

**LAPORAN PRAKTIKUM STRUKTUR DATA DAN
PEMROGRAMAN
MODUL IV
LINKED LIST CIRCULAR DAN NON CIRCULAR**



Disusun Oleh :

Ahmadan Syaridin

2311102038

IF-11-A

Dosen Pengampu :

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

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

BAB I

A. TUJUAN PRAKTIKUM

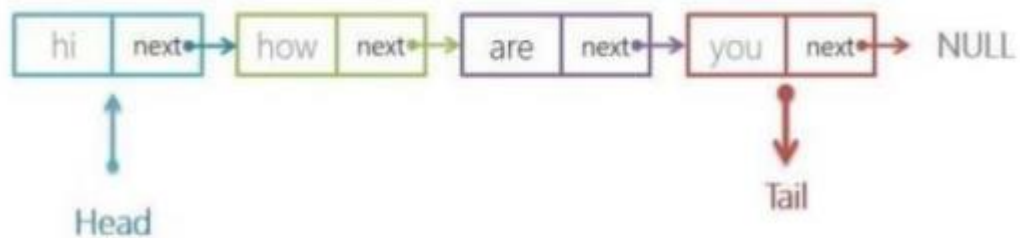
- a. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
- b. Praktikan dapat membuat linked list circular dan non circular.
- c. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

BAB II

B. DASAR TEORI

1. Linked List Non Sircular

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

3. Pengecekan Kondisi Linked List

```
bool isEmpty()  
{  
    if (head == NULL && tail == NULL)  
    {  
        return true;  
    }  
    Else  
    {  
        return false;  
    }  
}
```

4. Penambahan Simpul (Node)

```
void insertBelakang(string dataUser)  
{  
    If (isEmpty() == true)  
    {  
        node *baru = new node;  
        baru->data = dataUser;  
        head = baru;  
        tail = baru;  
        baru->next = NULL;  
    }  
    else  
    {  
        node *baru = new node;  
        baru->data = data User;  
        baru->next = NULL;  
        tail->next = baru;  
        tail = baru;  
    }  
};
```

5. Penghapusan Simpul (Node)

```
void hapusDepan()  
{  
    if (isEmpty()==true)  
    {  
        cout<<"Listkosong!"<<endl;  
    } else {  
        node *helper;  
        helper = head;  
        if (head == tail)
```

```

{
    head = NULL;
    tail = NULL;
    delete helper;
}
Else
head = head->next;
helper->next = NULL;
delete helper;}}}

```

6. Tampil Data Linked List

```

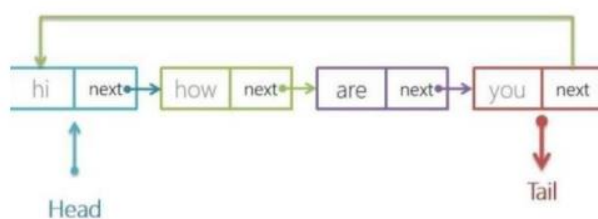
void tampil()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    Else
    {
        node *helper;
        helper = head;
        while (helper != NULL)
        {
            cout << helper->data << ends;
            helper = helper->next;
        }
    }
}

```

2. Linked List Circular

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

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



Gambar 2 Single Linked List Circular

1. Deklarasi Simpul (Node)

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

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
Node *head, *tail, *baru, *bantu, *hapus;
void init()
{
    head = NULL;
    tail = head;
}
```

3. Pengecekan dan Kondisi Linked List

```
int isEmpty()
{
    if (head == NULL)
        return 1; // true
    else return 0; // false
}
```

4. Pembuatan Simpul (Node)

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

5. Penambahan Simpul (Node)

```
// Tambah Depan
Void insertDepan (string data)
{
    // Buat Node baru
    buatNode(data);
    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        while (tail->next != head)
        {
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}
```

6. Penghapus 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

C. GUIDED

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;
    }

