LAPORAN PRAKTIKUM

MODUL 4

LINKED LIST CIRCULAR DAN NON CIRCULAR

Disusun oleh:
Nofita Fitriyani
NIM: 2311102001

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

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

BAB I

TUJUAN PRAKTIKUM

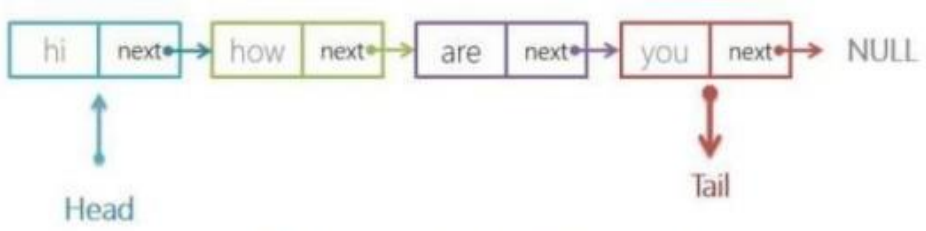
a. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
b. Praktikan dapat membuat linked list circular dan non circular.
c. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

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.

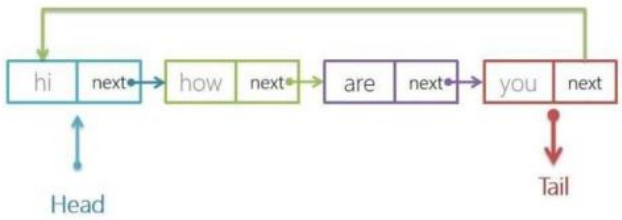


Gambar 1 *Single Linked List Non Circular*

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;
}

// Pengecekkan 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++;
```

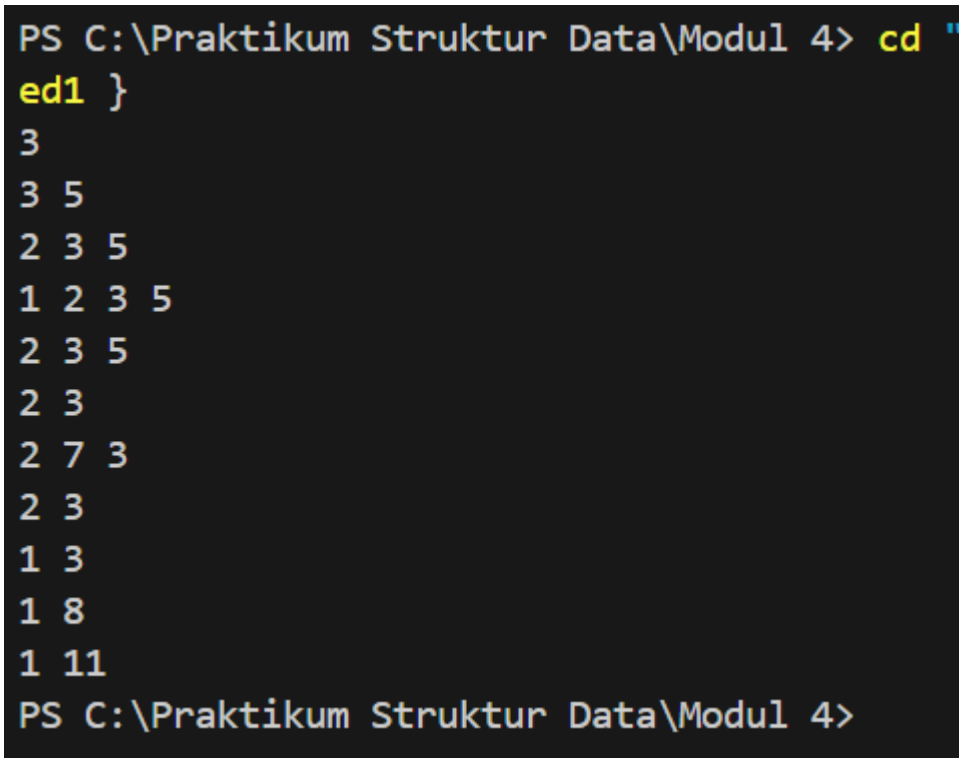
```
            }
            sebelum->next = bantu;
            delete hapus;
        }
    }

    // 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



Deskripsi program

Program ini adalah implementasi dari single linked list non-circular. Program ini memiliki beberapa fungsi utama, seperti menambahkan node baru di depan, belakang, atau di tengah linked list, menghapus node dari depan, belakang, atau di tengah linked list, mengubah nilai data pada node tertentu, membersihkan seluruh linked list, dan menampilkan isi linked list.

