

**LAPORAN PRAKTIKUM**  
**MODUL 4**  
**LINKED LIST CIRCULAR DAN NON CIRCULAR**



**Disusun oleh:**  
**MARSHELY AYU ISWANTO**  
**2311102073**

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

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

# **BAB I**

## **TUJUAN PRAKTIKUM**

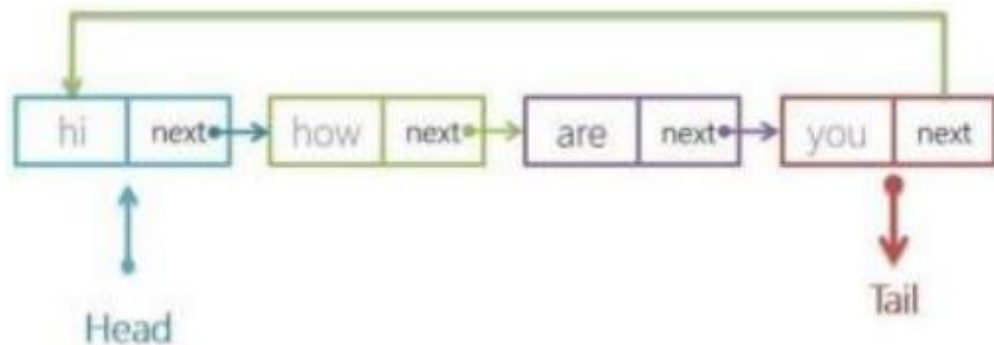
1. Mengetahui dan memahami linked list circular dan non circular.
2. Membuat linked list circular dan non circular
3. Dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

## BAB II

### DASAR TEORI

#### 1. Linked List Circular

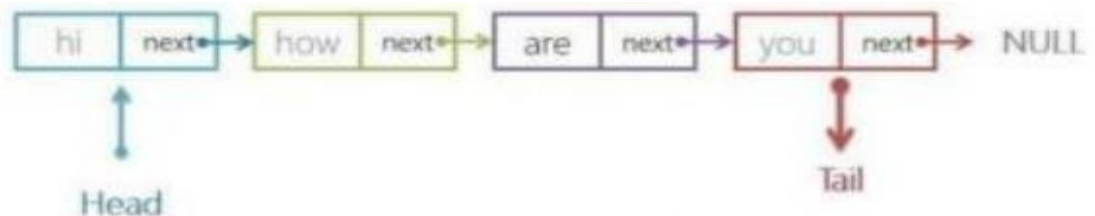
Linked list circular adalah jenis struktur data yang terdiri dari serangkaian node yang saling terhubung, dimana node terakhir kembali kepada node pertama. Untuk membuat program c++ yang mengimplementasikan linked list tersebut, dengan cara menggunakan struktur node yang berisi data dan pointer ke node berikutnya. Operasi-operasi seperti penambahan, penghapusan, dan pencarian kemudian diimplementasikan menggunakan manipulasi pointer. 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.



**Gambar 2** *Single Linked List Circular*

#### 2. Linked List Non Circular

Linked List Non Circular adalah jenis struktur data yang terdiri dari serangkaian node yang saling terhubung, dimana setiap node memiliki data dan pointer ke node berikutnya, tetapi pointer terakhir pada node tidak menunjuk ke node pertama. Kelebihan dari linked list non circular yaitu tidak memerlukan alokasi memori yang kontinu seperti pada array.



