

LAPORAN PRAKTIKUM

MODUL IV LINKED LIST CIRCULAR DAN NON CIRCULAR



Disusun oleh:
Imelda Fajar Awalina Crisyanti
NIM: 2311102004

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

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023**

BAB I

TUJUAN PRAKTIKUM

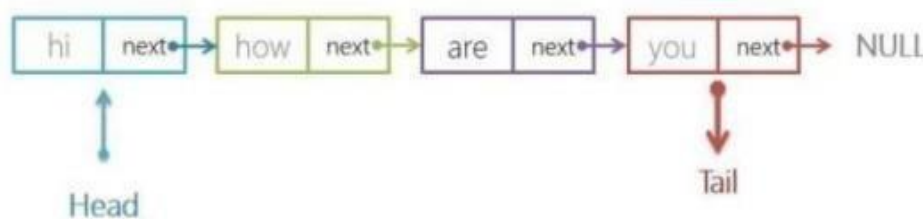
- a.** Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
- b.** Praktikan dapat membuat linked list circular dan non circular.
- c.** Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

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.

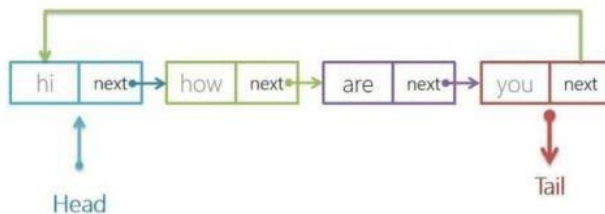


Gambar 1 *Single Linked List Non Circular*

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.



Gambar 2 *Single Linked List Circular*

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

```

[Running] cd "c:\Users\ASUS Vivo
Data\Modul 4\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

```

Deskripsi program

Program ini adalah implementasi dari struktur data linked list menggunakan bahasa pemrograman C++. Dalam linked list ini, setiap elemen memiliki dua bagian: data (yang berisi nilai) dan pointer ke elemen berikutnya. Program ini menyediakan sejumlah fungsi untuk mengelola linked list, seperti menambahkan elemen di depan (insertDepan), menambahkan elemen di belakang (insertBelakang), menambahkan elemen di tengah (insertTengah), menghapus elemen di depan (hapusDepan), menghapus elemen di belakang (hapusBelakang), menghapus elemen di tengah (hapusTengah), mengubah nilai elemen di depan (ubahDepan), mengubah nilai elemen di tengah (ubahTengah), mengubah nilai elemen di belakang (ubahBelakang), menghitung jumlah elemen dalam linked list (hitungList), mengosongkan linked list (clearList), dan menampilkan isi linked list (tampilList).

Fungsi-fungsi tersebut memungkinkan pengguna untuk melakukan operasi dasar pada linked list, seperti menambah, menghapus, mengubah, menghitung jumlah elemen, dan menampilkan isi linked list.

Dalam fungsi main(), program diinisialisasi dengan memanggil fungsi init(), kemudian dilakukan serangkaian operasi pada linked list seperti menambahkan elemen di depan dan di belakang, menghapus elemen di depan dan di belakang, menambahkan elemen di tengah, menghapus elemen di tengah, mengubah nilai elemen, dan menampilkan isi linked list setelah setiap operasi dilakukan.

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;
}

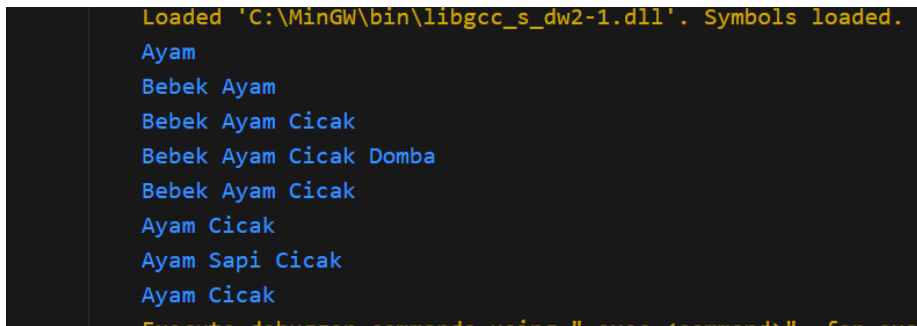
//Tampilkan List
void tampil(){
    if (isEmpty() == 0){
        tail = head;
        do {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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



A screenshot of a program's output in a terminal window. The output shows a linked list of animals: Ayam, Bebek Ayam, Bebek Ayam Cicak, Bebek Ayam Cicak Domba, Bebek Ayam Cicak, Ayam Cicak, Ayam Sapi Cicak, and Ayam Cicak. The text is displayed in a monospaced font with some color coding (yellow for the first line, blue for the others).

```
Loaded 'C:\MinGW\bin\libgcc_s_dw2-1.dll'. Symbols loaded.
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak
Ayam Cicak
Execute debugger commands using "exec <command>" for exec
```

Deskripsi program

Program ini adalah implementasi dari struktur data linked list sirkular menggunakan bahasa pemrograman C++. Dalam linked list ini, setiap elemen memiliki dua bagian: data (yang berisi nilai) dan pointer ke elemen berikutnya. Namun, dalam implementasi ini, elemen terakhir menunjuk kembali ke elemen pertama, membentuk lingkaran.