```

OUTPUT

```

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 mengimplementasikan operasi dasar pada struktur data linked list non-sirkular. Program ini menyediakan berbagai fungsi untuk menginisialisasi linked list, memeriksa apakah linked list kosong, menambahkan node di depan, di belakang, dan di tengah, menghitung jumlah node, menghapus node dari depan, belakang, dan tengah, mengubah nilai node di depan, belakang, dan tengah, menghapus seluruh linked list, dan menampilkan isi linked list. Fungsi-fungsi ini disusun untuk memungkinkan manipulasi dinamis dari node-node dalam linked list.

GUIDED 2

SOURCE CODE

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST CIRCULAR
// Deklarasi Struct Node
struct Node
{
    string data;
    Node *next;
};
Node *head, *tail, *baru, *bantu, *hapus;
void init()
{
    head = NULL;
    tail = head;
}
// Pengecekan
int isEmpty()
{
    if (head == NULL)
        return 1; // true
    else
        return 0; // false
}
// Buat Node Baru
void buatNode(string data)
{
    baru = new Node;
    baru->data = data;
    baru->next = NULL;
}
// Hitung List
int hitungList()
{
    bantu = head;
    int jumlah = 0;
    while (bantu != NULL)
    {
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}
// Tambah Depan
void insertDepan(string data)
{
    // Buat Node baru
    buatNode(data);
```

```

        if (isEmpty() == 1)
        {
            head = baru;
            tail = head;
            baru->next = head;
        }
        else
        {
            while (tail->next != head)
            {
                tail = tail->next;
            }
            baru->next = head;
            head = baru;
            tail->next = head;
        }
    }
    // Tambah Belakang
    void insertBelakang(string data)
    {
        // Buat Node baru
        buatNode(data);
        if (isEmpty() == 1)
        {
            head = baru;
            tail = head;
            baru->next = head;
        }
        else
        {
            while (tail->next != head)
            {
                tail = tail->next;
            }
            tail->next = baru;
            baru->next = head;
        }
    }
    // Tambah Tengah
    void insertTengah(string data, int posisi)
    {
        if (isEmpty() == 1)
        {
            head = baru;
            tail = head;
            baru->next = head;
        }
        else
        {
            baru->data = data;
            // transversing
            int nomor = 1;
            bantu = head;
            while (nomor < posisi - 1)
            {
                bantu = bantu->next;
                nomor++;
            }
            baru->next = bantu->next;
            bantu->next = baru;
        }
    }

```

```

}
// Hapus Depan
void hapusDepan()
{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;
            delete hapus;
        }
        else
        {
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            head = head->next;
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus Belakang
void hapusBelakang()
{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;
            delete hapus;
        }
        else
        {
            while (hapus->next != head)
            {
                hapus = hapus->next;
            }
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    }
}

```



```

        else
        {
            cout << "List masih kosong!" << endl;
        }
    }
    // Hapus Tengah
    void hapusTengah(int posisi)
    {
        if (isEmpty() == 0)
        {
            // transversing
            int nomor = 1;
            bantu = head;
            while (nomor < posisi - 1)
            {
                bantu = bantu->next;
                nomor++;
            }
            hapus = bantu->next;
            bantu->next = hapus->next;
            delete hapus;
        }
        else
        {
            cout << "List masih kosong!" << endl;
        }
    }
    // Hapus List
    void clearList()
    {
        if (head != NULL)
        {
            hapus = head->next;
            while (hapus != head)
            {
                bantu = hapus->next;
                delete hapus;
                hapus = bantu;
            }
            delete head;
            head = NULL;
        }
        cout << "List berhasil terhapus!" << endl;
    }
    // Tampilkan List
    void tampil()
    {
        if (isEmpty() == 0)
        {
            tail = head;
            do
            {
                cout << tail->data << ends;
                tail = tail->next;
            } while (tail != head);
            cout << endl;
        }
        else
        {
            cout << "List masih kosong!" << endl;
        }
    }

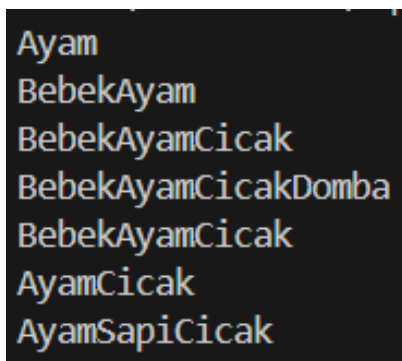
```

```

    }
    int main()
    {
        init();
        insertDepan("Ayam");
        tampil();
        insertDepan("Bebek");
        tampil();
        insertBelakang("Cicak");
        tampil();
        insertBelakang("Domba");
        tampil();
        hapusBelakang();
        tampil();
        hapusDepan();
        tampil();
        insertTengah("Sapi", 2);
        tampil();
        hapusTengah(2);
        tampil();
        return 0;
    }

```

SCREENSHOOT PROGRAM



```

Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak

```

DESKRIPSI PROGRAM

Program tersebut memiliki fungsi untuk menginisialisasi linked list, memeriksa apakah linked list kosong, membuat node baru, menambahkan node di depan, di belakang, dan di tengah, menghitung jumlah node, menghapus node dari depan, belakang, dan tengah, menghapus seluruh linked list, serta menampilkan isi linked list. Fungsi-fungsi ini memungkinkan manipulasi dinamis dari node-node yang saling terhubung dalam linked list, di mana node terakhir mengarah kembali ke node pertama, membentuk struktur sirkular. Program ini menunjukkan berbagai operasi tersebut melalui contoh-contoh seperti menambah dan menghapus node pada berbagai posisi dalam linked list.