**Gambar 1** *Single Linked List Non Circular*

## BAB III

### GUIDED

#### 1. GUIDED 1

##### Linked List Non Circular

##### SOURCE CODE

```
#include <iostream>

using namespace std;

// PROGRAM SINGLE LINKED LIST NON-CIRCULAR

// Deklarasi struct node
struct Node
{
    int data;
    Node *next;
};

Node *head; // Deklarasi head
Node *tail; // Deklarasi tail

// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}

// Pengecekan apakah linked list kosong
bool isEmpty()
{
    if (head == NULL)
```

```
{  
    return true;  
}  
else  
{  
    return false;  
}  
}  
  
// Tambah depan  
void insertDepan(int nilai)  
{  
  
    // buat node baru  
    Node *baru = new Node();  
    baru->data = nilai;  
    baru->next = NULL;  
    if (isEmpty() == true)  
    {  
        head = tail = baru;  
        head->next = NULL;  
    }  
    else  
    {  
        baru->next = head;  
        head = baru;  
    }  
}  
  
// Tambah belakang  
void insertBelakang(int nilai)  
{  
  
    // buat node baru  
    Node *baru = new Node();  
    baru->data = nilai;
```

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

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

// Tambah tengah
void insertTengah(int data, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
}
```

```

else if (posisi == 1)
{
    cout << "Posisi bukan posisi tengah" << endl;
}
else
{
    Node *baru, *bantu;
    baru = new Node();
    baru->data = data;

    // tranversing
    bantu = head;
    int nomor = 1;
    while (nomor < posisi - 1)
    {
        bantu = bantu->next;
        nomor++;
    }

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

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

```

```

        delete hapus;
    }
    else
    {
        head = tail = NULL;
    }
}
else
{
    cout << "Linked list masih kosong" << endl;
}
}

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

```



```

        head = tail = NULL;

    }

}

else
{
    cout << "Linked list masih kosong" << endl;
}

}

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

```

```

        bantu = bantu->next;
        nomor++;
    }
    sebelum->next = bantu;
    delete hapus;
}

// ubah depan
void ubahDepan(int data)
{
    if (isEmpty() == 0)
    {
        head->data = data;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

// ubah tengah
void ubahTengah(int data, int posisi)
{
    Node *bantu;
    if (isEmpty() == 0)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;

```

```

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

else
{
    cout << "Linked list masih kosong" << endl;
}

}

// ubah belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

// Hapus list
void clearList()

```

```

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

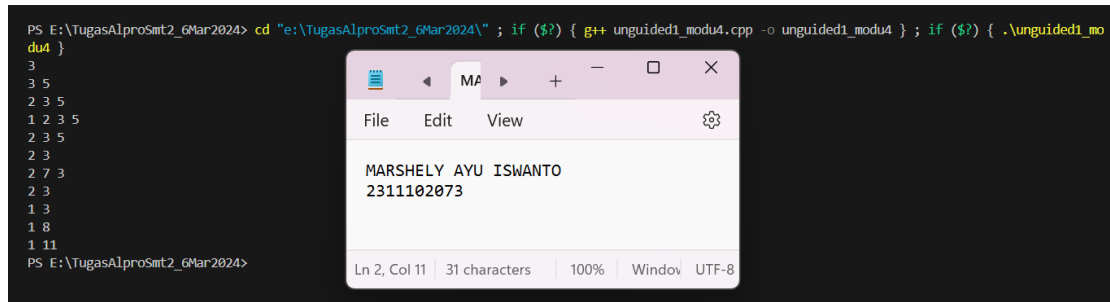
// Tampilkan list
void tampilList()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

int main()

```

```
{  
  
    init();  
    insertDepan(3);  
    tampilList();  
    insertBelakang(5);  
    tampilList();  
    insertDepan(2);  
    tampilList();  
    insertDepan(1);  
    tampilList();  
    hapusDepan();  
    tampilList();  
    hapusBelakang();  
    tampilList();  
    insertTengah(7, 2);  
    tampilList();  
    hapusTengah(2);  
    tampilList();  
    ubahDepan(1);  
    tampilList();  
    ubahBelakang(8);  
    tampilList();  
    ubahTengah(11, 2);  
    tampilList();  
  
    return 0;  
}
```

## SCREENSHOOT PROGRAM



```
PS E:\TugasAlproSmt2_6Mar2024> cd "e:\TugasAlproSmt2_6Mar2024\" ; if ($?) { g++ unguided1_modu4.cpp -o unguided1_modu4 } ; if ($?) { .\unguided1_modu4 }
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 E:\TugasAlproSmt2_6Mar2024>
```

## DESKRIPSI PROGRAM

Program tersebut merupakan program c++ yaitu memanfaatkan linked list non circular. Kode program tersebut merupakan pengimplementasian dengan fungsi-fungsi dasar untuk operasi tambah, hapus, ubah, dan tampilkan. Setiap node pada program ini memiliki pointer yang menunjuk ke node berikutnya, dan node terakhir menunjuk ke NULL. Output dari kode program tersebut adalah hasil dari operasi yang dilakukan pada linked list pada titik tertentu dalam urutan program. Setiap operasi dalam kode program tersebut berdampak pada struktur dan isi dari linked list. Output dari kode program tersebut adalah yang pertama menambah sebuah node dengan nilai 3 lalu menambah sebuah node dengan nilai 5 dibelakang linked list jadinya “3 5” dan seterusnya sesuai operasi.

## 2. GUIDED 2

Linked List Circular

### 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

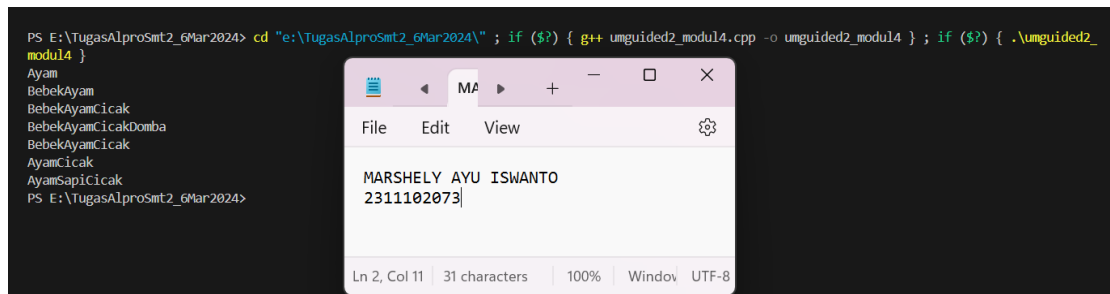
```

```
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



The screenshot shows a terminal window on the left and a text editor window on the right. The terminal window displays the output of a C++ program, which is a linked list containing the following elements: Ayam, BebekAyam, BebekAyamCicak, BebekAyamCicakDomba, BebekAyamCicak, AyamCicak, AyamSapiCicak, and PS E:\TugasAlproSmt2\_6Mar2024>. The text editor window shows the source code of the program, which is a C++ file named umguided2\_modul4.cpp. The code defines a linked list structure and implements functions for adding, deleting, and displaying elements. The output of the program is displayed in the terminal window, showing the elements of the linked list and the prompt for the next operation.

## DESKRIPSI PROGRAM

Program tersebut merupakan program c++ yaitu memanfaatkan linked list non circular, dimana setiap node memiliki pointer yang menunjukan ke node berikutnya, kecuali node terakhir yang menunjuk ke NULL. Program ini memiliki fungsi untuk menambah, menghapus, dan menampilkan elemen-elemen dalam linked list non-circular, serta untuk mengelola linked list secara keseluruhan, seperti inisialisasi dan penghapusan. Output dari program tersebut adalah hasil dari operasi yang dilakukan pada linked list pada titik tertentu dalam urutan program. Output dari kode program tersebut yaitu yang pertama sebuah node dengan nilai “Ayam” ditambahkan setelah linked list, lalu sebuah node ditambahkan dengan nilai “Bebek” didepan linked list menjadi “Bebek Ayam”, dan seterusnya sesuai operasi.

## UNGUIDED

### 1. UNGUIDED 1

Buatlah program menu Linked List Non Circular untuk menyimpan Nama dan NIM mahasiswa, dengan menggunakan input dari user.

1. Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM mahasiswa, berikut contoh tampilan output dari nomor 1:

- 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

```
-Tambah Depan
```

```
Masukkan Nama :
```

```
Masukkan NIM :
```

```
Data telah ditambahkan
```

```
-Tambah Tengah
```

```
Masukkan Nama :
```

```
Masukkan NIM :
```

```
Masukkan Posisi :
```

```
Data telah ditambahkan
```

- Tampilan Operasi Hapus

-Hapus Belakang

Data (nama mahasiswa yang dihapus) berhasil dihapus

-Hapus Tengah

Masukkan posisi :

Data (nama mahasiswa yang dihapus) berhasil dihapus

- Tampilan Operasi Ubah

-Ubah Belakang

Masukkan nama :

Masukkan NIM :

Data (nama lama) telah diganti dengan data (nama baru)

-Ubah Belakang

Masukkan nama :

Masukkan NIM :

Masukkan posisi :

Data (nama lama) telah diganti dengan data (nama baru)

- Tampilan Operasi Tampil Data

DATA MAHASISWA

NAMA NIM

Nama1 NIM1

Nama2 NIM2

**\*Buat tampilan output sebagus dan secantik mungkin sesuai kreatifitas anda masing-masing, jangan terpaku pada contoh output yang diberikan**



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

<b>Nama</b>	<b>NIM</b>
<b>Jawad</b>	<b>23300001</b>
<b>[Nama Anda]</b>	<b>[NIM Anda]</b>
<b>Farrel</b>	<b>23300003</b>
<b>Denis</b>	<b>23300005</b>
<b>Anis</b>	<b>23300008</b>
<b>Bowo</b>	<b>23300015</b>
<b>Gahar</b>	<b>23300040</b>
<b>Udin</b>	<b>23300048</b>
<b>Ucok</b>	<b>23300050</b>
<b>Budi</b>	<b>23300099</b>

3. Lakukan perintah berikut:

- a) Tambahkan data berikut diantara Farrel dan Denis: Wati 23300004
- b) Hapus data Denis
- c) Tambahkan data berikut di awal: Owi 23300000
- d) Tambahkan data berikut di akhir: David 23300100
- e) Ubah data Udin menjadi data berikut: Idin 23300045
- f) Ubah data terkahir menjadi berikut: Lucy 23300101
- g) Hapus data awal
- h) Ubah data awal menjadi berikut: Bagas 23300002

- i) Hapus data akhir
- j) Tampilkan seluruh data

### 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;
    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;
```

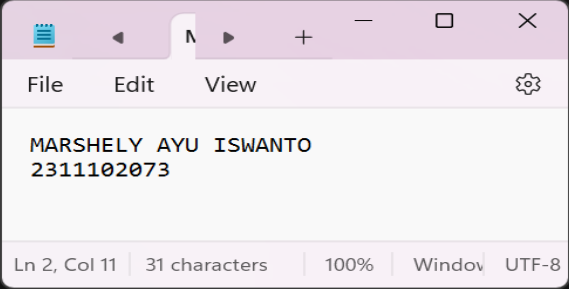
```
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.ubahTengah(nama, nim, posisi);
        break;
    case 7:
        linkedList.hapusDepan();
        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();
        break;
    case 11:
        linkedList.tampilkanData();
        break;
    default:
        cout << "Pilihan tidak valid." << endl;
    }
} while (choice != 12);

return 0;
}
```

## SCREENSHOOT PROGRAM

### Screenshots output no.2

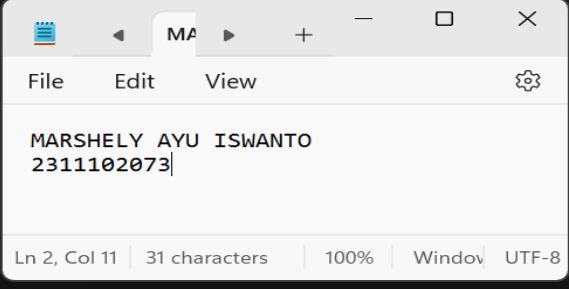
```
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. TAMPILKAN
0. Keluar
Pilih Operasi : 11
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Marshely  2311102073
Farrel    23300003
Denis     23300005
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
```



### Screenshots output no. 3

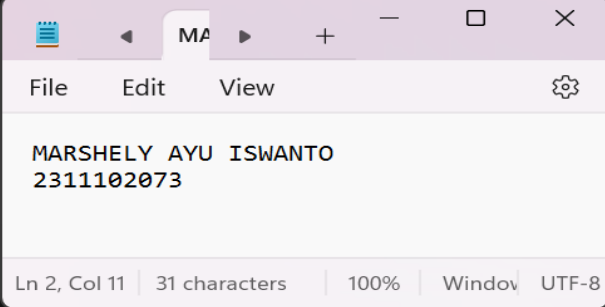
#### Perintah (a)

```
Pilih Operasi : 3
-Tambah Tengah-
Masukkan Nama : Wati
Masukkan NIM : 2330004
Masukkan Posisi : 3
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 : 11
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Marshely  2311102073
Farrel    23300003
Wati      23300004
Denis     23300005
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
```



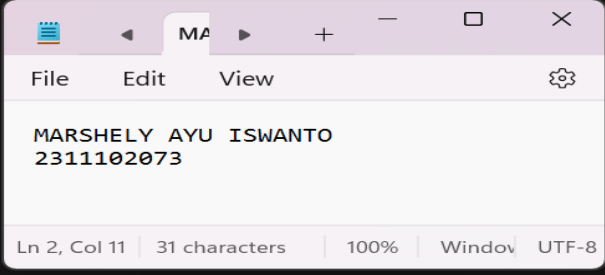
## Perintah (b)

```
Pilih Operasi : 7
Data berhasil 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
Jawad     23300001
Marshely  2311102073
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
Lucy      23300101
```



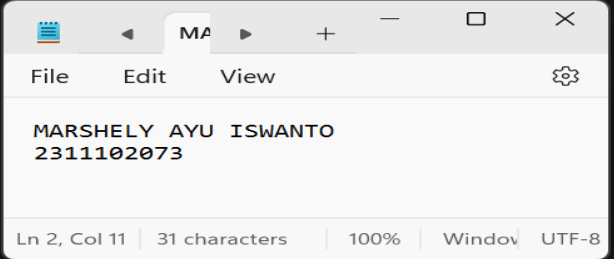
## Perintah (c)

```
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 : 11
DATA MAHASISWA
NAMA      NIM
Owi       23300000
Jawad     23300001
Marshely  2311102073
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
```



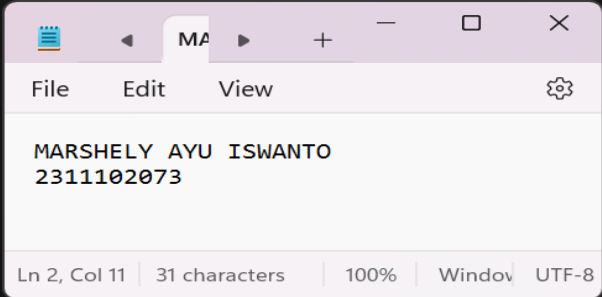
### Perintah (d)

```
Pilih Operasi : 2
-Tambah Belakang-
Masukkan Nama : David
Masukkan NIM : 23300100
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 : 11
DATA MAHASISWA
NAMA      NIM
Owi       2330000
Jawad     23300001
Marshely  2311102073
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
David     23300100
```



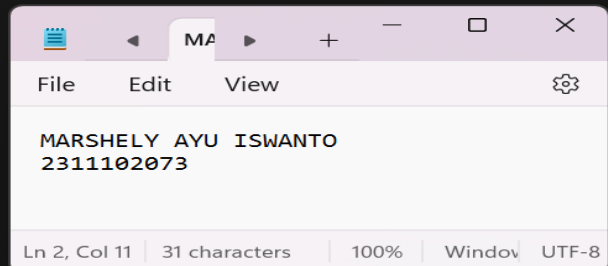
### Perintah (e)

```
Pilih Operasi : 6
-Ubah Tengah-
Masukkan Nama Baru : Idin
Masukkan NIM Baru : 23300045
Masukkan Posisi : 9
Data berhasil 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 : 11
DATA MAHASISWA
NAMA      NIM
Owi       2330000
Jawad     23300001
Marshely  2311102073
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
David     23300100
```



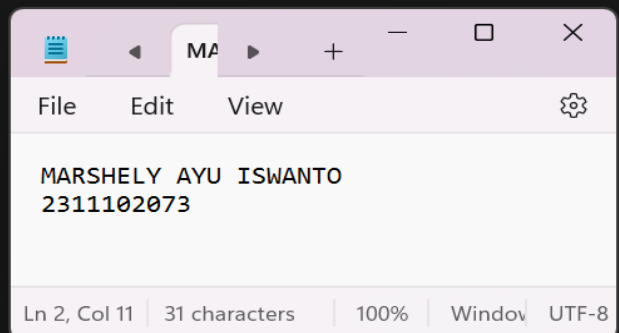
## Perintah (f)

```
Pilih Operasi : 6
-Ubah Tengah-
Masukkan Nama Baru : Idin
Masukkan NIM Baru : 23300045
Masukkan Posisi : 9
Data berhasil 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 : 11
DATA MAHASISWA
NAMA      NIM
Owi       23300000
Jawad     23300001
Marshely  2311102073
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo     23300015
Gahar    23300040
Idin     23300045
Ucok     23300050
Budi     23300099
David    23300100
```



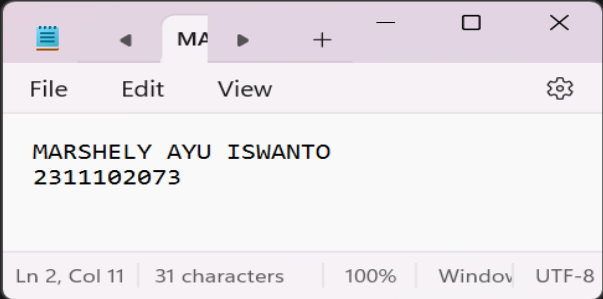
## Perintah (g)

```
Pilih Operasi : 7
Data berhasil 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
Jawad     23300001
Marshely  2311102073
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo     23300015
Gahar    23300040
Idin     23300045
Ucok     23300050
Budi     23300099
Lucy     23300101
```



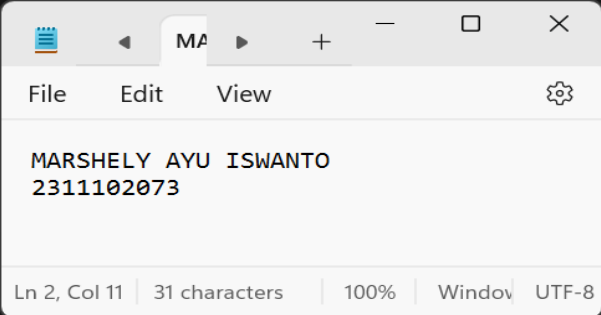
## Perintah (h)

```
Pilih Operasi : 4
-Ubah Depan-
Masukkan Nama Baru : Bagus
Masukkan NIM Baru : 2330002
Data berhasil 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 : 11
DATA MAHASISWA
NAMA      NIM
Bagas     2330002
Marshely  2311102073
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
Lucy      23300101
```

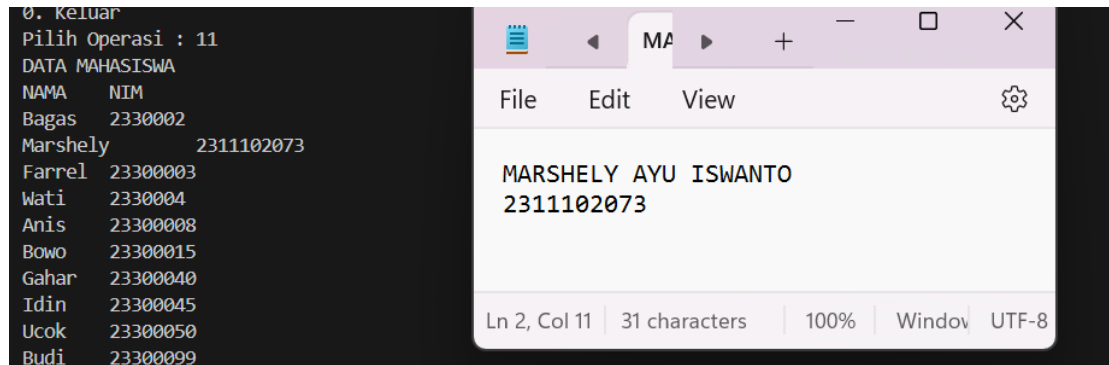


## Perintah (i)

```
Pilih Operasi : 8
Data berhasil 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
Marshely  2311102073
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
```



## Perintah (j)



The image shows a terminal window on the left and a code editor window on the right. The terminal window displays the output of a C++ program, which is a linked list of student data. The code editor window shows the source code of the program, specifically the part that prints the student data.

```
0. Keluar
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Bagas   2330002
Marshely 2311102073
Farrel  23300003
Wati    2330004
Anis    23300008
Bowo    23300015
Gahar   23300040
Idin    23300045
Ucok    23300050
Budi    23300099
```

```
File Edit View
MARSHELY AYU ISWANTO
2311102073
Ln 2, Col 11 | 31 characters | 100% | Window UTF-8
```

## DESKRIPSI PROGRAM

Program diatas merupakan program c++ yang memanfaatkan linked list non-circular, dalam program tersebut, linked list dibuat menggunakan struktur node yang mempunyai dua string yaitu nama dan nim, serta melibatkan pointer next yang menunjukan ke node selanjutnya dalam linked list ini. Output dari program ini adalah hasil dari operasi kode program yang telah dilakukan. Output yang pertama dari kode program tersebut adalah menambahkan seluruh nama dan nim sesuai perintah lalu melakukan perintah menambahkan data, hapus data, mengubah data, dan menampilkan seluruh data.



## **BAB IV**

### **KESIMPULAN**

1. Linked list circular dan linked list non circular masing-masing mempunyai struktur data linear yang terdiri dari serangkaian node yang saling terhubung.
2. Linked list circular mempunyai setiap node yang memiliki pointer yang menunjuk kembali node pertama, sedangkan untuk linked list non circular memiliki setiap node yang hanya menunjuk ke node berikutnya.
3. Linked list circular lebih kompleks, sedangkan linked list non circular lebih sederhana.

## DAFTAR PUSTAKA

Antonius Rachmat C, S. (n.d.). *Single Linked List Circular*. Retrieved from <https://www.scribd.com>: <https://www.scribd.com/doc/122367708/Single-Linked-List-Circular>

*Single Linked List Non Circular*. (n.d.). Retrieved from <https://repository.dinus.ac.id/docs>:  
[https://repository.dinus.ac.id/docs/ajar/Pertemuan\\_11\\_Single\\_Linked\\_List\\_Circular.pdf](https://repository.dinus.ac.id/docs/ajar/Pertemuan_11_Single_Linked_List_Circular.pdf)