Program ini menyediakan sejumlah fungsi untuk mengelola linked list, seperti menambahkan elemen di depan (insertDepan), menambahkan elemen di belakang (insertBelakang), menambahkan elemen di tengah (insertTengah), menghapus elemen di depan (hapusDepan), menghapus elemen di belakang (hapusBelakang), menghapus elemen di tengah (hapusTengah), mengosongkan linked list (clearList), dan menampilkan isi linked list (tampil).

Fungsi-fungsi tersebut memungkinkan pengguna untuk melakukan operasi dasar pada linked list sirkular, seperti menambah, menghapus, mengubah, menghitung jumlah elemen, dan menampilkan isi linked list.

Dalam fungsi main(), program diinisialisasi dengan memanggil fungsi init(), kemudian dilakukan serangkaian operasi pada linked list seperti menambahkan elemen di depan dan di belakang, menghapus elemen di depan dan di belakang, menambahkan elemen di tengah, menghapus elemen di tengah, dan menampilkan isi linked list setelah setiap operasi dilakukan.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

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;
    }
    string namaHapus = head->nama;
    Node *temp = head;
    head = head->next;
    delete temp;
    cout << "Data " << namaHapus << " berhasil dihapus" << endl;
}
void hapusBelakang()
{
    if (head == nullptr)
    {
        cout << "Linked list kosong" << endl;
        return;
    }
    Node *temp = head;
    Node *prev = nullptr;
    while (temp->next != nullptr)
    {
        prev = temp;
        temp = temp->next;
    }
    string namaHapus = temp->nama;
    if (prev != nullptr)
    {
        prev->next = nullptr;
    }
    else
    {
        head = nullptr;
    }
    delete temp;
    cout << "Data " << namaHapus << " berhasil dihapus" << endl;
}
void hapusTengah(int posisi)
{
    if (posisi <= 0 || head == nullptr)
    {
        cout << "Linked list kosong atau posisi tidak valid" <<
endl;
        return;
    }
    Node *temp = head;

```

```

Node *prev = nullptr;
for (int i = 0; i < posisi - 1; i++)
{
    if (temp == nullptr)
    {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    prev = temp;
    temp = temp->next;
}
if (temp == nullptr)
{
    cout << "Posisi tidak valid" << endl;
    return;
}
string namaHapus = temp->nama;
if (prev != nullptr)
{
    prev->next = temp->next;
}
else
{
    head = temp->next;
}
delete temp;
cout << "Data " << namaHapus << " berhasil dihapus" << endl;
}
void ubahDepan(string namaBaru, string nimBaru)
{
    if (head == nullptr)
    {
        cout << "Linked list kosong" << endl;
        return;
    }
    string namaLama = head->nama;
    string nimLama = head->nim;
    head->nama = namaBaru;
    head->nim = nimBaru;
    cout << "Data " << namaLama << " telah diganti denga data " <<
head->nama << endl;
}
void ubahBelakang(string namaBaru, string nimBaru)

```

```

{
    if (head == nullptr)
    {
        cout << "Linked list kosong" << endl;
        return;
    }
    Node *temp = head;
    string namaLama, nimLama;
    while (temp->next != nullptr)
    {
        temp = temp->next;
    }
    namaLama = temp->nama;
    nimLama = temp->nim;
    temp->nama = namaBaru;
    temp->nim = nimBaru;
    cout << "Data " << namaLama << " telah diganti denga data " <<
temp->nama << 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;
    }
    string namaLama = temp->nama;

```

```

        string nimLama = temp->nim;
        temp->nama = namaBaru;
        temp->nim = nimBaru;
        cout << "Data " << namaLama << " telah diganti denga data " <<
temp->nama << 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 << "*****" << endl;
        cout << "1. Tambah Depan" << endl;
        cout << "2. Tambah Belakang" << endl;
    }
}

```



