

LAPORAN PRAKTIKUM

MODUL IV

“Linked List Circular dan Non Circular”



Disusun oleh:

Shiva Indah Kurnia
NIM 2311102035

Dosen Pengampu:

Wahyu Andi Saputra, SPd., M.Eng

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

TUJUAN PRAKTIKUM

Setelah melakukan praktikum ini diharapkan mahasiswa dapat:

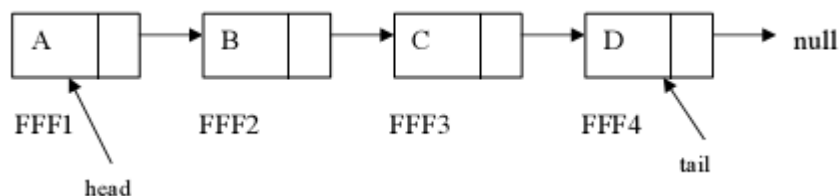
1. Memahami konsep dasar Linked List Circular dan Non Circular.
2. Mampu membuat program dengan mengaplikasikan Linked List Circular dan Non Circular.
3. Mampu menerapkan konsep Linked List Circular dan Non Circular dalam studi kasus nyata.

DASAR TEORI

Linked list dapat dibagi menjadi dua jenis, yaitu linked list circular dan non-circular.

1. Linked list non-circular

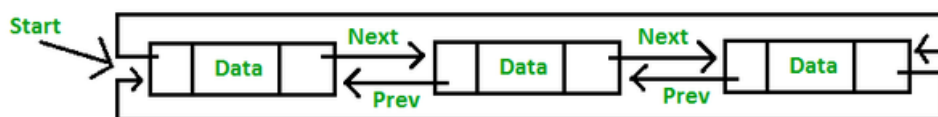
Pada linked list non-circular, node terakhir tidak menunjuk pada simpul pertama. Artinya, linked list ini memiliki simpul terakhir yang menunjuk ke null atau tidak menunjuk ke simpul mana pun. Dalam linked list non-circular, traversal atau penelusuran dari awal hingga akhir dapat dilakukan dengan mudah dengan mengikuti alamat setiap node.



2. Linked list circular

Pada linked list circular, simpul terakhir menunjuk pada simpul pertama. Artinya, linked list ini membentuk sebuah lingkaran, dan traversal dapat dimulai dari mana saja dalam linked list ini. Linked list circular memiliki kelebihan yaitu memungkinkan untuk melakukan traversal dari mana saja dan tidak memerlukan operasi tambahan untuk mengembalikan pointer ke simpul awal.

Dalam implementasinya, linked list circular dan non-circular memiliki beberapa perbedaan. Salah satunya adalah pada operasi penambahan atau penghapusan node, dimana linked list circular memerlukan operasi tambahan untuk mengubah pointer pada simpul terakhir agar menunjuk pada simpul pertama. Sedangkan linked list non-circular tidak memerlukan operasi tersebut karena simpul terakhirnya tidak menunjuk ke simpul pertama.



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;
    }

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

// Hapus Depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
}

else

```

```

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

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

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

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

```

```

{
    cout << "Posisi bukan posisi tengah" << endl;
}
else
{
    int nomor = 1;
    bantu = head;
    while (nomor <= posisi)
    {
        if (nomor == posisi - 1)
        {
            sebelum = bantu;
        }
        if (nomor == posisi)
        {
            hapus = bantu;
        }
        bantu = bantu->next;
        nomor++;
    }
    sebelum->next = bantu;
    delete hapus;
}
}

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

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

```



```

        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
        }
        else
        {
            cout << "Posisi bukan posisi tengah" << endl;

            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

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

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

// Hapus List
void clearList()
{
    Node *bantu, *hapus;

```

```

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

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

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

    int main()
    {
        init();
        insertDepan(3);
        tampil();
        insertBelakang(5);
        tampil();
        insertDepan(2);
        tampil();
        insertDepan(1);
        tampil();
        hapusDepan();
        tampil();
    }

```