Dalam program ini, setiap node memiliki dua bagian utama: data yang merupakan nilai integer, dan pointer next yang menunjukkan ke node berikutnya dalam linked list. Pada awalnya, head dan tail diinisialisasi sebagai NULL untuk menandakan bahwa linked list kosong. Fungsi isEmpty() digunakan untuk memeriksa apakah linked list tersebut kosong atau tidak.

Fungsi insertDepan() dan insertBelakang() digunakan untuk menambahkan node baru di depan atau belakang linked list. Fungsi insertTengah() digunakan untuk menambahkan node baru di posisi tertentu di antara dua node. Fungsi

hapusDepan(), hapusBelakang(), dan hapusTengah() digunakan untuk menghapus node dari depan, belakang, atau di posisi tertentu dalam linked list.

Fungsi ubahDepan(), ubahBelakang(), dan ubahTengah() digunakan untuk mengubah nilai data pada node tertentu. Fungsi clearList() digunakan untuk menghapus semua node dalam linked list dan mengembalikan linked list ke kondisi awal (kosong). Fungsi tampilList() digunakan untuk menampilkan isi linked list pada saat itu. Program ini mengimplementasikan operasi-operasi dasar pada single linked list non-circular dengan menggunakan konsep pointer untuk menghubungkan antara satu node dengan node lainnya.

2. Guided 2

Source code

```
#include <iostream>

using namespace std;

// Deklarasi Struct Node



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

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

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

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

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

//Hitung List
int hitungList(){
    bantu = head;
    int jumlah = 0;
    while (bantu != NULL) {
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}

//Tambah Depan
void insertDepan(string data){
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1){
```

```
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        while (tail->next != head){
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}

//Tambah Belakang
void insertBelakang(string data){
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1){
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        while (tail->next != head){
            tail = tail->next;
        }
        tail->next = baru;
        baru->next = head;
    }
}

//Tambah Tengah
void insertTengah(string data, int posisi){
    if (isEmpty() == 1){
```

```
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        baru->data = data;
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1){
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

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

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

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

// Hapus Tengah
void hapusTengah(int posisi){
    if (isEmpty() == 0){
```

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

//Hapus List
void clearList(){
    if (head != NULL){
        hapus = head->next;
        while (hapus != head){
            bantu = hapus->next;
            delete hapus;
            hapus = bantu;
        }
        delete head;
        head = NULL;
    }
    cout << "List berhasil terhapus!" << endl;
}

//Tampilkan List
void tampil(){
    if (isEmpty() == 0){
        tail = head;
```

```
        do {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

    return 0;
}
```

Screenshoot program

```
PS C:\Praktikum Struktur Data\Modul 4> cd
ed2 }
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
PS C:\Praktikum Struktur Data\Modul 4>
```

Deskripsi program

Program ini adalah implementasi dari circular linked list menggunakan bahasa C++. Circular linked list adalah jenis linked list di mana setiap node memiliki pointer yang menunjuk kembali ke node pertama, membentuk lingkaran. Program ini memiliki beberapa fungsi utama, seperti menambahkan node baru di depan, belakang, atau di tengah linked list, menghapus node dari depan, belakang, atau di tengah linked list, membersihkan seluruh linked list, dan menampilkan isi linked list.

Pada awalnya, program menginisialisasi node head dan tail sebagai NULL untuk menandakan bahwa linked list masih kosong. Fungsi isEmpty() digunakan untuk memeriksa apakah linked list tersebut kosong atau tidak. Fungsi buatNode(string data) digunakan untuk membuat node baru dengan data yang diberikan.

Fungsi insertDepan(string data) dan insertBelakang(string data) digunakan untuk menambahkan node baru di depan atau belakang linked list. Jika linked list masih kosong, head dan tail akan menunjuk ke node baru. Jika tidak, tail akan

dipindahkan ke node terakhir, dan node baru akan ditambahkan setelah tail, kemudian tail menunjuk kembali ke head, membentuk lingkaran.

Fungsi insertTengah(string data, int posisi) digunakan untuk menambahkan node baru di posisi tertentu di antara dua node. Fungsi hapusDepan(), hapusBelakang(), dan hapusTengah(int posisi) digunakan untuk menghapus node dari depan, belakang, atau di posisi tertentu dalam linked list. Fungsi clearList() digunakan untuk menghapus semua node dalam linked list dan mengembalikan linked list ke kondisi awal (kosong).

Fungsi tampil() digunakan untuk menampilkan isi linked list pada saat itu. Program ini mengimplementasikan operasi-operasi dasar pada circular linked list dengan menggunakan konsep pointer untuk menghubungkan antara satu node dengan node lainnya, serta menjaga lingkaran pada linked list dengan mengatur pointer tail.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM mahasiswa

Source code