```

cout << "3. Tambah Tengah" << endl;
cout << "4. Ubah Depan" << endl;
cout << "5. Ubah Belakang" << endl;
cout << "6. Ubah Tengah" << endl;
cout << "7. Hapus Depan" << endl;
cout << "8. Hapus Belakang" << endl;
cout << "9. Hapus Tengah" << endl;
cout << "10. Hapus List" << endl;
cout << "11. TAMPILKAN" << endl;
cout << "0. 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:
        cout << "-Hapus Depan-" << endl;
        linkedList.hapusDepan();
        break;
    case 8:
        cout << "-Hapus Belakang-" << endl;
        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 0:
            cout << "Program selesai." << endl;
            break;
        default:
            cout << "Pilihan tidak valid." << endl;
    }
} while (choice != 12);
return 0;
}

```

Screenshoot program

- 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

```

Pilih Operasi : 1
-Tambah Depan-
Masukkan Nama : Imelda
Masukkan NIM : 2311102004
Data telah ditambahkan

```

```
Pilih Operasi : 2
-Tambah Belakang-
Masukkan Nama : Fajar
Masukkan NIM : 2311102005
Data telah ditambahkan
```

- Tampilan Operasi Hapus

```
Pilih Operasi : 7
-Hapus Depan-
Data Imelda berhasil dihapus
```

```
Pilih Operasi : 8
-Hapus Belakang-
Data Fajar berhasil dihapus
```

- Tampilan Operasi Ubah

```
Pilih Operasi : 4
-Ubah Depan-
Masukkan Nama Baru : Awalina
Masukkan NIM Baru : 2311102006
Data Imelda telah diganti denga data Awalina
```

```
Pilih Operasi : 5
-Ubah Belakang-
Masukkan Nama Baru : Crisyanti
Masukkan NIM Baru : 2311102007
Data Fajar telah diganti denga data Crisyanti
```

- Tampilan Operasi Tampil Data

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Awalina 2311102006
Crisyanti 2311102007
```

Deskripsi program

Deskripsi program Program yang diberikan adalah implementasi dari struktur data linked list menggunakan bahasa C++. Program ini memiliki fitur untuk menambah, mengubah, dan menghapus data dalam linked list, serta menampilkan seluruh data yang tersimpan. Pada awalnya, program meminta pengguna memilih operasi yang ingin dilakukan, seperti menambah data di depan, belakang, atau di posisi tertentu, mengubah data, menghapus data di berbagai posisi, menghapus seluruh data dalam linked list, dan menampilkan seluruh data yang telah dimasukkan. Program menggunakan perulangan do-while untuk terus berjalan hingga pengguna memilih untuk keluar (pilihan nomor 0). Ini adalah program yang interaktif dan memungkinkan pengguna untuk mengelola data dengan mudah dalam struktur linked list non-circular

2. Unguided 2

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

Screenshot Program

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Imelda    2311102004
Farrel    23300003
Denis     23300005
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
```

3. Unguided 3

a) Tambahkan data berikut diantara Farrel dan Denis:

Wati 23300004

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Imelda    2311102004
Farrel    23300003
Wati      23300004
Denis     23300005
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
```

b) Hapus data Denis

```
Pilih Operasi : 9
-Hapus Tengah-
Masukkan Posisi : 5
Data Denis berhasil dihapus
```

c) Tambahkan data berikut di awal:

Owi 2330000

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Owi       2330000
Jawad     23300001
Imelda    2311102004
Farrel    23300003
Wati      2330004
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
```

d) Tambahkan data berikut di akhir :

David 23300100

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Owi       2330000
Jawad     23300001
Imelda    2311102004
Farrel    23300003
Wati      2330004
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
David     23300100
```

e) Ubah data Udin menjadi data berikut:

Idin 23300045

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Owi       23300000
Jawad     23300001
Imelda    2311102004
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
David     23300100
```

f) Ubah data terakhir menjadi berikut:

Lucy 23300101

```
Pilih Operasi : 5
-Ubah Belakang-
Masukkan Nama Baru : Lucy
Masukkan NIM Baru : 23300101
Data David telah diganti denga data Lucy
```

g) Hapus data awal

```
Pilih Operasi : 7
-Hapus Depan-
Data Owi berhasil dihapus
```

h) Ubah data awal menjadi berikut:

Bagas 2330002

```
Pilih Operasi : 4
-Ubah Depan-
Masukkan Nama Baru : Bagas
Masukkan NIM Baru : 2330002
Data Jawad telah diganti denga data Bagas
```

i) Hapus data akhir

```
Pilih Operasi : 8
-Hapus Belakang-
Data Lucy berhasil dihapus
```

j) Tampilkan seluruh data

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Bagas     2330002
Imelda    2311102004
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      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 mengulangi 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

BAB IV

KESIMPULAN

Linked List non circular adalah struktur data yang terdiri dari node dengan pointer yang menunjuk ke node berikutnya, kecuali pada node terakhir yang menunjuk ke nilai null sebagai penanda akhir dari linked list. Sedangkan Linkedlist circular adalah struktur data yang digunakan yang terdiri dari kumpulan simpul (node) yang terhubung dengan pointer yang saling terkait membentuk sirkuit.