

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



**Disusun oleh:**  
**Rasyid Nafsyarie**  
**NIM : 2311102011**

**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. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
2. Praktikan dapat membuat linked list circular dan non circular.
3. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

## **BAB II**

### **DASAR TEORI**

Apa yang dimaksud dengan Single Linked List? Single Linked List merupakan sekumpulan dari node yang saling terhubung dengan node lain melalui sebuah pointer. Single Linked List hanya memiliki satu arah dan tidak memiliki dua arah atau bulak-balik, dua tersebut disebut dengan Double Linked List. Fungsi dari Linked List adalah sekumpulan elemen yang bertipe sama, yang mempunyai keterurutan tertentu, yang setiap elemennya terdiri dari dua bagian Linked List juga merupakan suatu cara untuk menyimpan data dengan struktur sehingga dapat secara otomatis menciptakan suatu tempat baru untuk menyimpan data yang diperlukan. Perbedaan array dan single linked list, Array adalah kumpulan objek data yang mirip satu sama lain dan disimpan di lokasi memori secara berurutan. Sementara itu, linked list merupakan sekumpulan data yang berisi urutan elemen dalam strukturnya. setiap elemen saling terkait dengan elemen berikutnya

Circular Single Linked List adalah Single Linked List yang pointernya menunjukpada dirinya sendiri. Jika Single Linked List tersebut terdiri dari beberapa node,maka pointer next pada node terakhir akan menunjuk ke node terdepannya.Pengertiannya sendiri terdiri dari 2 kata yakni single yang artinya field pointer nyahanya satu buah saja dan satu arah dan circular yang artinya pointer next nya akanmenunjuk pada dirinya sendiri sehingga berputar.Perbedaan antara Circular Single Linked List dengan Single Linked List yakni,Circular Single Linked List merupakan suatu linked leist dimana tail menunjuk kehead. Jadi, tidak ada pointer yang menunjuk NULL. Sedangkan pada Single LinkedList merupakan suatu linked list yang hanya memiliki satu variabel pointer saja.

## BAB III

### GUIDED

#### 1. GUIDED 1

##### SOURCE CODE

```
#include <iostream>

using namespace std;

// PROGRAM SINGLE LINKED LIST NON-CIRCULAR

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

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

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

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

```
        return true;
    }
    else
    {
        return false;
    }
}

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

// Tambah belakang
void insertBelakang(int nilai)
{
    // buat node baru
    Node *baru = new Node();
    baru->data = nilai;
    baru->next = NULL;
```

```

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

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

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

```

```

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

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

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

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

```

```

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

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

```



```

    }

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

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

```

```

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

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

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

```

```

        else
        {
            int nomor = 1;
            bantu = head;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

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

// Hapus list
void clearList()
{

```

```

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

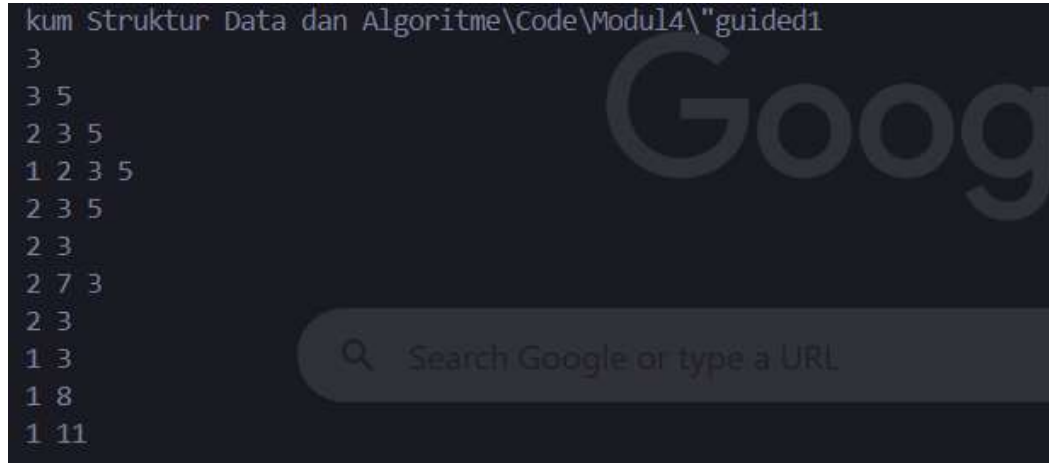
// Tampilkan list
void tampilList()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "Linked list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