```

        hapusBelakang();
        tampil();
        insertTengah(7, 2);
        tampil();
        hapusTengah(2);
        tampil();
        ubahDepan(1);
        tampil();
        ubahBelakang(8);
        tampil();
        ubahTengah(11, 2);
        tampil();

        return 0;
    }

```

Screenshot Program

```

PS C:\Laprak Sem 2 Strukdat\Guided 1> g++ cc.cpp
PS C:\Laprak Sem 2 Strukdat\Guided 1> ./a
3
35
235
1235
235
23
273
23
13
18
Posisi bukan posisi tengah
111
PS C:\Laprak Sem 2 Strukdat\Guided 1> 

```

Deskripsi Program

Program diatas merupakan implementasi dari linked list (non-circular). `init()` akan menginisialisasi pointer `head` dan `tail` menjadi `NULL`. Kemudian `isEmpty()` memeriksa apakah daftar kosong atau tidak. Fungsi `insertDepan(int nilai)` menyisipkan node baru di awal daftar. Fungsi `insertBelakang(int nilai)` menyisipkan node baru di akhir daftar. Fungsi `hitungList()` mengembalikan jumlah node dalam daftar. Fungsi `insertTengah(int data, int posisi)` menyisipkan node baru pada posisi

yang ditentukan dalam daftar. Fungsi `hapusDepan()` menghapus node pertama dari daftar. Fungsi `hapusBelakang()` menghapus node terakhir dari daftar. Fungsi `hapusTengah(int posisi)` menghapus node dari posisi yang ditentukan dalam daftar. Fungsi `ubahDepan(int data)` memodifikasi nilai node pertama dalam daftar. Fungsi `ubahTengah(int data, int posisi)` memodifikasi nilai node pada posisi yang ditentukan dalam daftar. Fungsi `ubahBelakang(int data)` memodifikasi nilai node terakhir dalam daftar. Fungsi `clearList()` menghapus semua node dalam daftar. Fungsi `tampil()` menampilkan nilai dari semua node dalam daftar. Selanjutnya program akan mengembalikan nilai 0

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;
    }
}

```

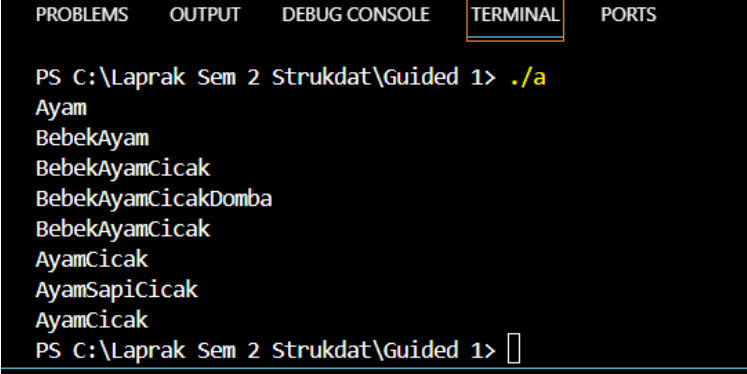


```

    }
    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;
}