BAB IV

D. UNGUIDED

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;
        }
        Node* temp = head;
        head = head->next;
        delete temp;
        cout << "Data berhasil dihapus" << endl;
    }

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

    void hapusTengah(int posisi) {
        if (posisi <= 0 || head == nullptr) {
            cout << "Linked list kosong atau posisi tidak
valid" << endl;
            return;
        }
        if (posisi == 1) {
            hapusDepan();
            return;
        }
        Node* temp = head;
        for (int i = 0; i < posisi - 2; i++) {

```

```

        if (temp->next == nullptr) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        temp = temp->next;
    }
    if (temp->next == nullptr) {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    Node* nodeToDelete = temp->next;
    temp->next = temp->next->next;
    delete nodeToDelete;
    cout << "Data berhasil dihapus" << endl;
}

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

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

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

```

```

        cout << "Data berhasil diubah" << endl;
    }

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

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

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

    do {
        cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" <<
endl;
        cout << "1. Tambah Depan" << endl;
        cout << "2. Tambah Belakang" << endl;
        cout << "3. Tambah Tengah" << endl;
        cout << "4. Ubah Depan" << endl;
        cout << "5. Ubah Belakang" << endl;
        cout << "6. Ubah Tengah" << endl;
        cout << "7. Hapus Depan" << endl;
        cout << "8. Hapus Belakang" << endl;
        cout << "9. Hapus Tengah" << endl;
        cout << "10. Hapus List" << endl; // Menu untuk hapus
list
        cout << "11. Tampilkan Data" << endl;
        cout << "12. Keluar" << endl;
        cout << "Pilih Operasi : ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "-Tambah Depan-" << endl;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                linkedList.tambahDepan(nama, nim);
                break;
            case 2:

```

```

        cout << "-Tambah Belakang-" << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.tambahBelakang(nama, nim);
        break;
    case 3:
        cout << "-Tambah Tengah-" << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.tambahTengah(nama, nim, posisi);
        break;
    case 4:
        cout << "-Ubah Depan-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.ubahDepan(nama, nim);
        break;
    case 5:
        cout << "-Ubah Belakang-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.ubahBelakang(nama, nim);
        break;
    case 6:
        cout << "-Ubah Tengah-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.ubahTengah(nama, nim, posisi);
        break;
    case 7:
        linkedList.hapusDepan();
        break;
    case 8:
        linkedList.hapusBelakang();
        break;
    case 9:
        cout << "-Hapus Tengah-" << endl;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.hapusTengah(posisi);
        break;
    case 10:
        linkedList.hapusList(); // Hapus List
        break;
    case 11:
        linkedList.tampilkanData();
        break;

```

```

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

return 0;
}

```

OUTPUT

Tampilan menu

```

PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan Data
12. Keluar
Pilih Operasi : 

```

Tampilan Operasi Tambah

```

Pilih Operasi : 1
-Tambah Depan-
Masukkan Nama : AmadanSyaridin
Masukkan NIM : 2311102038
Data telah ditambahkan
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

```

```

Pilih Operasi : 3
-Tambah Tengah-
Masukkan Nama : Jawad
Masukkan NIM : 23300001
Masukkan Posisi : 2
Posisi tidak valid
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

```


Tampilan Operasi Hapus

```
Pilih Operasi : 8
Data berhasil dihapus
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi : 7
Linked list kosong
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

Tampilan Operasi Ubah

```
Pilih Operasi : 5
-Ubah Belakang-
Masukkan Nama Baru : Farel
Masukkan NIM Baru : 23300003
Linked list kosong
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

Tampilan Operasi Tampil data

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

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

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
AhmadanSyaridin 2311102038
Farrel    23300003
Dennis    23300005
Anis      23300008
Bowo      23300015
Gahar     233000040
Udin      233000048
Ucok      233000050
Budi      233000099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

- a. Tambahkan data berikut diantara Farrel dan Denis
Wati 2330004

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
AhmadanSyaridin 2311102038
Farrel    23300003
Wati      23300004
Dennis    23300005
Anis      23300008
Bowo      23300015
Gahar     233000040
Udin      233000048
Ucok      233000050
Budi      233000099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

- b. Hapus Data Denis

```
Pilih Operasi : 9
-Hapus Tengah-
Masukkan Posisi : 5
Data berhasil dihapus
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
AhmadanSyaridin 2311102038
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     233000040
Udin      233000048
Ucok      233000050
Budi      233000099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

- c. Tambahkan data Berikut di Awal
Owi 2330000

```
Pilih Operasi : 1
-Tambah Depan-
Masukkan Nama : Owi
Masukkan NIM : 2330000
Data telah ditambahkan
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Owi       2330000
Jawad     23300001
AhmadanSyaridin 2311102038
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     233000040
Udin      233000048
Ucok      233000050
Budi      233000099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

- d. Tambahkan Data Berikut di Akhir
David 23300100

