

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



**Disusun oleh:**

**Arjun Werdho Kumoro**

**NIM : 2311102009**

**Dosen Pengampu:**

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

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

# **BAB I**

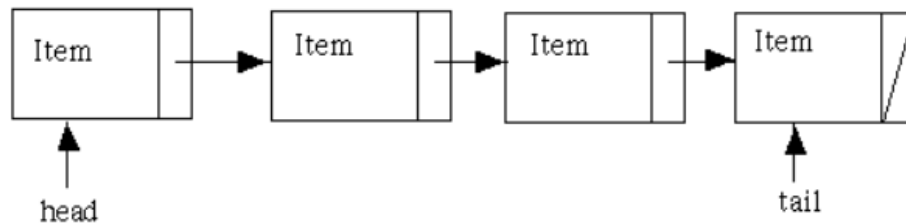
## **TUJUAN PRAKTIKUM**

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

Linked List atau dikenal juga dengan sebutan senarai berantai adalah struktur 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 didalam 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. Single Linked List merupakan suatu linked list yang hanya memiliki satu variabel pointer saja. Dimana pointer tersebut menunjuk ke node selanjutnya. Biasanya field pada tail menunjuk ke NULL. Contoh :



Contoh kode :

```
struct Mahasiswa{  
  
char nama[25];  
  
int usia;  
  
struct Mahasiswa *next;  
  
}*head,*tail;
```

## 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

```
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
```

## DESKRIPSI PROGRAM

Program diatas merupakan program linked list non circular yang menampilkan output angka-angka seperti diatas.

## 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;

```

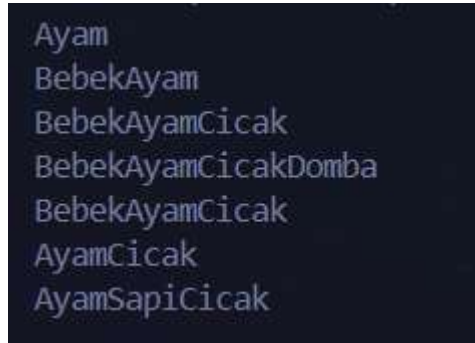
```

        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



```
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
```

## DESKRIPSI PROGRAM

Program diatas menampilkan sebuah output berupa string. Struct Node: Digunakan untuk mendeklarasikan data simpul yang ada didalam Linked List. Setiap simpul memiliki dua bagian, yaitu data yang bertipe string dan pointer next.

## 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
Arjun	2311102009
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 option;
    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 >> option;

    switch (option) {
        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 ada" << endl;
    }
} while (option != 12);

return 0;
}

```

## SCREENSHOOT PROGRAM

```
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Arjun     2311102009
Farrel    23300003
Denis     23300005
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Data
```

2. Menginput data yang diinginkan

```
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Arjun     2311102009
Farrel    23300003
Wati      23300004
Denis     23300005
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
2. Tambah Data
```

3a. Menambahkan data wati



```
DATA MAHASISWA
NAMA      NIM
Jawad     23300001
Arjun     2311102009
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

3b. Hapus denis

```
DATA MAHASISWA
NAMA      NIM
Owi       23300000
Jawad     23300001
Arjun     2311102009
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

3c. Tambah Owi

```
DATA MAHASISWA
NAMA      NIM
Owi       2330000
Jawad     23300001
Arjun     2311102009
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
David     23300100
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

3d. Tambah David

```
DATA MAHASISWA
NAMA      NIM
Owi       2330000
Jawad     23300001
Arjun     2311102009
Farrel    23300003
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
David     23300100
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

3e. ubah udin

```
DATA MAHASISWA
NAMA      NIM
Owi        2330000
Jawad      23300001
Arjun      2311102009
Farrel     23300003
Wati       23300004
Anis       23300008
Bowo       23300015
Gahar      23300040
Idin       23300045
Ucok       23300050
Budi       23300099
Lucy       23300101
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Data
```

3f. ubah data terakhir

```
DATA MAHASISWA
NAMA      NIM
Jawad      23300001
Arjun      2311102009
Farrel     23300003
Wati       23300004
Anis       23300008
Bowo       23300015
Gahar      23300040
Idin       23300045
Ucok       23300050
Budi       23300099
Lucy       23300101
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Data
```

3g. hapus data awal

```
DATA MAHASISWA
NAMA      NIM
Bagas     2330002
Arjun     2311102009
Farrel    23300003
Wati      2330004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
Lucy      23300101
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

3h. ubah data awal

```
DATA MAHASISWA
NAMA      NIM
Bagas     2330002
Arjun     2311102009
Farrel    23300003
Wati      2330004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Depan
```

3i. hapus data akhir

## **DESKRIPSI PROGRAM**

Kode diatas memiliki dua kelas utama, yaitu Node dan LinkedList. Node adalah struct yang mengatur satu node, dengan atribut nama, nim, dan next yang digunakan untuk menyimpan data dan menyimpan pointer ke node berikutnya. LinkedList adalah kelas yang mengatur linked list, dengan atribut head yang digunakan untuk menyimpan pointer ke node pertama 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 adalah data structure yang terdiri dari node-node yang terhubung satu sama lain melalui pointer.
2. Penggunaan linked list dapat mempermudah dalam melakukan operasi seperti menambah, mengubah, atau menghapus data.
3. Linked list dapat digunakan untuk menyimpan data yang berhubungan satu sama lain, seperti data pada sebuah daftar.
4. Double linked list memiliki dua pointer, yaitu pointer yang menunjuk ke node selanjutnya dan pointer yang menunjuk ke node sebelumnya.

## **DAFTAR PUSTAKA**

Annisa Puspa Kirana, S.Kom, M.Kom. 2016. Modul Praktikum Algoritma dan Struktur Data.  
Malang : Universitas Negeri Malang.