```

Screenshot Program



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Laprak Sem 2 Strukdat\Guided 1> ./a
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
AyamCicak
PS C:\Laprak Sem 2 Strukdat\Guided 1> 
```

Deskripsi Program

Program di atas adalah contoh implementasi linked list circular. Pada program ini, terdapat beberapa fungsi untuk memanipulasi linked list circular, seperti ``init()`` inisialisasi head dan tail linked list menjadi NULL, ``isEmpty()`` fungsi untuk memeriksa apakah linked list kosong atau tidak, Mengembalikan nilai 1 jika kosong dan 0 jika tidak kosong. ``buatNode(string data)`` membuat node baru dengan data yang diberikan. ``hitungList()``: menghitung jumlah elemen dalam linked list. ``insertDepan(string data)``: menambahkan node baru pada awal linked list. ``insertBelakang(string data)``: menambahkan node baru pada akhir linked list. ``insertTengah(string data, int posisi)``: menambahkan node baru pada posisi tertentu dalam linked list. ``hapusDepan()``: menghapus node pada awal linked list. ``hapusBelakang()``: menghapus node pada akhir linked list. ``hapusTengah(int posisi)``: menghapus node pada posisi tertentu dalam linked list. ``clearList()``: menghapus seluruh elemen dalam linked list. ``tampil()``: menampilkan seluruh data dalam linked list. Pada program ini, fungsi-fungsi tersebut dijalankan dalam main function dengan cara memanggilnya dan menyertakan argumen yang diperlukan. Beberapa contoh penggunaan fungsi tersebut antara lain menambahkan data pada linked list, menghapus data pada linked list, serta menampilkan seluruh data dalam linked list.

UNGUIDED

1. Unguided 1

Source Code

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

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

class LinkedListCircular {
private:
    Mahasiswa* head;

public:
    LinkedListCircular() {
        head = nullptr;
    }

    // Menambahkan mahasiswa di depan
    void tambahDepan(string nama, string nim) {
        Mahasiswa* newMahasiswa = new Mahasiswa;
        newMahasiswa->nama = nama;
        newMahasiswa->nim = nim;

        if (head == nullptr) {
            head = newMahasiswa;
            newMahasiswa->next = head;
        } else {
            Mahasiswa* last = head;
            while (last->next != head) {
                last = last->next;
            }
            newMahasiswa->next = head;
            last->next = newMahasiswa;
            head = newMahasiswa;
        }
        cout << "Mahasiswa berhasil ditambahkan di depan." <<
endl;
    }
}
```

```

// Menambahkan mahasiswa di belakang
void tambahBelakang(string nama, string nim) {
    Mahasiswa* newMahasiswa = new Mahasiswa;
    newMahasiswa->nama = nama;
    newMahasiswa->nim = nim;

    if (head == nullptr) {
        head = newMahasiswa;
        newMahasiswa->next = head;
    } else {
        Mahasiswa* last = head;
        while (last->next != head) {
            last = last->next;
        }
        last->next = newMahasiswa;
        newMahasiswa->next = head;
    }
    cout << "Mahasiswa berhasil ditambahkan di belakang." <<
endl;
}

// Menambahkan mahasiswa di tengah
void tambahTengah(string nama, string nim, int posisi) {
    if (posisi <= 0) {
        cout << "Posisi harus lebih dari 0." << endl;
        return;
    }
    Mahasiswa* newMahasiswa = new Mahasiswa;
    newMahasiswa->nama = nama;
    newMahasiswa->nim = nim;

    if (head == nullptr) {
        head = newMahasiswa;
        newMahasiswa->next = head;
        cout << "Mahasiswa berhasil ditambahkan di tengah."
<< endl;
        return;
    }

    Mahasiswa* temp = head;
    int count = 1;
    while (count < posisi - 1 && temp->next != head) {
        temp = temp->next;
        count++;
    }

```

```

    }

    if (count < posisi - 1) {
        cout << "Posisi terlalu besar." << endl;
        return;
    }

    newMahasiswa->next = temp->next;
    temp->next = newMahasiswa;
    cout << "Mahasiswa berhasil ditambahkan di tengah." <<
endl;
}

// Ubah data mahasiswa di depan
void ubahDepan(string nama, string nim) {
    if (head == nullptr) {
        cout << "Linked List Kosong." << endl;
        return;
    }
    head->nama = nama;
    head->nim = nim;
    cout << "Data mahasiswa di depan berhasil diubah." <<
endl;
}

// Ubah data mahasiswa di belakang
void ubahBelakang(string nama, string nim) {
    if (head == nullptr) {
        cout << "Linked List Kosong." << endl;
        return;
    }
    Mahasiswa* temp = head;
    while (temp->next != head) {
        temp = temp->next;
    }
    temp->nama = nama;
    temp->nim = nim;
    cout << "Data mahasiswa di belakang berhasil diubah." <<
endl;
}

// Ubah data mahasiswa di tengah
void ubahTengah(string nama, string nim, int posisi) {
    if (head == nullptr) {
        cout << "Linked List Kosong." << endl;

```

```

        return;
    }
    Mahasiswa* temp = head;
    int count = 1;
    while (count < posisi && temp->next != head) {
        temp = temp->next;
        count++;
    }
    if (count != posisi) {
        cout << "Posisi terlalu besar." << endl;
        return;
    }
    temp->nama = nama;
    temp->nim = nim;
    cout << "Data mahasiswa di posisi " << posisi << "
berhasil diubah." << endl;
}

// Hapus data mahasiswa di depan
void hapusDepan() {
    if (head == nullptr) {
        cout << "Linked List Kosong." << endl;
        return;
    }
    if (head->next == head) {
        delete head;
        head = nullptr;
    } else {
        Mahasiswa* temp = head;
        while (temp->next != head) {
            temp = temp->next;
        }
        Mahasiswa* del = head;
        head = head->next;
        temp->next = head;
        delete del;
    }
    cout << "Data mahasiswa di depan berhasil dihapus." <<
endl;
}

// Hapus data mahasiswa di belakang
void hapusBelakang() {
    if (head == nullptr) {
        cout << "Linked List Kosong." << endl;