```
Pilih Operasi : 2
-Tambah Belakang-
Masukkan Nama : David
Masukkan NIM : 23300100
Data telah ditambahkan
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Owi        2330000
Jawad      23300001
AhmadanSyaridin 2311102038
Farrel     23300003
Wati       2330004
Anis       23300008
Bowo       23300015
Gahar      233000040
Udin       233000048
Ucok       233000050
Budi       233000099
David      23300100
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

- e. Ubah Data Udin Menjadi Data Berikut
Udin 23300045

```
Pilih Operasi : 6
-Ubah Tengah-
Masukkan Nama Baru : Udin
Masukkan NIM Baru : 23300045
Masukkan Posisi : 9
Data berhasil diubah
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Owi        2330000
Jawad      23300001
AhmadanSyaridin 2311102038
Farrel     23300003
Wati       2330004
Anis       23300008
Bowo       23300015
Gahar      233000040
Udin       23300045
Ucok       233000050
Budi       233000099
David      23300100
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

- f. Ubah Data Terakhir Menjadi Berikut

```
Pilih Operasi : 5
-Ubah Belakang-
Masukkan Nama Baru : Lucy
Masukkan NIM Baru : 23300101
Data berhasil diubah
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Owi        2330000
Jawad      23300001
AhmadanSyaridin 2311102038
Farrel     23300003
Wati       23300004
Anis       23300008
Bowo       23300015
Gahar      233000040
Udin       23300045
Ucok       233000050
Budi       233000099
Lucy       23300101
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

- g. Hapus Data Awal

```
Pilih Operasi : 7
Data berhasil dihapus
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
DATA MAHASISWA
NAMA      NIM
Jawad      23300001
AhmadanSyaridin 2311102038
Farrel     23300003
Wati       23300004
Anis       23300008
Bowo       23300015
Gahar      233000040
Udin       23300045
Ucok       233000050
Budi       233000099
Lucy       23300101
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

- h. Ubah Data Awal Menjadi Berikut
Bagas 2330002

```
Pilih Operasi : 4
-Ubah Depan-
Masukkan Nama Baru : Bagas
Masukkan NIM Baru : 2330002
Data berhasil diubah
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Bagas     2330002
AhmadanSyaridin 2311102038
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     233000040
Udin      23300045
Ucok      233000050
Budi      233000099
Lucy      23300101
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

- i. Hapus Data Akhir

```
Pilih Operasi : 8
Data berhasil dihapus
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Bagas     2330002
AhmadanSyaridin 2311102038
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     233000040
Udin      23300045
Ucok      233000050
Budi      233000099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

j. Tampilkan Seluruh Data

```
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Bagas     2330002
AhmadanSyaridin 2311102038
Farrel    23300003
Wati      2330004
Anis      23300008
Bowo      23300015
Gahar     233000040
Udin      23300045
Ucok      233000050
Budi      233000099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

DESKRIPSI PROGRAM

Program ini mengimplementasikan struktur data Single Linked List non-circular yang memungkinkan berbagai operasi untuk memanipulasi data mahasiswa, seperti menambah, menghapus, dan mengubah node. Pengguna dapat menambah data di awal, akhir, atau tengah list, mengubah data di posisi tertentu, serta menghapus node dari awal, akhir, atau tengah. Selain itu, program ini menyediakan opsi untuk menghapus seluruh linked list dan menampilkan data yang ada di dalamnya. Program berjalan dalam loop yang terus meminta input dari pengguna hingga mereka memilih untuk keluar.

BAB V

E. KESIMPULAN

Single dan Double Linked List adalah dua tipe struktur data yang memungkinkan penyimpanan dan manipulasi elemen data secara dinamis. Single Linked List non-sirkular memiliki node yang terhubung secara linear, di mana setiap node mengarah ke node berikutnya sampai node terakhir mengarah ke null. Double Linked List non-sirkular memperluas konsep ini dengan setiap node juga menunjuk kembali ke node sebelumnya, memudahkan traversal dua arah. Dalam Linked List sirkular, baik single maupun double, node terakhir mengarah kembali ke node pertama, menciptakan struktur melingkar tanpa ujung null. Perbedaan utama antara sirkular dan non-sirkular adalah bahwa dalam sirkular, traversal dapat terus-menerus berputar melalui list tanpa mencapai akhir, sedangkan dalam non-sirkular traversal berakhir ketika mencapai node terakhir. Struktur ini masing-masing memiliki keuntungan dalam situasi tertentu, seperti manajemen memori dan kemudahan akses elemen data.

BAB VI

F. DAFTAR PUSTAKA

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications.