

**LAPORAN PRAKTIKUM  
STRUKTUR DATA PEMOGRAMAN**

**MODUL IV  
“LINKED LIST CIRCULAR DAN NON CIRCULAR”**



**Disusun Oleh:**

NAMA : Trie Nabilla Farhah

NIM : 2311102071

**Dosen Pengampu:**

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

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

**2024**

## A. DASAR TEORI

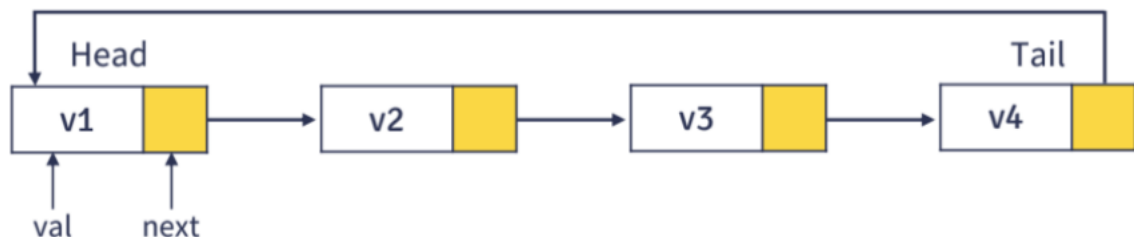
Linked List adalah suatu cara untuk menyimpan data dengan struktur sehingga programmer dapat secara otomatis menciptakan suatu tempat baru untuk menyimpan data kapan saja diperlukan. Linked list dikenal juga dengan sebutan senarai berantai adalah stuktur data yang terdiri dari urutan record data dimana setiap record memiliki field yang menyimpan alamat/referensi dari record selanjutnya (dalam urutan). Elemen data yang dihubungkan dengan link pada linked list disebut Node. Biasanya dalam suatu linked list, terdapat istilah head dan tail .

- Head adalah elemen yang berada pada posisi pertama dalam suatu linked list
- Tail adalah elemen yang berada pada posisi terakhir dalam suatu linked list.

### 1. Linked List Circular

Circular Linked List adalah suatu linked list yang tidak memiliki nilai nil/NULL untuk medan sambungannya. Circular Linked List dapat dilakukan terhadap Singly Linked List maupun Doubly Linked List. Dalam Circular Linked List tidak terdapat suatu simpul yang bernilai NULL. Hal ini terjadi karena simpul terakhir dihubungkan terhadap simpul pertama untuk Single Linked List dan simpul pertama dengan simpul terakhir saling terhubung untuk Double Linked List. Semua simpul berhak diperlakukan sebagai simpul depan.

Circular linked list dapat disebut juga dengan linked list unidirectional. Kita hanya dapat melintasinya dalam satu arah. Tetapi jenis linked list ini memiliki simpul terakhir yang menunjuk ke simpul kepala. Jadi saat melintas, kita harus berhati-hati dan berhenti saat mengunjungi kembali simpul kepala..



### Operasi Pada 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 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. Penghapusan 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;
    }
}

```

```
}
```

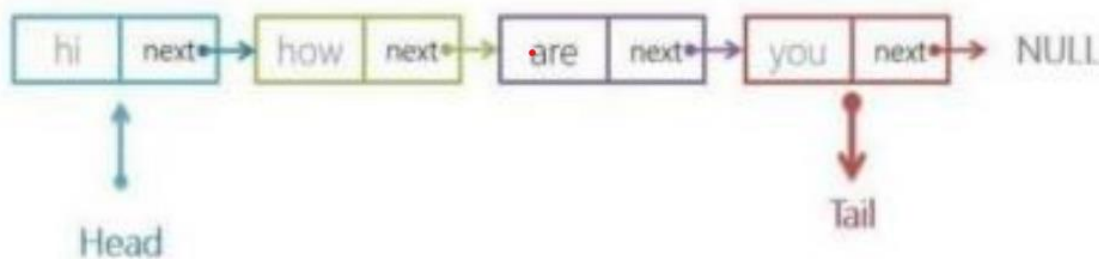
## 7. Menampilkan Data Linked List

```
void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
}
```

## 2. Linked list Non Circular

Linked list non-circular adalah jenis struktur data linked list di mana tidak ada node yang menunjuk kembali ke node sebelumnya untuk membentuk suatu lingkaran. Dengan kata lain, linked list non-circular memiliki titik akhir yang menunjuk ke null atau ke node yang tidak ada di dalam list (biasanya disebut sebagai tail node atau end node).

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.



## 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 = dataUser;  
        baru->next = NULL;  
        tail->next = baru;  
    }  
}
```

```
        tail = baru;
    }
};
```

## 5. Penghapusan Simpul (Node)

```
void hapusDepan()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" <<
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;  
        }  
    }  
}
```



## B. GUIDED

### GUIDED 1 : Linked List Non Circular

#### Source Code

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

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

Node *head;
Node *tail;

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

bool isEmpty() {
    return head == NULL;
}

void insertDepan(int nilai) {
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

void insertBelakang(int nilai) {
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
```

```

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

int hitungList() {
    Node *hitung = head;
    int jumlah = 0;
    while (hitung != NULL) {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

void insertTengah(int data, int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {
        Node *baru = new Node();
        baru->data = data;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan() {
    if (!isEmpty()) {
        Node *hapus = head;
        if (head->next != NULL) {

```

```

        head = head->next;
    } else {
        head = tail = NULL;
    }
    delete hapus;
} else {
    cout << "List kosong!" << endl;
}
}

void hapusBelakang() {
    if (!isEmpty()) {
        Node *hapus = tail;
        if (head != tail) {
            Node *bantu = head;
            while (bantu->next != tail) {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
        } else {
            head = tail = NULL;
        }
        delete hapus;
    } else {
        cout << "List kosong!" << endl;
    }
}

void hapusTengah(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 *bantu = head;
        Node *hapus;
        Node *sebelum = NULL;
        int nomor = 1;
        while (nomor < posisi) {
            sebelum = bantu;

```

```

        bantu = bantu->next;
        nomor++;
    }
    hapus = bantu;
    if (sebelum != NULL) {
        sebelum->next = bantu->next;
    } else {
        head = bantu->next;
    }
    delete hapus;
}
}

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

void ubahTengah(int data, int posisi) {
    if (!isEmpty()) {
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi di luar jangkauan" << endl;
        } else if (posisi == 1) {
            cout << "Posisi bukan posisi tengah" << endl;
        } else {
            Node *bantu = head;
            int nomor = 1;
            while (nomor < posisi) {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}
}

```

```

void ubahBelakang(int data) {
    if (!isEmpty()) {
        tail->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

void tampil() {
    Node *bantu = head;
    if (!isEmpty()) {
        while (bantu != NULL) {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

```

```

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

        return 0;
}

```

## Output

```

PS D:\Semester2\Struktur Data\Praktikum4> & 'c:\Users\Asus\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugL
auncher.exe' '--stdin=Microsoft-MIEngine-In-idr1yono.v8o' '--stdout=Microsoft-MIEngine-Out-2uzr54zx.wsy' '--stderr=Microsoft-MIEngine-Error-dgqf0g
30.5xa' '--pid=Microsoft-MIEngine-Pid-zu20jkwg.ava' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS D:\Semester2\Struktur Data\Praktikum4>

```

Na

File Edit View

Nama : Trie Nabilla Farhah

Nim : 2311102071

Ln 2, Col 17 | 44 characters | 100% | Window | UTF-8

## Deskripsi Program

Program di atas adalah contoh implementasi linked list non-circular dalam C++. Program ini memiliki beberapa fungsi, seperti `init()`, `isEmpty()`, `insertDepan()`, `insertBelakang()`,

hitungList(), insertTengah(), hapusDepan(), hapusBelakang(), hapusTengah(), ubahDepan(), ubahTengah(), ubahBelakang(), clearList(), dan tampil().

- Fungsi `init()` digunakan untuk menginisialisasi linked list dengan head dan tail dengan nilai NULL. Selanjutnya, program akan menambahkan beberapa node ke linked list menggunakan `insertDepan()` dan `insertBelakang()`. `insertDepan()` digunakan untuk menambahkan node baru di depan linked list, sedangkan `insertBelakang()` digunakan untuk menambahkan node baru di belakang linked list.
- Fungsi `hitungList()` digunakan untuk menghitung jumlah node dalam linked list. Selanjutnya, program akan memasukkan node baru di tengah linked list menggunakan `insertTengah()`. `insertTengah()` memerlukan tiga parameter, yaitu data, posisi, dan node sebagai acuan.
- Fungsi `hapusDepan()`, `hapusBelakang()`, dan `hapusTengah()` digunakan untuk menghapus node dari linked list. `hapusDepan()` digunakan untuk menghapus node pertama, `hapusBelakang()` digunakan untuk menghapus node terakhir, dan `hapusTengah()` digunakan untuk menghapus node tertentu berdasarkan posisi.
- Fungsi `ubahDepan()`, `ubahTengah()`, dan `ubahBelakang()` digunakan untuk mengubah data pada node tertentu. `ubahDepan()` digunakan untuk mengubah data pada node pertama, `ubahTengah()` digunakan untuk mengubah data pada node tertentu berdasarkan posisi, dan `ubahBelakang()` digunakan untuk mengubah data pada node terakhir.
- Fungsi `clearList()` digunakan untuk menghapus semua node dalam linked list. Selama head tidak NULL, program akan terus memasukkan head ke variabel hapus dan mempergeser head ke node berikutnya. Setelah itu, program akan menghapus hapus dari memori. Selama program berjalan, fungsi `tampil()` akan dipanggil setelah setiap perintah untuk menampilkan data pada setiap node dalam linked list.
- Fungsi `tampil()` akan mengiterasi linked list dari head dan akan mencetak data pada setiap node hingga NULL.

## GUIDED 2 : Linked List Circular

### Source Code

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

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

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

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

int isEmpty() {
    return head == NULL;
}

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

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

void insertDepan(string data) {
    buatNode(data);
    if (isEmpty()) {
```



```

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

void insertBelakang(string data) {
    buatNode(data);
    if (isEmpty()) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        while (tail->next != head) {
            tail = tail->next;
        }
        tail->next = baru;
        baru->next = head;
    }
}

void insertTengah(string data, int posisi) {
    if (isEmpty()) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        baru->data = data;
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
    }
}

```

```

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

void hapusDepan() {
    if (!isEmpty()) {
        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;
    }
}

void hapusBelakang() {
    if (!isEmpty()) {
        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;
    }
}

void hapusTengah(int posisi) {
    if (!isEmpty()) {
        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;
    }
}

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;
}

void tampil() {

```

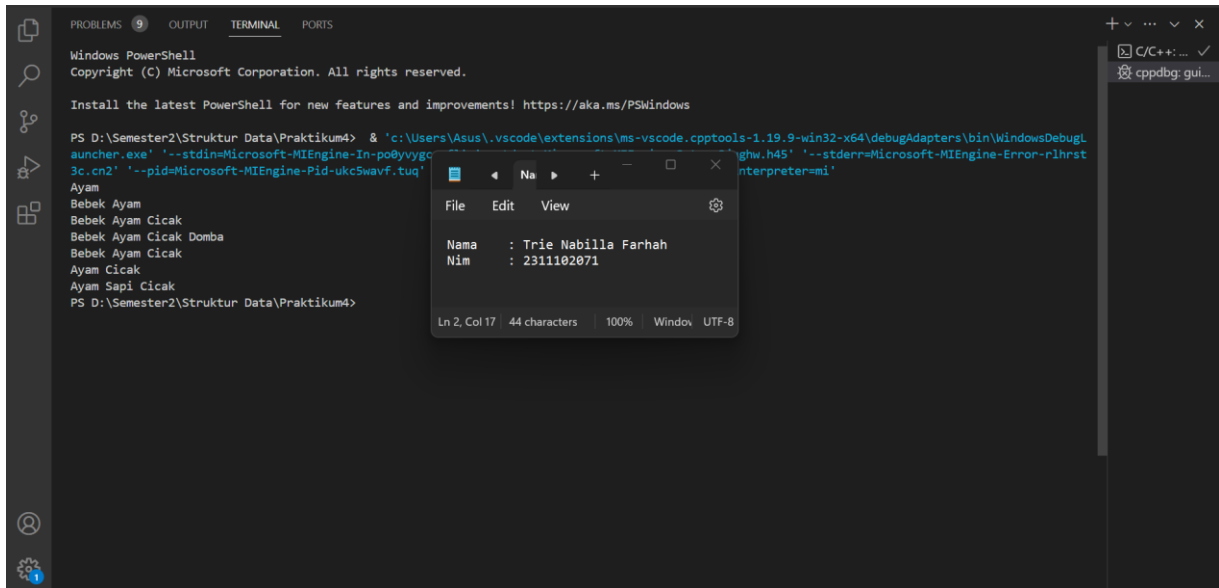
```

        if (!isEmpty()) {
            tail = head;
            do {
                cout << tail->data << " ";
                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;
}

```

## Output



```
Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Semester2\Struktur Data\Praktikum4> & 'c:\Users\Asus\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugL
auncher.exe' '--stdin=Microsoft-MIEngine-In-po0yvvgc' 'ghw.h45' '--stderr=Microsoft-MIEngine-Error-rlhrst
3c.cn2' '--pid=Microsoft-MIEngine-Pid-ukc5wavf.tuq' 'nterpreter=mi'
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak
PS D:\Semester2\Struktur Data\Praktikum4>
```

Dialog Box Output:

```
Nama : Trie Nabilla Farhah
Nim  : 2311102071
```

## Deskripsi Program

Program tersebut adalah implementasi dari linked list non-circular menggunakan bahasa C++. Program ini memiliki beberapa fungsi untuk mengelola linked list, seperti `init()`, `insertDepan()`, `insertBelakang()`, `insertTengah()`, `hapusDepan()`, `hapusBelakang()`, `hapusTengah()`, `clearList()`, `tampil()`, dan `main()`.

- Fungsi `init()` digunakan untuk menginisialisasi linked list dengan membuat head dan tail menunjuk ke NULL.
- Fungsi `insertDepan()` digunakan untuk menambahkan node baru di depan linked list.
- Fungsi `insertBelakang()` digunakan untuk menambahkan node baru di belakang linked list.
- Fungsi `insertTengah()` digunakan untuk menambahkan node baru di tengah linked list.
- Fungsi `hapusDepan()` digunakan untuk menghapus node pertama dari linked list.
- Fungsi `hapusBelakang()` digunakan untuk menghapus node terakhir dari linked list.
- Fungsi `hapusTengah()` digunakan untuk menghapus node tertentu dari linked list.
- Fungsi `clearList()` digunakan untuk menghapus semua node dari linked list. Fungsi `tampil()` digunakan untuk menampilkan semua node dari linked list.
- Fungsi `main()` digunakan sebagai program utama yang mengelola semua operasi tersebut.

Program ini menggunakan struktur data Node yang memiliki dua atribut, yaitu data dan next. Atribut data digunakan untuk menyimpan data dari node, sedangkan atribut next digunakan untuk menyimpan alamat memori dari node berikutnya.

### C. UNGUIDED

Buatlah program menu Single Linked List Non-Circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user.

1. Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM mahasiswa.
2. Setelah membuat menu tersebut, masukkan data sesuai urutan berikut, lalu tampilkan data yang telah dimasukkan. (Gunakan insert depan, belakang atau tengah).
3. Lakukan perintah berikut:
  - a) Tambahkan data berikut diantara Farrel dan Denis: Wati 2330004
  - b) Hapus data Denis
  - c) Tambahkan data berikut di awal: Owi 2330000
  - d) Tambahkan data berikut di akhir: David 23300100
  - e) Ubah data Udin menjadi data berikut: Idin 23300045
  - f) Ubah data terakhir menjadi berikut: Lucy 23300101
  - g) Hapus data awal
  - h) Ubah data awal menjadi berikut: Bagas 2330002
  - i) Hapus data akhir
  - j) Tampilkan seluruh data

#### Unguided 1

Source Code

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

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

class LinkedList {
private:
    identitasMahasiswa* head;

public:
    LinkedList() {
        head = NULL;
    }
};
```

```

    }

    void addFirst(string nama, string nim) {
        identitasMahasiswa* newNode = new
    identitasMahasiswa();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = head;
        head = newNode;
        cout << endl << "Data telah ditambahkan"<<endl;
    }

    void addLast(string nama, string nim) {
        identitasMahasiswa* newNode = new
    identitasMahasiswa();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = NULL;
        if (head == NULL) {
            head = newNode;
        } else {
            identitasMahasiswa* temp = head;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
        cout << endl << "Data telah ditambahkan"<<endl;
    }

    void addMiddle(string nama, string nim, int posisi) {
        if (posisi < 1) {
            cout << endl << "Posisi tidak valid"<<endl;
            return;
        }
        identitasMahasiswa* newNode = new
    identitasMahasiswa();
        newNode->nama = nama;
        newNode->nim = nim;
        if (posisi == 1) {
            newNode->next = head;

```

```

        head = newNode;
    } else {
        identitasMahasiswa* temp = head;
        for (int i = 1; i < posisi - 1; i++) {
            if (temp == NULL) {
                cout << endl << "Posisi tidak
valid"<<endl;
                return;
            }
            temp = temp->next;
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
    cout << endl << "Data telah ditambahkan"<<endl;
}

void changeFirst(string nama, string nim) {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    head->nama = nama;
    head->nim = nim;
    cout << endl << "Data di depan telah
diubah"<<endl;
}

void changeLast(string nama, string nim) {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    identitasMahasiswa* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->nama = nama;
    temp->nim = nim;
    cout << endl << "Data di belakang telah
diubah"<<endl;
}

```



```

    }

    void changeMiddle(string nama, string nim, int posisi)
    {
        if (posisi < 1) {
            cout << endl << "Posisi tidak valid"<<endl;
            return;
        }
        if (head == NULL) {
            cout << endl << "Linked List kosong"<<endl;
            return;
        }
        identitasMahasiswa* temp = head;
        for (int i = 1; i < posisi; i++) {
            if (temp == NULL) {
                cout << endl << "Posisi tidak
valid"<<endl;
                return;
            }
            temp = temp->next;
        }
        temp->nama = nama;
        temp->nim = nim;
        cout << endl << "Data di posisi " << posisi << "
telah diubah"<<endl;
    }

    void deleteFirst() {
        if (head == NULL) {
            cout << endl << "Linked List kosong"<<endl;
            return;
        }
        identitasMahasiswa* temp = head;
        head = head->next;
        delete temp;
        cout << endl << "Data di depan telah
dihapus"<<endl;
    }

    void deleteLast() {
        if (head == NULL) {

```

```

        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    if (head->next == NULL) {
        delete head;
        head = NULL;
        cout << endl << "Data di belakang telah
dihapus"<<endl;
        return;
    }
    identitasMahasiswa* temp = head;
    while (temp->next->next != NULL) {
        temp = temp->next;
    }
    delete temp->next;
    temp->next = NULL;
    cout << endl << "Data di belakang telah
dihapus"<<endl;
}

void deleteMiddle(int posisi) {
    if (posisi < 1) {
        cout << endl << "Posisi tidak valid"<<endl;
        return;
    }
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    identitasMahasiswa* temp = head;
    if (posisi == 1) {
        head = head->next;
        delete temp;
        cout << endl << "Data di posisi " << posisi <<
" telah dihapus"<<endl;
        return;
    }
    for (int i = 1; i < posisi - 1; i++) {
        if (temp == NULL || temp->next == NULL) {
            cout << endl << "Posisi tidak
valid"<<endl;

```

```

        return;
    }
    temp = temp->next;
}
identitasMahasiswa* nextNode = temp->next->next;
delete temp->next;
temp->next = nextNode;
cout << endl << "Data di posisi " << posisi << "
telah dihapus"<<endl;
}

void print() {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    cout << "===== DATA MAHASISWA =====<<
endl;
    cout << "=====<<
endl;
    cout << setw(20) << left << "NAMA" << setw(20) <<
left << "NIM" << endl;
    identitasMahasiswa* temp = head;
    while (temp != NULL) {
        cout << setw(20) << left << temp->nama <<
setw(20) << left << temp->nim << endl;
        temp = temp->next;
    }
}

void deleteAll() {
    while (head != NULL) {
        identitasMahasiswa* temp = head;
        head = head->next;
        delete temp;
    }
    cout << endl << "Semua data mahasiswa telah
dihapus"<<endl;
}
};

```

```

int main() {
    LinkedList linkedList;
    int pilihan;
    string nama, nim;
    int posisi;
    while (true) {
        cout << endl;
        linkedList.print();
        cout << endl;
        cout << "=====
<< endl;
        cout << "PROGRAM SINGLE LINKED LIST NON-
CIRCULAR"<<endl;
        cout << "=====
<< 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 << endl;

        cout << "Pilih Operasi: ";
        cin >> pilihan;

        cout << endl;

        switch (pilihan) {
            case 1:
                cout << "Tambah Depan"<<endl;
                cout << endl;
                cout << "Masukkan Nama : ";

```

```
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.addFirst(nama, nim);
        break;
    case 2:
        cout << "Tambah Belakang"<<endl;
        cout << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.addLast(nama, nim);
        break;
    case 3:
        cout << "Tambah Tengah"<<endl;
        cout << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.addMiddle(nama, nim, posisi);
        break;
    case 4:
        cout << "Ubah Depan"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.changeFirst(nama, nim);
        break;
    case 5:
        cout << "Ubah Belakang"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
```

```

        linkedList.changeLast(nama, nim);
        break;
    case 6:
        cout << "Ubah Tengah"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.changeMiddle(nama, nim,
posisi);

        break;
    case 7:
        cout << "Hapus Depan"<<endl;
        cout << endl;
        linkedList.deleteFirst();
        break;
    case 8:
        cout << "Hapus Belakang"<<endl;
        cout << endl;
        linkedList.deleteLast();
        break;
    case 9:
        cout << "Hapus Tengah"<<endl;
        cout << endl;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.deleteMiddle(posisi);
        break;
    case 10:
        cout << "Hapus List"<<endl;
        cout << endl;
        linkedList.deleteAll();
        break;
    case 11:
        cout << "Tampilkan"<<endl;
        cout << endl;
        linkedList.print();
        break;

```

```

        case 0:
            exit(0);
        default:
            cout << "Pilihan tidak valid"<<endl;
    }
}

return 0;
}

```

## Output

```

Linked List kosong

=====
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

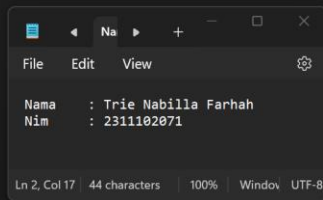
Pilih Operasi: 1

Tambah Depan

Masukkan Nama : denis
Masukkan NIM : 23300005

Data telah ditambahkan

```



```

=====
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

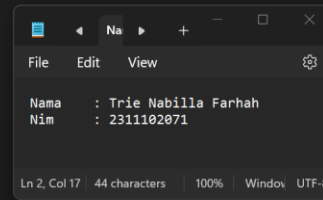
Pilih Operasi: 1

Tambah Depan

Masukkan Nama : farel
Masukkan NIM : 23300003

Data telah ditambahkan

```



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====

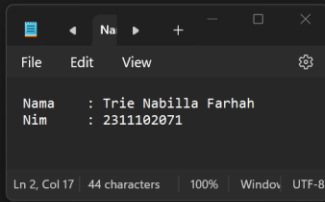
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 1

Tambah Depan

Masukkan Nama : nabilla
Masukkan NIM : 2311102071

Data telah ditambahkan
```



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====

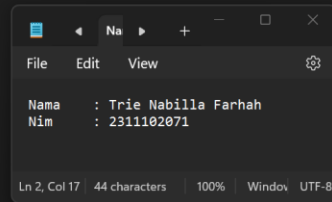
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 1

Tambah Depan

Masukkan Nama : jawad
Masukkan NIM : 23300001

Data telah ditambahkan
```



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====

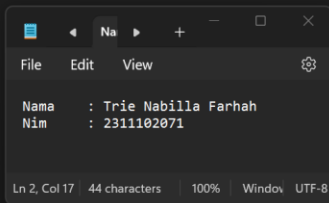
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 2

Tambah Belakang

Masukkan Nama : gahar
Masukkan NIM : 23300040

Data telah ditambahkan
```





```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

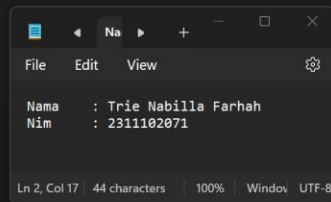
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 2

Tambah Belakang

Masukkan Nama : udin  
Masukkan NIM : 23300048

Data telah ditambahkan



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

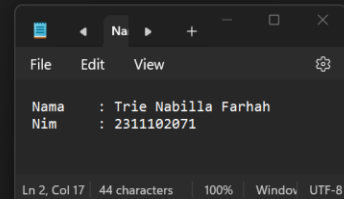
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 2

Tambah Belakang

Masukkan Nama : ucok  
Masukkan NIM : 23300050

Data telah ditambahkan



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

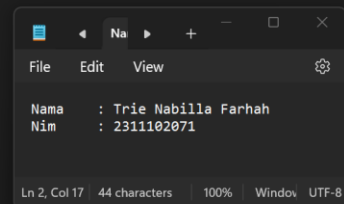
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 2

Tambah Belakang

Masukkan Nama : budi  
Masukkan NIM : 23300099

Data telah ditambahkan



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====

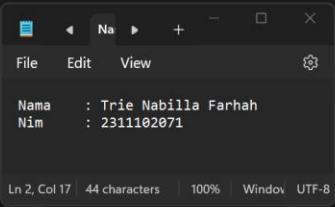
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 3

Tambah Tengah

Masukkan Nama : anis
Masukkan NIM : 23300008
Masukkan Posisi : 5

Data telah ditambahkan
```



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====

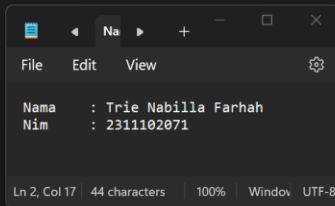
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 3

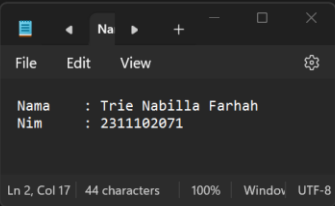
Tambah Tengah

Masukkan Nama : bowo
Masukkan NIM : 23300015
Masukkan Posisi : 6

Data telah ditambahkan
```



```
===== DATA MAHASISWA =====
=====
NAMA      NIM
jawad     23300001
nabilla   2311102071
farel     23300003
denis     23300005
anis      23300008
bowo      23300015
gahar     23300040
udin      23300048
ucok      23300050
budi      23300099
```



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

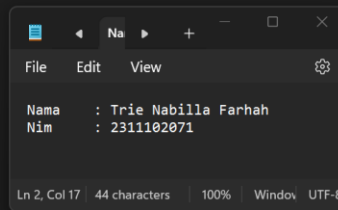
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 3

Tambah Tengah

Masukkan Nama : wati  
Masukkan NIM : 2330004  
Masukkan Posisi : 4

Data telah ditambahkan



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

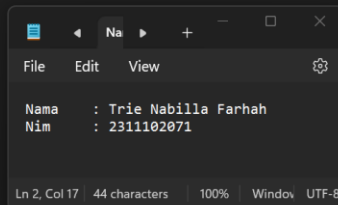
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 9

Hapus Tengah

Masukkan Posisi : 5

Data di posisi 5 telah dihapus



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

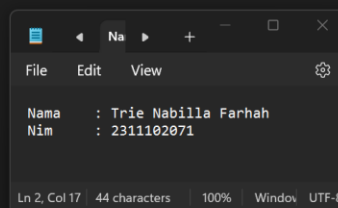
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 1

Tambah Depan

Masukkan Nama : owi  
Masukkan NIM : 2330000

Data telah ditambahkan



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

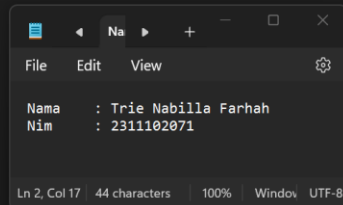
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 2

Tambah Belakang

Masukkan Nama : david  
Masukkan NIM : 23300045

Data telah ditambahkan



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

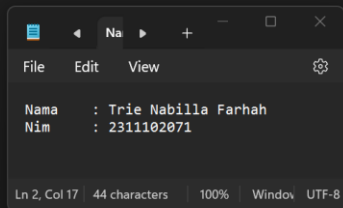
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 6

Ubah Tengah

Masukkan Nama Baru : idin  
Masukkan NIM Baru : 23300045  
Masukkan Posisi : 9

Data di posisi 9 telah diubah



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

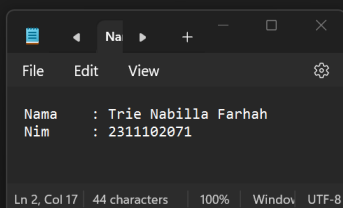
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 5

Ubah Belakang

Masukkan Nama Baru : lucy  
Masukkan NIM Baru : 23300101

Data di belakang telah diubah



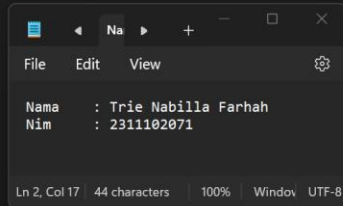
```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

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: 7

Hapus Depan

Data di depan telah dihapus



```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

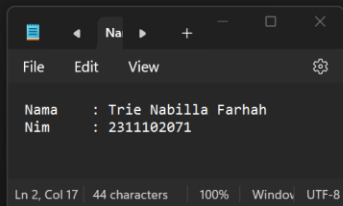
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan
0. Keluar

Pilih Operasi: 4

Ubah Depan

Masukkan Nama Baru : bagus  
Masukkan NIM Baru : 2330002

Data di depan telah diubah



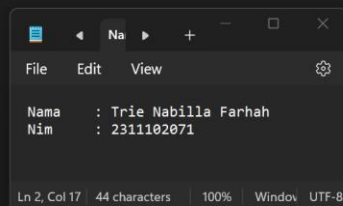
```
=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
```

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: 8

Hapus Belakang

Data di belakang telah dihapus