```

```

        return;
    }
    if (head->next == head) {
        delete head;
        head = nullptr;
    } else {
        Mahasiswa* temp = head;
        Mahasiswa* prev = nullptr;
        while (temp->next != head) {
            prev = temp;
            temp = temp->next;
        }
        prev->next = head;
        delete temp;
    }
    cout << "Data mahasiswa di belakang berhasil dihapus."
<< endl;
}

// Hapus data mahasiswa di tengah
void hapusTengah(int posisi) {
    if (head == nullptr) {
        cout << "Linked List Kosong." << endl;
        return;
    }
    if (posisi <= 0) {
        cout << "Posisi harus lebih dari 0." << endl;
        return;
    }
    Mahasiswa* temp = head;
    Mahasiswa* prev = nullptr;
    int count = 1;
    while (count < posisi && temp->next != head) {
        prev = temp;
        temp = temp->next;
        count++;
    }
    if (count != posisi) {
        cout << "Posisi terlalu besar." << endl;
        return;
    }
    if (temp == head) {
        hapusDepan();
        return;
    }
}

```

```

        prev->next = temp->next;
        delete temp;
        cout << "Data mahasiswa di posisi " << posisi << "
berhasil dihapus." << endl;
    }

    // Hapus semua data mahasiswa
    void hapusList() {
        if (head == nullptr) {
            cout << "Linked List Kosong" << endl;
            return;
        }
        Mahasiswa* current = head;
        Mahasiswa* next = nullptr;
        while (current->next != head) {
            next = current->next;
            delete current;
            current = next;
        }
        delete current;
        head = nullptr;
        cout << "Semua data mahasiswa berhasil dihapus." <<
endl;
    }

    // Menampilkan daftar mahasiswa
    void tampilkan() {
        if (head == nullptr) {
            cout << "Linked List Kosong." << endl;
            return;
        }
        Mahasiswa* temp = head;
        do {
            cout << "Nama: " << temp->nama << ", NIM: " << temp-
>nim << endl;
            temp = temp->next;
        } while (temp != head);
    }
};

int main() {
    LinkedListCircular list;
    int choice;
    string nama, nim;
    int posisi;

```



```

do {
    cout << "-----";
    cout << "\nLIST DATA DAN NIM MAHASISWA\n";
    cout << "-----\n";
    cout << "Menu :\n";
    cout << "1. Tambah Depan\n";
    cout << "2. Tambah Belakang\n";
    cout << "3. Tambah Tengah\n";
    cout << "4. Ubah Depan\n";
    cout << "5. Ubah Belakang\n";
    cout << "6. Ubah Tengah\n";
    cout << "7. Hapus Depan\n";
    cout << "8. Hapus Belakang\n";
    cout << "9. Hapus Tengah\n";
    cout << "10. Hapus List\n";
    cout << "11. Tampilkan Data\n";
    cout << "0. Keluar\n";
    cout << "Pilih Operasi: ";
    cin >> choice;

    switch (choice) {
        case 1:
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            list.tambahDepan(nama, nim);
            break;
        case 2:
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            list.tambahBelakang(nama, nim);
            break;
        case 3:
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            cout << "Masukkan Posisi: ";
            cin >> posisi;
            list.tambahTengah(nama, nim, posisi);
            break;
    }
}

```

```

case 4:
    cout << "Masukkan Nama: ";
    cin >> nama;
    cout << "Masukkan NIM: ";
    cin >> nim;
    list.ubahDepan(nama, nim);
    break;
case 5:
    cout << "Masukkan Nama: ";
    cin >> nama;
    cout << "Masukkan NIM: ";
    cin >> nim;
    list.ubahBelakang(nama, nim);
    break;
case 6:
    cout << "Masukkan Nama: ";
    cin >> nama;
    cout << "Masukkan NIM: ";
    cin >> nim;
    cout << "Masukkan Posisi: ";
    cin >> posisi;
    list.ubahTengah(nama, nim, posisi);
    break;
case 7:
    list.hapusDepan();
    break;
case 8:
    list.hapusBelakang();
    break;
case 9:
    cout << "Masukkan Posisi: ";
    cin >> posisi;
    list.hapusTengah(posisi);
    break;
case 10:
    list.hapusList();
    break;
case 11:
    list.tampilkan();
    break;
case 0:
    cout << "Terima kasih:)" << endl;
    break;
default:
    cout << "Maaf, pilihan tidak valid." << endl;