int main()
{

```

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

## SCREENSHOOT PROGRAM



## DESKRIPSI PROGRAM

Kode ini mencakup operasi dasar seperti menambahkan simpul di depan, di belakang, di tengah, menghapus simpul, dan mengubah nilai simpul. Dengan menggunakan Linked List, kita dapat dengan mudah memanipulasi dan mengelola kumpulan data yang dinamis.

## 2. GUIDED 2

### SOURCE CODE

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

class Node {
public:
    int data;
    string kata;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
```

```
public:
    Node* head;
    Node* tail;
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void push(int data, string kata) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->kata = kata;
        newNode->prev = nullptr;
        newNode->next = head;
        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }
        head = newNode;
    }

    void pop() {
        if (head == nullptr) {
            return;
        }
        Node* temp = head;
        head = head->next;
        if (head != nullptr) {
            head->prev = nullptr;
        } else {
            tail = nullptr;
        }
        delete temp;
    }
}
```

```

    }

    bool update(int oldData, int newData, string newKata) {
        Node* current = head;
        while (current != nullptr) {
            if (current->data == oldData) {
                current->data = newData;
                current->kata = newKata;
                return true;
            }
            current = current->next;
        }
        return false;
    }

    void deleteAll() {
        Node* current = head;
        while (current != nullptr) {
            Node* temp = current;
            current = current->next;
            delete temp;
        }
        head = nullptr;
        tail = nullptr;
    }

    void display() {
        Node* current = head;
        while (current != nullptr) {
            cout << current->data << " ";
            cout << current->kata << endl;
            current = current->next;
        }
        cout << endl;
    }

```



```

    }
};

int main() {
    DoublyLinkedList list;
    while (true) {
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Exit" << endl;
        int choice;
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                int data;
                string kata;
                cout << "Enter data to add: ";
                cin >> data;
                cout << "Enter kata to add: ";
                cin >> kata;
                list.push(data, kata);
                break;
            }
            case 2: {
                list.pop();
                break;
            }
            case 3: {
                int oldData, newData;
                string newKata;
                cout << "Enter old data: ";

```

```

        cin >> oldData;
        cout << "Enter new data: ";
        cin >> newData;
        cout << "Enter new kata: ";
        cin >> newKata;
        bool updated = list.update(oldData,
        newData, newKata);
        if (!updated) {
            cout << "Data not found" << endl;
        }
        break;
    }
    case 4: {
        list.deleteAll();
        break;
    }
    case 5: {
        list.display();
        break;
    }
    case 6: {
        return 0;
    }
    default: {
        cout << "Invalid choice" << endl;
        break;
    }
}

return 0;
}

```

## SCREENSHOOT PROGRAM

```
kum Struktur Data dan Algoritme\Code\Modul4\guided2
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
```

## DESKRIPSI PROGRAM

Struct Node: Digunakan untuk mendefinisikan struktur data simpul dalam Linked List. Setiap simpul memiliki dua bagian, yaitu data (bertipe string) dan pointer next (menunjuk ke simpul berikutnya). Fungsi clearList(): Digunakan untuk menghapus semua simpul dalam Linked List.

## 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

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

Nama	NIM
Jawad	23300001
Rasyid Nafsyarie	2311102011
Farrel	23300003
Denis	23300005

Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

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 terkahir menjadi berikut:

Lucy 23300101

g) Hapus data awal

h) Ubah data awal menjadi berikut:

Bagas 2330002

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; // Menu untuk hapus
list
    cout << "11. Tampilkan Data" << endl;
    cout << "12. Keluar" << endl;
    cout << "Pilih Operasi : ";
    cin >> choice;

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

```

```
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.tambahBelakang(nama, nim);
        break;
    case 3:
        cout << "-Tambah Tengah-" << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.tambahTengah(nama, nim, posisi);
        break;
    case 4:
        cout << "-Ubah Depan-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.ubahDepan(nama, nim);
        break;
    case 5:
        cout << "-Ubah Belakang-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.ubahBelakang(nama, nim);
        break;
    case 6:
        cout << "-Ubah Tengah-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
```

```

        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.ubahTengah(nama, nim, posisi);
        break;
    case 7:
        linkedList.hapusDepan();
        break;
    case 8:
        linkedList.hapusBelakang();
        break;
    case 9:
        cout << "-Hapus Tengah-" << endl;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.hapusTengah(posisi);
        break;
    case 10:
        linkedList.hapusList(); // Hapus List
        break;
    case 11:
        linkedList.tampilkanData();
        break;
    case 12:
        cout << "Program selesai." << endl;
        break;
    default:
        cout << "Pilihan tidak valid." << endl;
    }
} while (choice != 12);

return 0;
}

```

## SCREENSHOOT PROGRAM

DATA MAHASISWA		
NAMA	NIM	
Jawad	23300001	
Rasyid	2311102011	
Farrel	23300003	
Denis	23300005	
Anis	23300008	
Bowo	23300015	
Gahar	23300040	
Udin	23300048	
Ucok	23300050	
Budi	23300099	
PROGRAM SINGLE LINKED LIST NON-CIRCULAR		

*1.1. Tampilan data yang sudah dimasukkan (untuk perintah kedua)*

DATA MAHASISWA		
NAMA	NIM	
Jawad	23300001	
Rasyid	2311102011	
Farrel	23300003	
Wati	23300004	
Denis	23300005	
Anis	23300008	
Bowo	23300015	
Gahar	23300040	
Udin	23300048	
Ucok	23300050	
Budi	23300099	
PROGRAM SINGLE LINKED LIST NON-CIRCULAR		

*1.2. Menambahkan data wati diantara farrel dan denis*



```

DATA MAHASISWA
NAMA      NIM      Lucy  23300101
Jawad     23300001  g) Hapus data awal
Rasyid    2311102011 h) Ubah data awal menjadi berikut:
Farrel     23300003      Bagus  23300002
Wati       23300004
Anis       23300008  i) Hapus data akhir
Bowo       23300015  j) Tampilkan seluruh data
Gahar      23300040
Udin       23300048
Ucok       23300050
Budi       23300099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

```

### 1.3. Menghapus data denis

```

DATA MAHASISWA
NAMA      NIM      Bagus  23300002
Owi        23300000  i) Hapus data akhir
Jawad      23300001  j) Tampilkan seluruh data
Rasyid     2311102011
Farrel      23300003
Wati        23300004
Anis        23300008
Bowo        23300015
Gahar       23300040
Udin        23300048
Ucok        23300050
Budi        23300099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

```

### 1.4. Menambahkan data owi diawal

DATA MAHASISWA		
NAMA	NIM	
Owi	2330000	h) Ubah data awal menjadi berikut:
		<b>Bagas 2330002</b>
Jawad	23300001	i) Hapus data akhir
Rasyid	2311102011	j) Tampilkan seluruh data
Farrel	23300003	
Wati	23300004	
Anis	23300008	
Bowo	23300015	
Gahar	23300040	
Udin	23300048	
Ucok	23300050	
Budi	23300099	
David	23300100	

#### 1.5. Menambahkan data David diakhir

DATA MAHASISWA		
NAMA	NIM	
Owi	2330000	h) Ubah data awal menjadi berikut:
		<b>Bagas 2330002</b>
Jawad	23300001	i) Hapus data akhir
Rasyid	2311102011	j) Tampilkan seluruh data
Farrel	23300003	
Wati	23300004	
Anis	23300008	
Bowo	23300015	
Gahar	23300040	
Idin	23300045	
Ucok	23300050	
Budi	23300099	
David	23300100	

PROGRAM SINGLE LINKED LIST NON-CIRCULAR

1. Tambah Depan

#### 1.6. Mengubah data udin menjadi idin

DATA MAHASISWA		h) Ubah data awal menjadi berikut:
NAMA	NIM	<b>Bagas 2330002</b>
Owi	2330000	
Jawad	23300001	i) Hapus data akhir
Rasyid	2311102011	j) Tampilkan seluruh data
Farrel	23300003	
Wati	2330004	
Anis	23300008	
Bowo	23300015	
Gahar	23300040	
Idin	23300045	
Ucok	23300050	
Budi	23300099	
Lucy	23300101	

#### 1.7. Mengubah data terakhir menjadi Lucy

DATA MAHASISWA		h) Ubah data awal menjadi berikut:
NAMA	NIM	<b>Bagas 2330002</b>
Jawad	23300001	i) Hapus data akhir
Rasyid	2311102011	j) Tampilkan seluruh data
Farrel	23300003	
Wati	2330004	
Anis	23300008	
Bowo	23300015	
Gahar	23300040	
Idin	23300045	
Ucok	23300050	
Budi	23300099	
Lucy	23300101	

#### 1.8. Menghapus data awal

DATA MAHASISWA	
NAMA	NIM
Bagas	2330002
Rasyid	2311102011
Farrel	23300003
Wati	2330004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099
Lucy	23300101

*1.9. Mengubah data awal menjadi bagas*

DATA MAHASISWA	
NAMA	NIM
Bagas	2330002
Rasyid	2311102011
Farrel	23300003
Wati	2330004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099

*1.10. Menghapus data akhir dan menampilkan seluruh data*

## DESKRIPSI PROGRAM

Program ini memiliki beberapa fungsi seperti tambahDepan(string nama, string nim): Menambahkan data baru ke depan linked list. tambahBelakang(string nama, string nim): Menambahkan data baru ke belakang linked list. tambahTengah(string nama, string nim, int posisi): Menambahkan data baru pada posisi tertentu dalam linked list.

## **BAB IV**

### **KESIMPULAN**

Setelah melakukan pembelajaran mengenai tipe data di Bahasa Pemrograman C++ berikut poin utama yang telah dipelajari :

1. Linked List Non Circular : Memiliki dua pointer, yaitu head dan tail, yang menunjukkan awal dan akhir dari linked list.
2. Linked List Circular : Memiliki pointer head dan tail, tetapi tail tidak pernah bernilai NULL, dan terhubung kembali dengan head, membentuk suatu lingkaran.
3. Kesimpulannya, kedua jenis linked list memiliki perbedaan pada cara penyimpanan dan manipulasinya. Linked list non circular memiliki akhir yang ditandai dengan NULL pada pointer tail, sedangkan linked list circular tidak memiliki akhir yang jelas karena tail selalu terhubung kembali dengan head.

## DAFTAR PUSTAKA

Hari, Mahardika. 2022. *Apa itu Single Linked List Non Circular dan contoh program*, (Online), (<https://www.studocu.com/id/document/institut-teknologi-sepuluh-nopember/algoritma-pemrograman-komputer-algorithm-and-computer-programming/apa-itu-single-linked-list-non-circular-dan-contoh-program/45206382>, diakses 05 April 2024).