```
=====
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: 11

===== DATA MAHASISWA =====
=====
NAMA      NIM
bagas     2330002
nabilla   2311102071
farel     2330003
wati      2330004
anis      2330008
bowo      2330015
gahar     2330040
idin      2330045
ucok      2330050
budi      2330099
```

## Deskripsi Program

Program diatas merupakan contoh implementasi dari linked list non-circular. Program ini terdiri dari beberapa fungsi yang digunakan untuk mengelola linked list, seperti `addFirst()`, `addLast()`, `addMiddle()`, `changeFirst()`, `changeLast()`, `changeMiddle()`, `deleteFirst()`, `deleteLast()`, `deleteMiddle()`, `print()`, dan `deleteAll()`.

- Fungsi `addFirst()` digunakan untuk menambahkan node baru di depan linked list.
- Fungsi `addLast()` digunakan untuk menambahkan node baru di belakang linked list.
- Fungsi `addMiddle()` digunakan untuk menambahkan node baru di tengah linked list.
- Fungsi ini memeriksa apakah posisi yang diminta lebih besar dari jumlah node pada linked list atau kurang dari 1.
- Fungsi `changeFirst()` digunakan untuk mengubah data pada node pertama pada linked list.
- Fungsi `changeLast()` digunakan untuk mengubah data pada node terakhir pada linked list.
- Fungsi `changeMiddle()` digunakan untuk mengubah data pada node tertentu pada linked list.
- Fungsi `deleteFirst()` digunakan untuk menghapus node pertama pada linked list.
- Fungsi `deleteLast()` digunakan untuk menghapus node terakhir pada linked list. Jika linked list hanya memiliki satu node, maka fungsi akan menghapus node terakhir dan menetapkan head menjadi NULL. Jika linked list memiliki lebih dari satu node, maka fungsi akan berjalan pada setiap node sampai node sebelum node terakhir, lalu menghapus node terakhir dan menghapusnya dari memori.

- Fungsi `deleteMiddle()` digunakan untuk menghapus node tertentu pada linked list. Fungsi ini memeriksa apakah posisi yang diminta lebih besar dari jumlah node pada linked list atau kurang dari 1.
- Fungsi `print()` digunakan untuk menampilkan data pada setiap node pada linked list.
- Fungsi `deleteAll()` digunakan untuk menghapus semua node pada linked list dan menghapusnya dari memori. Fungsi ini akan berjalan pada setiap node pada linked list hingga semua node telah dihapus.

Program juga memiliki program utama (main) yang menjalankan beberapa perintah dasar pada linked list, seperti menambahkan node, mengubah node, menghapus node, dan menampilkan node. Program utama juga meminta masukan dari pengguna untuk melakukan perintah tersebut.

#### **D. KESIMPULAN**

Linked List Non-Circular adalah struktur data yang terdiri dari sekumpulan node yang saling berhubungan, dimana setiap node memiliki penunjuk ke node berikutnya dalam urutan linier. Node terakhir dari daftar tertaut ini menunjuk ke null, menandakan akhir dari Linked List. Linked List Non-Circular memiliki beberapa keuntungan yaitu, kemudahan implementasi dan penggunaan memori yang efisien, tetapi tidak mengizinkan perulangan atau iterasi terbatas tanpa mekanisme tambahan.

Linked List Circular memiliki sifat unik yaitu simpul terakhir menunjuk ke simpul pertama, sehingga membentuk lingkaran. Hal ini memungkinkan iterasi tanpa batas dan cocok untuk aplikasi yang memerlukan iterasi berulang. Namun, penggunaan Linked List Circular memerlukan penanganan khusus untuk mencegah pengulangan tak terbatas, dan mungkin memerlukan lebih banyak memori.

Kedua jenis daftar tertaut ini memiliki kegunaan dan manfaatnya masing-masing. Penting untuk memilih jenis yang memenuhi persyaratan aplikasi Anda dan memahami karakteristik serta perilakunya.



## E. REFERENSI

Asisten praktikum, “Modul 4 Linked List Circular dan Linked List Non Circular”, learning Management System, 2024

Trivusi. (16 September 2022). *Struktur Data Linked List : Pengertian, Karakteristik, dan jenis-jenisnya*. Diakses pada 14 April 2024, dari <https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html>

widyapitaloka. (12 mei 2020). *Circular Linked List*. Diakses pada 14 April 2024, dari <https://ulinngaprawyblog.wordpress.com/2020/05/12/circular-linked-list/>