```

```

        break;
    }
} while (choice != 0);

return 0;
}

```

Screenshot Program

- Tampilan Menu

```

-----
LIST DATA DAN NIM MAHASISWA
-----

Menu :
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
0. Keluar
Pilih Operasi: 

```

- Tampilan Operasi Tambah

```

-----
LIST DATA DAN NIM MAHASISWA
-----

Menu :
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
0. Keluar
Pilih Operasi: 1
Masukkan Nama: Shiva
Masukkan NIM: 2311102035
Mahasiswa berhasil ditambahkan di depan.

```

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 2  
Masukkan Nama: Indah  
Masukkan NIM: 231110980  
Mahasiswa berhasil ditambahkan di belakang.
```

- Tampilan Operasi Hapus

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 8  
Data mahasiswa di belakang berhasil dihapus.
```

- Tampilan Operasi Ubah

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 4  
Masukkan Nama: Kurnia  
Masukkan NIM: 231111908  
Data mahasiswa di depan berhasil diubah.
```

LIST DATA DAN NIM MAHASISWA

Menu :

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
0. Keluar

Pilih Operasi: 11

Nama: Cipa, NIM: 231111987

- Tampilan Operasi Tampilkan List

LIST DATA DAN NIM MAHASISWA

Menu :

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
0. Keluar

Pilih Operasi: 11

Nama: Kurnia, NIM: 231111908

Nama: Cipa, NIM: 231111987

LIST DATA DAN NIM MAHASISWA

Menu :

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
0. Keluar

Pilih Operasi: 11

Nama: Jawad, NIM: 23300001

Nama: Shiva, NIM: 2311102035

Nama: Denis, NIM: 23300005

Nama: Farrel, NIM: 23300003

Nama: Anis, NIM: 23300008

Nama: Bowo, NIM: 23300015

Nama: Gahar, NIM: 23000040

Nama: Udin, NIM: 23300048

Nama: Ukok, NIM: 23300050

Nama: Budi, NIM: 23300099

- **Tambah Wati**

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 3  
Masukkan Nama: Wati  
Masukkan NIM: 233004  
Masukkan Posisi: 4  
Mahasiswa berhasil ditambahkan di tengah.
```

- **Hapus Data Denis**

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 9  
Masukkan Posisi: 3  
Data mahasiswa di posisi 3 berhasil dihapus.
```

- **Tambah data Owi**

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 1  
Masukkan Nama: Owi  
Masukkan NIM: 2330000  
Mahasiswa berhasil ditambahkan di depan.
```

- **Tambah data david di akhir**

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 2  
Masukkan Nama: David  
Masukkan NIM: 233000100  
Mahasiswa berhasil ditambahkan di belakang.
```

- **Ganti nama udin jadi idin**

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 6  
Masukkan Nama: Idin  
Masukkan NIM: 23300045  
Masukkan Posisi: 9  
Data mahasiswa di posisi 9 berhasil diubah.
```

- **Tambah data lucy**

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 5  
Masukkan Nama: Lucy  
Masukkan NIM: 23300101  
Data mahasiswa di belakang berhasil diubah.
```


- **Hapus data awal**

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 7  
Data mahasiswa di depan berhasil dihapus.
```

- **Ubah data awal menjadi bagus**

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 4  
Masukkan Nama: Bagus  
Masukkan NIM: 2330002  
Data mahasiswa di depan berhasil diubah.
```

- **Hapus data akhir**

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 8  
Data mahasiswa di belakang berhasil dihapus.
```

- **Tampilkan seluruh data**

```
-----  
LIST DATA DAN NIM MAHASISWA  
-----  
Menu :  
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  
0. Keluar  
Pilih Operasi: 11  
Nama: Bagus, NIM: 2330002  
Nama: Shiva, NIM: 2311102035  
Nama: Farrel, NIM: 23300003  
Nama: Wati, NIM: 233004  
Nama: Anis, NIM: 23300008  
Nama: Bowo, NIM: 23300015  
Nama: Gahar, NIM: 2300040  
Nama: Idin, NIM: 23300045  
Nama: Ucok, NIM: 23300050  
Nama: Budi, NIM: 23300099
```

