

LAPORAN PRAKTIKUM

MODUL IV LINKED LIST CIRCULAR DAN NON CIRCULAR



Disusun oleh:
Rizal Dwi Anggoro
NIM: 2311102034

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

A. Tujuan Praktikum

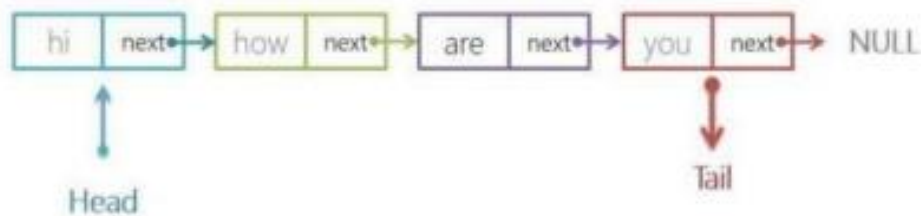
1. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
2. Praktikan dapat membuat linked list circular dan non circular.
3. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

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.



Gambar 1 Single Linked List Non Circular

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, *baru, *bantu, *hapus;

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

```
}  
}
```

3. Pengecekan Kondisi Linked List (Node)

```
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. Menambah 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. Menghapus 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;
    }

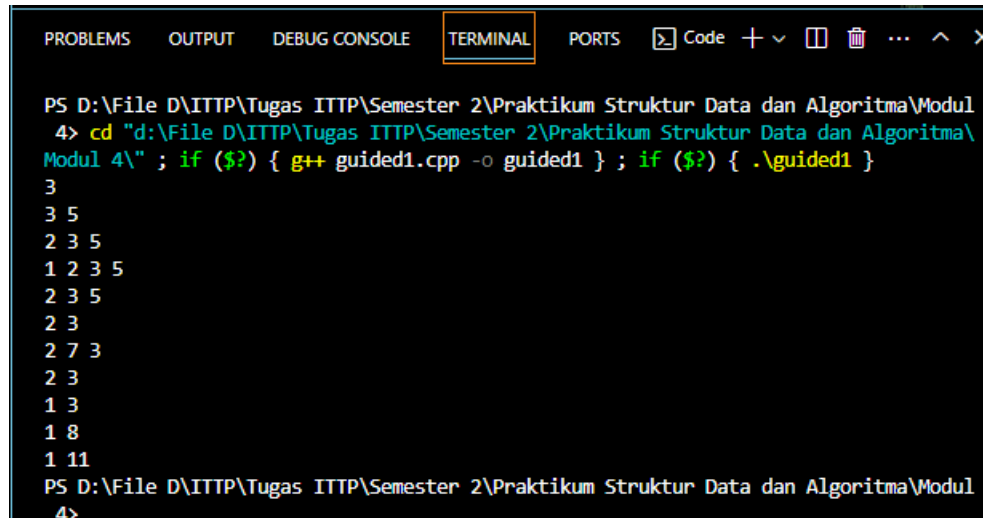
// Tampilkan list
void tampilList()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "Linked list 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



```
PS D:\File D\ITTP\Tugas ITTP\Semester 2\Praktikum Struktur Data dan Algoritma\Modul 4> cd "d:\File D\ITTP\Tugas ITTP\Semester 2\Praktikum Struktur Data dan Algoritma\Modul 4\" ; if ($?) { g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
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
PS D:\File D\ITTP\Tugas ITTP\Semester 2\Praktikum Struktur Data dan Algoritma\Modul 4>
```

Deskripsi program

Program C++ diatas menyediakan fungsi-fungsi untuk menambah, menghapus, dan mengubah nilai elemen di depan, di belakang, atau di tengah linked list. Terdapat juga fungsi untuk menghitung jumlah elemen dalam linked list dan menghapus seluruh isi linked list. Dalam int main(), operasi-operasi ini digunakan untuk menunjukkan cara kerja dari linked list tersebut.

2. Guided 2

Source code

```
#include <iostream>

using namespace std;

// Deklarasi Struct Node

struct Node
{
```

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

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

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

// Pengecekan isi list
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;
    }
}

```

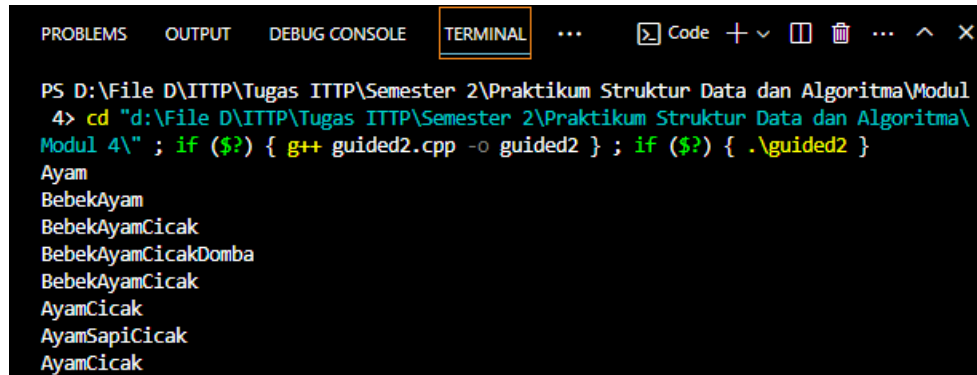


```
        else
        {
            cout << "List masih kosong!" << endl;
        }
    }

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



```
PS D:\File D\ITTP\Tugas ITTP\Semester 2\Praktikum Struktur Data dan Algoritma\Modul 4> cd "d:\File D\ITTP\Tugas ITTP\Semester 2\Praktikum Struktur Data dan Algoritma\Modul 4\" ; if ($?) { g++ guided2.cpp -o guided2 } ; if ($?) { .\guided2 }
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
AyamCicak
```

Deskripsi program

Program C++ di atas merupakan implementasi dari single linked list circular, yang berarti setiap elemen terakhir dalam list memiliki pointer yang menunjuk kembali ke elemen pertama, membentuk lingkaran. Program ini menyediakan fungsi-fungsi untuk menambahkan, menghapus, dan menampilkan elemen-elemen dalam list. Terdapat juga fungsi untuk menghitung jumlah elemen dalam list dan menghapus seluruh isi list.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

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

Source code

```
#include <iostream>
#include <iomanip>

//2311102034_Rizal Dwi Anggoro_IF-11-A

using namespace std;

struct mahasiswa
{
    string nama;
    string nim;
};

struct node
{
    mahasiswa identitas;
    node *next;
};

node *head, *tail, *bantu, *hapus, *before, *baru;

void init()
{
    head = NULL;
    tail = NULL;
    bantu = NULL;
    hapus = NULL;
```

```
        before = NULL;
    }

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

    mahasiswa mintaData()
    {
        system("cls");
        mahasiswa identitas;
        cout << "\nMasukkan Nama\t: ";
        cin.ignore();
        getline(cin, identitas.nama);
        cout << "Masukkan NIM\t: ";
        cin >> identitas.nim;
        return identitas;
    }

    void insertDepan(mahasiswa identitas)
    {
        node *baru = new node;
        baru->identitas.nama = identitas.nama;
```

```

        baru->identitas.nim = identitas.nim;
        baru->next = NULL;
        if (isEmpty() == true)
        {
            head = tail = baru;
            tail->next = NULL;
        }
        else
        {
            baru->next = head;
            head = baru;
        }
        cout << "Data " << identitas.nama << " berhasil diinput!\n";
        system("pause");
    }

void insertBelakang(mahasiswa identitas)

{
    node *baru = new node;
    baru->identitas.nama = identitas.nama;
    baru->identitas.nim = identitas.nim;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

```

```
int hitungList()

{
    int penghitung = 0;
    node *bantu;
    bantu = head;
    while (bantu != NULL)
    {
        penghitung++;
        bantu = bantu->next;
    }
    return penghitung;
}

void insertTengah(mahasiswa identitastitas, int posisi)

{
    node *baru = new node;
    baru->identitas.nama = identitastitas.nama;
    baru->identitas.nim = identitastitas.nim;
    node *bantu;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "posisi diluar jangkauan";
    }
    else if (posisi == 1)
    {
        cout << "Ini bukan posisi tengah\n";
    }
    else
    {
        bantu = head;
        int penghitung = 1;
```

```

        while (penghitung != posisi - 1)
        {
            penghitung++;
            bantu = bantu->next;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void ubahDepan(mahasiswa data)
{
    string namaLama = head->identitas.nama;
    head->identitas.nama = data.nama;
    head->identitas.nim = data.nim;

    cout << "data " << namaLama << " telah diganti dengan data "
         << data.nama << endl;
}

void ubahBelakang(mahasiswa data)
{
    string namaLama = tail->identitas.nama;
    tail->identitas.nama = data.nama;
    tail->identitas.nim = data.nim;
    cout << "data " << namaLama << " telah diganti dengan data "
         << data.nama << endl;
}

void ubahTengah(mahasiswa data)
{

```

```

int posisi;
cout << "\nMasukkan posisi data yang akan diubah : ";
cin >> posisi;
if (posisi < 1 || posisi > hitungList())
{
    cout << "\nPosisi diluar jangkauan\n";
}
else if (posisi == 1)
{
    cout << "\nBukan posisi tengah\n";
}
else
{
    bantu = head;
    int penghitung = 1;
    while (penghitung != posisi)
    {
        penghitung++;
        bantu = bantu->next;
    }
    bantu->identitas.nama = data.nama;
    bantu->identitas.nim = data.nim;
}
}

void tampil()
{
    system("cls");
    node *bantu = head;
    cout << "Nama "
        << " Nim\n";
    while (bantu != NULL)
    {

```



```
        cout << bantu->identitas.nama << " " << bantu->identitas.nim << endl;
        bantu = bantu->next;
    }
}

void hapusDepan()
{
    string dataLama = head->identitas.nama;
    hapus = head;
    if (head != tail)
    {
        head = head->next;
        delete hapus;
    }
    else
    {
        head = tail = NULL;
    }
    cout << "Data " << dataLama << " berhasil dihapus\n";
}

void hapusBelakang()
{
    string dataLama = head->identitas.nama;
    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;
    }
    cout << "Data " << dataLama << " berhasil dihapus\n";
}

void hapusTengah()
{
    tampil();
    cout << endl;
    if (isEmpty() == false)
    {
        back:
        int posisi;
        cout << "Masukkan Posisi yang dihapus : ";
        cin >> posisi;
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "\nPosisi di luar jangkauan!\n";
            system("pause");
            system("cls");
            cout << "Masukkan posisi baru\n";
            goto back;
        }
        else if (posisi == 1 || posisi == hitungList())
        {
            cout << "\nBukan Posisi tengah\n";

```

```

        system("pause");
        // system("cls");
        cout << "Masukkan posisi baru\n";
        goto back;
    }
    else
    {
        bantu = head;
        int penghitung = 1;

        while (penghitung <= posisi)
        {
            if (penghitung == posisi - 1)
            {
                before = bantu;
            }
            if (penghitung == posisi)
            {
                hapus = bantu;
            }
            bantu = bantu->next;
            penghitung++;
        }
        string dataLama = hapus->identitas.nama;
        before->next = bantu;
        delete hapus;

        cout << "\nData " << dataLama << " berhasil dihapus
!\n ";

        system("pause");
    }
}
else
{

```

```

        cout << "\n!!! List Data Kosong !!!\n";
        system("pause");
    }
}

void hapusList()
{
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        delete hapus;
        bantu = bantu->next;
    }
    init();
    cout << "\nsemua data berhasil dihapus\n";
}

int main()
{
    init();
    mahasiswa identitas;
    int operasi, posisi;

    do
    {
        cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR\n\n";
        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 << "0.Exit" << endl;
cout << "\nPilih Operasi : ";
cin >> operasi;

switch (operasi)
{
case 1:
    cout << "tambah depan\n";
    insertDepan(mintaData());
    cout << endl;
    break;
case 2:
    cout << "tambah belakang\n";
    insertBelakang(mintaData());
    cout << endl;
    break;
case 3:
    cout << "tambah tengah\n";
    cout << "nama : ";
    cin >> identitas.nama;
    cout << "NIM : ";
    cin >> identitas.nim;
    cout << "Posisi: ";
    cin >> posisi;

    insertTengah(identitas, posisi);
    cout << endl;
    break;
```

```
case 4:
    cout << "ubah depan\n";
    ubahDepan(mintaData());
    cout << endl;
    break;
case 5:
    cout << "ubah belakang\n";
    ubahBelakang(mintaData());
    cout << endl;
    break;
case 6:
    cout << "ubah tengah\n";
    ubahTengah(mintaData());
    cout << endl;
    break;
case 7:
    cout << "hapus depan\n";
    hapusDepan();
    cout << endl;
    break;
case 8:
    cout << "hapus belakang\n";
    hapusBelakang();
    cout << endl;
    break;
case 9:
    cout << "hapus tengah\n";
    hapusTengah();
    cout << endl;
    break;
case 10:
    cout << "hapus list\n";
    hapusList();
    cout << endl;
```

```

        break;
    case 11:
        tampil();
        cout << endl;
        break;
    case 0:
        cout << "\nEXIT PROGRAM\n";
        break;
    default:
        cout << "\nSalah input operasi\n";
        cout << endl;
        break;
    }
} while (operasi != 0);

return 0;
}

```

Screenshoot program NOMER 1 :



```

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.Exit

Pilih Operasi : 

```

Tampilan Menu

```
Nama Nim
Decul 23111204
Rizal 23111203
Koci 23111202
Mudrik 23111201

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.Exit

Pilih Operasi : 
```

Tambah Depan

```
Nama Nim
Decul 23111204
Rizal 23111203
Demit 23111205
Koci 23111202
Mudrik 23111201

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.Exit

Pilih Operasi : 
```

Tambah Tengah


```
Nama Nim
Decul 23111204
Rizal 23111203
Demit 23111205
Koci 23111202

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.Exit

Pilih Operasi : 
```

Hapus Belakang

```
Nama Nim
Decul 23111204
Rizal 23111203
Koci 23111202

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.Exit

Pilih Operasi : 
```

Hapus Tengah

```
Nama Nim
Decul 23111204
Rizal 23111203
Madun 23111206

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.Exit

Pilih Operasi : 
```

Ubah Belakang

```
Nama Nim
Decul 23111204
SON 23111207
Madun 23111206

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.Exit

Pilih Operasi : 
```

Ubah Tengah & Tampilan Kondisi Data Terakhir

Screenshoot program NOMOR 2 :

```
Nama  Nim
Jawad 23300001
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Denis 23300005
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099

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.Exit

Pilih Operasi : [ ]
```

Tampilan Data Nomor 2

Screenshoot program NOMOR 3 :

```
Nama  Nim
Jawad 23300001
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Wati 23300004
Denis 23300005
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099

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.Exit

Pilih Operasi : [ ]
```

a. tambah data Wati 23300004 diantara Farrel dan Denis

```

Nama  Nim
Jawad 23300001
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099

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.Exit

Pilih Operasi : 

```

b. Hapus data denis

```

Nama  Nim
Owi 23300000
Jawad 23300001
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099

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.Exit

Pilih Operasi : 

```

c. Tambah data Owi 23300000 di awal

```
Nama Nim
Owi 2330000
Jawad 23300001
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Wati 2330004
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099
David 23300100
```

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.Exit

Pilih Operasi :

d. Tambah data David 23300100 di akhir

```
Nama Nim
Owi 2330000
Jawad 23300001
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Wati 2330004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
David 23300100
```

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.Exit

Pilih Operasi :

e. Ubah data Udin menjadi Idin 23300045

```
Nama  Nim
Owi  2330000
Jawad 23300001
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Wati 2330004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
Lucy 23300101

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.Exit

Pilih Operasi : 
```

f. Ubah data terakhir menjadi Lucy 23300101

```
Nama  Nim
Jawad 23300001
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Wati 2330004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
Lucy 23300101

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.Exit

Pilih Operasi : 
```

g. Hapus data awal

```
Nama Nim
Bagas 2330002
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Wati 2330004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
Lucy 23300101
```

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.Exit

Pilih Operasi :

h. Ubah data awal menjadi Bagas 2330002

```
Nama Nim
Bagas 2330002
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Wati 2330004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
```

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.Exit

Pilih Operasi :

i. Hapus data terakhir

```
Nama Nim
Bagas 2330002
Rizal Dwi Anggoro 2311102034
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099

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.Exit

Pilih Operasi : 11
```

j. Tampilan seluruh data

Deskripsi program

Program C++ diatas merupakan implementasi dari single linked list non-circular untuk menyimpan data mahasiswa yang di input dari user. Struktur node digunakan untuk merepresentasikan node dalam linked list, dan variabel global seperti head, tail, dan pointer. Program menggunakan loop do-while untuk menampilkan menu interaktif kepada pengguna, yang memungkinkan mereka untuk melakukan operasi seperti menambah, mengubah, atau menghapus data mahasiswa, serta menampilkan seluruh data yang tersimpan.

BAB IV

KESIMPULAN

Dari praktikum Linked List Non-Circular yang dapat saya simpulkan adalah bahwa struktur data ini menyediakan fleksibilitas dalam penyimpanan dan pengelolaan data, terutama ketika data yang dimasukkan memiliki panjang atau jenis yang beragam. Dengan menggunakan linked list, kita dapat dengan mudah menambah, mengubah, atau menghapus data tanpa perlu membatasi diri pada urutan tertentu sehingga lebih efisien.