena node terakhir (tail) tidak bernilai 'NULL', tetapi terhubung dengan node pertama (head). Saat menggunakan linked list circular kita membutuhkan dummy node atau node pengecoh yang biasanya dinamakan dengan node current supaya program dapat berhenti menghitung data ketika node current mencapai node pertama (head).

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



Operasi pada Linked List Circular

1) Deklarasi Simpul (Node)

```
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 Simpul (Node)

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

5) Penambahan Simpul (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 simpul (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) Menampilkan data linked list

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

BAB III

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; // Deklarasi head
Node *tail; // Deklarasi tail

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

// Pengecekan apakah linked list kosong
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;
        head->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;
        head->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 di luar 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 << "Linked list masih 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 << "Linked list masih kosong" << endl;
    }
}

// Hapus tengah
void hapusTengah(int posisi)
{
    Node *hapus, *bantu, *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 << "Linked 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
        {
            int nomor = 1;
            bantu = head;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }
    else

```



```

    {
        cout << "Linked list masih kosong" << endl;
    }
}

// ubah belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }
    else
    {
        cout << "Linked 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 tampilList()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {

```

```

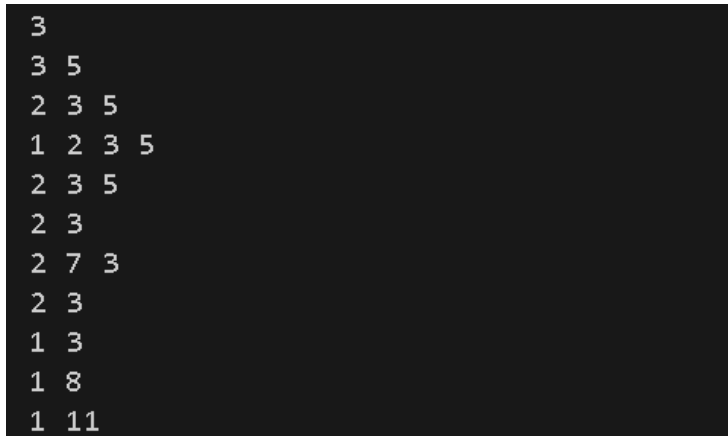
        cout << bantu->data << " ";
        bantu = bantu->next;
    }
    cout << endl;
}
else
{
    cout << "Linked list masih kosong" << endl;
}
}

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

    return 0;
}

```

Screenshoot program



```
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
```

Deskripsi program

Program ini merupakan implementasi dari sebuah single linked list yang non-circular. Program ini memiliki berbagai operasi yang dapat dilakukan pada linked list, seperti menambahkan node di awal, akhir, atau tengah, menghapus node di awal, akhir, atau tengah, mengubah nilai data di awal, akhir, atau tengah, serta menampilkan seluruh daftar node dalam linked list. Program juga menyediakan fungsi untuk memeriksa apakah linked list kosong, menghitung jumlah node dalam linked list, dan menghapus seluruh node dalam linked list. Secara keseluruhan, program ini menyediakan berbagai fungsi dasar untuk mengelola dan memanipulasi linked list secara efisien.

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

```

Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
AyamCicak

```

Deskripsi program

Program ini merupakan implementasi dari sebuah single linked list yang bersifat circular. Program ini memiliki berbagai operasi yang dapat dilakukan pada linked list, seperti menambahkan node di awal, akhir, atau tengah, menghapus node di awal, akhir, atau tengah, serta menampilkan seluruh daftar node dalam linked list. Program juga menyediakan fungsi untuk memeriksa apakah linked list kosong dan menghitung jumlah node dalam linked list. Secara keseluruhan, program ini menyediakan berbagai fungsi dasar untuk mengelola dan memanipulasi linked list yang bersifat circular secara efisien.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

1. Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM mahasiswa, berikut **contoh** tampilan output dari nomor 1:

- 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 Tengah-
```

```
Masukkan Nama  :
```

```
Masukkan NIM   :
```

```
Masukkan Posisi :
```

```
Data telah ditambahkan
```

```
-Tambah Depan-
```

```
Masukkan Nama  :
```

```
Masukkan NIM   :
```

```
Data telah ditambahkan
```

- Tampilan Operasi Hapus:

-Hapus Belakang-

Data (nama mahasiswa yang dihapus) berhasil dihapus

-Hapus Tengah-

Masukkan posisi :

Data (nama mahasiswa yang dihapus) berhasil dihapus

- Tampilan Operasi Ubah:

-Ubah Belakang-

Masukkan nama :

Masukkan NIM :

Data (nama lama) telah diganti dengan data (nama baru)

-Ubah Belakang-

Masukkan nama :

Masukkan NIM :

Masukkan posisi :

Data (nama lama) telah diganti dengan data (nama baru)

- Tampilan Operasi Tampil Data:

DATA MAHASISWA

NAMA	NIM
------	-----

Nama	NIM1
------	------

***Buat tampilan output sebagus dan secantik mungkin sesuai kreatifitas anda masing-masing, jangan terpaku pada contoh output yang diberikan**

2. Setelah membuat menu tersebut, masukkan data sesuai urutan berikut, lalu tampilkan data yang telah dimasukkan. (Gunakan insert depan, belakang atau tengah)

Nama	NIM
Jawad	23300001
[Nama Anda]	[NIM Anda]
Farrel	23300003
Denis	23300005
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

3. Lakukan perintah berikut:

- a) Tambahkan data berikut diantara Farrel dan Denis:

Wati 23300004

- b) Hapus data Denis

- c) Tambahkan data berikut di awal:

Owi 23300000

- d) Tambahkan data berikut di akhir:

David 23300100

- e) Ubah data Udin menjadi data berikut:

Idin 23300045

- f) Ubah data terakhir menjadi berikut:

Lucy 23300101

- g) Hapus data awal

h) Ubah data awal menjadi berikut:

Bagas 2330002

i) Hapus data akhir

j) Tampilkan seluruh data

Source code

```
#include <iostream>
using namespace std;

struct Node
{
    string nama;
    string nim;
    Node *next;
};

class LinkedList
{
private:
    Node *head;

public:
    LinkedList()
    {
        head = nullptr;
    }

    void tambahDepan(string nama, string nim)
    {
        Node *newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = head;
        head = newNode;
    }
};
```

```

        cout << "Data telah ditambahkan" << endl;
    }

void tambahBelakang(string nama, string nim)
{
    Node *newNode = new Node;
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next = nullptr;
    if (head == nullptr)
    {
        head = newNode;
        return;
    }
    Node *temp = head;
    while (temp->next != nullptr)
    {
        temp = temp->next;
    }
    temp->next = newNode;
    cout << "Data telah ditambahkan" << endl;
}

void tambahTengah(string nama, string nim, int posisi)
{
    if (posisi <= 0)
    {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    Node *newNode = new Node;
    newNode->nama = nama;
    newNode->nim = nim;
    Node *temp = head;

```

```

        for (int i = 0; i < posisi - 1; i++)
        {
            if (temp == nullptr)
            {
                cout << "Posisi tidak valid" << endl;
                return;
            }
            temp = temp->next;
        }
        if (temp == nullptr)
        {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        newNode->next = temp->next;
        temp->next = newNode;
        cout << "Data telah ditambahkan" << endl;
    }

    void hapusDepan()
    {
        if (head == nullptr)
        {
            cout << "Linked list kosong" << endl;
            return;
        }
        Node *temp = head;
        head = head->next;
        delete temp;
        cout << "Data berhasil dihapus" << endl;
    }

    void hapusBelakang()
    {

```



```

        if (head == nullptr)
        {
            cout << "Linked list kosong" << endl;
            return;
        }
        if (head->next == nullptr)
        {
            delete head;
            head = nullptr;
            cout << "Data berhasil dihapus" << endl;
            return;
        }
        Node *temp = head;
        while (temp->next->next != nullptr)
        {
            temp = temp->next;
        }
        delete temp->next;
        temp->next = nullptr;
        cout << "Data berhasil dihapus" << endl;
    }

    void hapusTengah(int posisi)
    {
        if (posisi <= 0 || head == nullptr)
        {
            cout << "Linked list kosong atau posisi tidak valid"
<< endl;
            return;
        }
        if (posisi == 1)
        {
            hapusDepan();
            return;
        }
    }

```

```

    }
    Node *temp = head;
    for (int i = 0; i < posisi - 2; i++)
    {
        if (temp->next == nullptr)
        {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        temp = temp->next;
    }
    if (temp->next == nullptr)
    {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    Node *nodeToDelete = temp->next;
    temp->next = temp->next->next;
    delete nodeToDelete;
    cout << "Data berhasil dihapus" << endl;
}

void ubahDepan(string namaBaru, string nimBaru)
{
    if (head == nullptr)
    {
        cout << "Linked list kosong" << endl;
        return;
    }
    head->nama = namaBaru;
    head->nim = nimBaru;
    cout << "Data berhasil diubah" << endl;
}

```

```

void ubahBelakang(string namaBaru, string nimBaru)
{
    if (head == nullptr)
    {
        cout << "Linked list kosong" << endl;
        return;
    }
    Node *temp = head;
    while (temp->next != nullptr)
    {
        temp = temp->next;
    }
    temp->nama = namaBaru;
    temp->nim = nimBaru;
    cout << "Data berhasil diubah" << endl;
}

void ubahTengah(string namaBaru, string nimBaru, int posisi)
{
    if (posisi <= 0 || head == nullptr)
    {
        cout << "Linked list kosong atau posisi tidak valid"
<< endl;
        return;
    }
    Node *temp = head;
    for (int i = 0; i < posisi - 1; i++)
    {
        if (temp == nullptr)
        {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        temp = temp->next;
    }

```

```

    }
    if (temp == nullptr)
    {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    temp->nama = namaBaru;
    temp->nim = nimBaru;
    cout << "Data berhasil diubah" << endl;
}

void hapusList()
{
    Node *current = head;
    Node *next;
    while (current != nullptr)
    {
        next = current->next;
        delete current;
        current = next;
    }
    head = nullptr;
    cout << "Linked list berhasil dihapus" << endl;
}

void tampilkanData()
{
    Node *temp = head;
    cout << "DATA MAHASISWA" << endl;
    cout << "NAMA\tNIM" << endl;
    while (temp != nullptr)
    {
        cout << temp->nama << "\t" << temp->nim << endl;
        temp = temp->next;
    }
}

```

```

    }
}
};

int main()
{
    LinkedList linkedList;
    int choice;
    string nama, nim;
    int posisi;

    do
    {
        cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" <<
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; // Menu untuk hapus
list

        cout << "11. Tampilkan Data" << endl;
        cout << "12. Keluar" << endl;
        cout << "Pilih Operasi : ";
        cin >> choice;

        switch (choice)
        {
            case 1:

```

```
        cout << "-Tambah Depan-" << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.tambahDepan(nama, nim);
        break;
    case 2:
        cout << "-Tambah Belakang-" << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.tambahBelakang(nama, nim);
        break;
    case 3:
        cout << "-Tambah Tengah-" << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.tambahTengah(nama, nim, posisi);
        break;
    case 4:
        cout << "-Ubah Depan-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.ubahDepan(nama, nim);
        break;
    case 5:
```

```

        cout << "-Ubah Belakang-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.ubahBelakang(nama, nim);
        break;
    case 6:
        cout << "-Ubah Tengah-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.ubahTengah(nama, nim, posisi);
        break;
    case 7:
        linkedList.hapusDepan();
        break;
    case 8:
        linkedList.hapusBelakang();
        break;
    case 9:
        cout << "-Hapus Tengah-" << endl;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.hapusTengah(posisi);
        break;
    case 10:
        linkedList.hapusList(); // Hapus List
        break;
    case 11:
        linkedList.tampilkanData();

```

```

        break;
    case 12:
        cout << "Program selesai." << endl;
        break;
    default:
        cout << "Pilihan tidak valid." << endl;
    }
} while (choice != 12);

return 0;
}

```

Screenshoot program

1. 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 Data
12. Keluar
Pilih Operasi :

```

2. Tampilan data berdasarkan inputan

```

DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Diva      2311102006
Farrel     23300003
Denis      23300005
Anis       23300008
Bowo       23300015
Gahar      23300040
Udin       23300048
Ucok       23300050
Budi       23300099

```


3. Lakukan perintah berikut.

- a) Menambahkan data Wati 2330004 diantara Farrel dan Denis

```
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Diva      2311102006
Farrel    23300003
Wati      23300004
Denis     23300005
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
```