```
#include <iostream>
using namespace std;
//2311102001_Nofita

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 : Nofita
Masukkan NIM : 2311102001
Data telah ditambahkan
```

```
Pilih Operasi : 2
-Tambah Belakang-
Masukkan Nama : Fitriyani
Masukkan NIM : 2311102003
Data telah ditambahkan
```

```
Pilih Operasi : 3
-Tambah Tengah-
Masukkan Nama : Dan
Masukkan NIM : 2311102002
Masukkan Posisi : 1
Data telah ditambahkan
```

- Tampilan Operasi Hapus

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

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

```
Pilih Operasi : 9
-Hapus Tengah-
Masukkan Posisi : 2
Data Dan berhasil dihapus
```

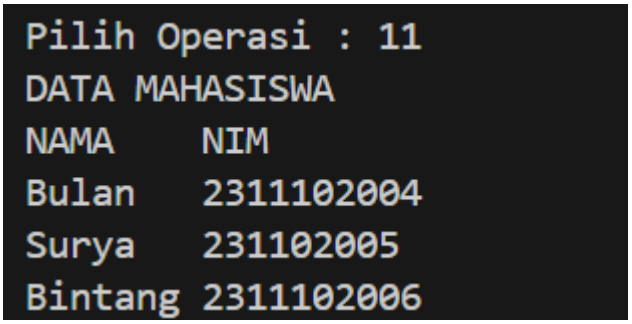
- Tampilan Operasi Ubah

```
Pilih Operasi : 4
-Ubah Depan-
Masukkan Nama Baru : Bulan
Masukkan NIM Baru : 2311102004
Data Nofita telah diganti denga data Bulan
```

```
Pilih Operasi : 5
-Ubah Belakang-
Masukkan Nama Baru : Bintang
Masukkan NIM Baru : 2311102006
Data Fitriyani telah diganti denga data Bintang
```

```
Pilih Operasi : 6
-Ubah Tengah-
Masukkan Nama Baru : Surya
Masukkan NIM Baru : 231102005
Masukkan Posisi : 2
Data Dan telah diganti denga data Surya
```

- Tampilan Operasi Tampil Data



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    2300001
Nofita   2311102001
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 2330004

```
DATA MAHASISWA
NAMA     NIM
Jawad    2300001
Nofita   2311102001
Farrel   23300003
Wati     2330004
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

```
DATA MAHASISWA
NAMA    NIM
Owi     2330000
Jawad   2300001
Nofita  2311102001
Farrel  23300003
Wati    2330004
Anis    23300008
Bowo    23300015
Gahar   23300040
Udin    23300048
Ucok    23300050
Budi    23300099
```

d) Tambahkan data berikut di akhir :
David 23300100

```
DATA MAHASISWA
NAMA    NIM
Owi     2330000
Jawad   2300001
Nofita  2311102001
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 : 6
-Ubah Tengah-
Masukkan Nama Baru : Idin
Masukkan NIM Baru : 23300045
Masukkan Posisi : 9
Data Udin telah diganti denga data Idin
```

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
Nofita  2311102001
Farrel  23300003
Wati    2330004
Anis    23300008
Bowo    23300015
Gahar   23300040
Idin    23300045
Ucok    23300050
Budi    23300099
```

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