Deskripsi Program

Program diatas adalah program yang mengimplementasikan struktur dari linked list non circular. Ada beberapa fungsi yang dapat Anda gunakan untuk menambah, mengedit, dan menghapus node dalam Linked List. Linked List diimplementasikan menggunakan simpul pohon dengan tiga anggota: Integer NIM, string nama, dan pointer ke node berikutnya dalam daftar. Indikator head dan tail juga dilaporkan secara global. **Fungsi init()** menginisialisasi pointer kepala dan ekor ke NULL. **Fungsi is Empty()** memeriksa apakah penunjuk utama adalah NULL, yang merupakan daftar kosong.

Fungsi insertDepan() menambahkan node baru ke awal daftar. Membuat node baru dengan nilai NIM dan nama yang ditentukan dan menetapkan pointer berikut ke NULL. Jika daftar kosong, pointer kepala dan ekor diatur ke node baru. Jika tidak, penunjuk simpul baru berikutnya ditetapkan sebagai kepala saat ini dan penunjuk kepala diperbarui untuk menunjuk ke simpul baru.

Fungsi insertBelakang() menambahkan simpul baru ke akhir daftar. Membuat node baru dengan nilai NIM dan nama yang ditentukan dan menetapkan pointer berikut ke NULL. Jika daftar kosong, pointer kepala dan ekor diatur ke node baru. Jika tidak, penunjuk berikutnya ke simpul terminal saat ini diatur ke simpul baru, dan penunjuk terminal diperbarui untuk menunjuk ke simpul baru.

Fungsi JumlahList() menghitung jumlah node dalam daftar dengan mengunjungi setiap node dan menambahkan penghitung.

Fungsi insertCenter() menyisipkan node baru ke dalam daftar pada posisi yang ditentukan. Jika lokasi yang ditentukan berada di luar jangkauan daftar, pesan kesalahan dikeluarkan. Jika situs adalah simpul pertama, itu tidak dianggap sebagai hub yang valid dan pesan kesalahan akan ditampilkan. Sebaliknya, simpul baru dibuat dan disisipkan setelah simpul pada posisi yang ditentukan.

Fungsi ubahDepan() memperbarui nilai NIM dan Name dari node pertama dalam daftar jika daftar tidak kosong. Fungsi ubahBelakang() memperbarui nilai NIM dan Name dari node terakhir dalam daftar jika daftar tidak kosong.

Fungsi ubahTengah() memperbarui nilai NIM dan nama node pada posisi daftar yang ditentukan jika daftar tidak kosong dan posisinya valid. Jika daftar tidak kosong, **fungsi hapusDepan()** menghapus node pertama

dalam daftar. Jika daftar tidak kosong, **fungsi hapusBelakang()** menghapus simpul terakhir dari daftar. **Fungsi hapusTengah()** menghapus sebuah node dari daftar yang ditentukan jika daftar tidak kosong dan posisinya valid.

KESIMPULAN

Linked List Non-Circular adalah linked List yang simpul terakhirnya menunjuk ke NULL. Daftar tertaut ini hanya dapat diputar dari awal hingga akhir. Untuk menangani daftar tertaut non-lingkaran, didalam contoh yang telah diaplikasikan terdapat contoh seperti fungsi seperti insertForward(), insertBack(), insertMiddle(), removeForward(), removeBack() dan removeMiddle() yang diaplikasikan kedalam unguided. Sedangkan Linked List Circular adalah Linked List di mana simpul terakhir menunjuk kembali ke simpul pertama. Daftar tertaut ini dapat diulang dari awal hingga akhir atau dari akhir ke awal. Linked List Circular memiliki fungsi yang sama seperti Linked List Non Circular, tetapi dalam Linked List Circular bisa menggunakan fungsi tambahan seperti insertAfter(), insertBefore(), deleteAfter(), dan removeBefore() untuk menambah atau menghapus node pada titik tertentu dalam list.

Perbedaan antara kedua jenis Linked List tersebut adalah bahwa Linked List Circular lebih baik dalam melintasi List di kedua arah, sedangkan Linked List Non-Circular hanya dapat dilalui dalam satu arah. Namun, Linked List Circular juga lebih sulit untuk diterapkan dan loop tak terbatas lebih sulit dihindari, tetapi dapat berguna dalam beberapa kasus penggunaan tertentu.