- b) Menghapus data Denis

```
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Diva      2311102006
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
```

- c) Menambahkan data Owi 2330000 di awal

```
DATA MAHASISWA
NAMA      NIM
Owi       23300000
Jawad     23300001
Diva      2311102006
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
```

- d) Menambahkan data David 23300100 di akhir

```
DATA MAHASISWA
NAMA      NIM
Owi        2330000
Jawad      23300001
Diva       2311102006
Farrel     23300003
Wati       23300004
Anis       23300008
Bowo       23300015
Gahar      23300040
Udin       23300048
Ucok       23300050
Budi       23300099
David      23300100
```

- e) Mengubah data Udin menjadi Idin 23300045

```
DATA MAHASISWA
NAMA      NIM
Owi        2330000
Jawad      23300001
Diva       2311102006
Farrel     23300003
Wati       23300004
Anis       23300008
Bowo       23300015
Gahar      23300040
Idin       23300045
Ucok       23300050
Budi       23300099
David      23300100
```

- f) Mengubah data terakhir menjadi Lucy 23300101

```
DATA MAHASISWA
NAMA      NIM
Owi        2330000
Jawad      23300001
Diva       2311102006
Farrel     23300003
Wati       23300004
Anis       23300008
Bowo       23300015
Gahar      23300040
Idin       23300045
Ucok       23300050
Budi       23300099
Lucy       23300101
```

g) Menghapus data awal

```
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Diva      2311102006
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
Lucy      23300101
```

h) Mengubah data awal menjadi Bagas 2330002

```
DATA MAHASISWA
NAMA      NIM
Bagas     2330002
Diva      2311102006
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
Lucy      23300101
```

i) Menghapus data akhir

```
DATA MAHASISWA
NAMA      NIM
Bagas     2330002
Diva      2311102006
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
```

j) Menampilkan seluruh data

```
DATA MAHASISWA
NAMA      NIM
Bagas     2330002
Diva      2311102006
Farrel    23300003
Wati      2330004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
```

Deskripsi program

Program ini merupakan implementasi dari sebuah single linked list non-circular yang digunakan untuk menyimpan data mahasiswa berupa nama dan NIM. Program ini menyediakan berbagai operasi yang dapat dilakukan pada linked list, seperti menambahkan node di awal, akhir, atau tengah, mengubah data pada node di awal, akhir, atau tengah, menghapus node di awal, akhir, atau tengah, serta menampilkan seluruh data mahasiswa yang tersimpan dalam linked list. Program juga menyediakan fungsi untuk menghapus seluruh node dalam linked list. Secara keseluruhan, program ini menyediakan antarmuka interaktif bagi pengguna untuk melakukan berbagai operasi pada linked list dengan mudah.

BAB IV

KESIMPULAN

Linked list non circular dan linked list circular merupakan dua struktur data yang memiliki beberapa kesamaan, namun juga memiliki perbedaan signifikan dalam cara kerjanya dan penggunaannya. Linked List Non Circular memiliki node pertama dan terakhir yang tidak saling terhubung, dengan tail selalu berakhir dengan 'NULL', menandakan akhir dari list. Operasi pada Linked List Non Circular melibatkan pengecekan kondisi linked list, penambahan simpul, penghapusan simpul, dan tampilan data.

Di sisi lain, Linked List Circular tidak memiliki akhir yang jelas karena node terakhir terhubung dengan node pertama tanpa menggunakan 'NULL' sebagai penanda akhir. Operasi pada Linked List Circular meliputi pembuatan simpul, pengecekan kondisi linked list, penambahan simpul (baik di depan maupun belakang), penghapusan simpul, dan menampilkan data. Linked List Circular cocok digunakan untuk kasus di mana data perlu diakses secara berulang tanpa henti, seperti dalam daftar putar lagu atau antrian pesan, sementara Linked List Non Circular lebih sesuai untuk aplikasi yang tidak memerlukan pengulangan data secara terus menerus.

DAFTAR PUSTAKA

Asisten Praktikum, *“Modul 4 Linked List Circular dan Non Circular”*, Learning Management System, 2024.

Asisten Praktikum. *“Modul Praktikum Algoritma dan Struktur Data”*, Universitas Negeri Malang, 2016.