

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL IV  
CIRCULAR DAN NON CIRCULAR**



**Disusun Oleh :**

**NAMA : WISNU RANANTA RADITYA  
PUTRA  
NIM : 2311102013**

**Dosen :**

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

**PROGRAM STUDI S1 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**

#### **1) Linked List Non Circular**

Linked list non circular merupakan linked list dengan node pertama (head) dan node terakhir (tail) yang tidak saling terhubung. Pointer terakhir (tail) pada Linked List ini selalu bernilai 'NULL' sebagai pertanda data terakhir dalam list-nya.

#### **2) Linked List Circular**

Linked list circular merupakan linked list yang tidak memiliki akhir karena node terakhir (tail) tidak bernilai 'NULL', tetapi terhubung dengan node pertama (head). Saat menggunakan linked list circular kita membutuhkan dummy node atau node pengecoh yang biasanya dinamakan dengan node current supaya program dapat berhenti menghitung data ketika node current mencapai node pertama (head).

Linked list circular dapat digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu, daftar pesan dalam antrian, atau penggunaan memori berulang dalam suatu aplikasi.

# BAB III

## GUIDED

### Guided 1

```
#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;
}

```

## Screenshots Output

```
PS C:\Semester 2\Praktikum Strudur Data\Modul_4> &
dowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-
-Error-45gwjo2.mdt' '--pid=Microsoft-MIEngine-Pid-
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 C:\Semester 2\Praktikum Strudur Data\Modul_4>
```

## Deskripsi:

Program diatas merupakan program implementasi dari Linked List Non Circular. Program diatas mendeklarasikan sebuah Struct Node dan menginisialisasi 'tail' dan 'head'. Program diatas juga memiliki fungsi untuk menambahkan elemen di depan, menambahkan elemen di belakang, menambahkan elemen di posisi tertentu, menghapus elemen pertama, menghapus elemen di posisi tertentu, menghapus elemen terakhir, mengubah elemen pertama, mengubah elemen di posisi tertentu, mengubah elemen terakhir, menghapus semua elemen dari linked list, menampilkan isi elemen linked list.

## Guided 2

```
#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;
}

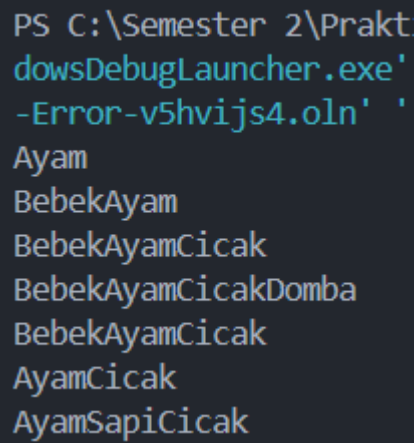
// Tampilkan List
void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
    }
}

```

```
        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;
}
```

## Screenshots Output



```
PS C:\Semester 2\Prakt  
dowsDebugLauncher.exe'  
-Error-v5hvijs4.0ln' '  
Ayam  
BebekAyam  
BebekAyamCicak  
BebekAyamCicakDomba  
BebekAyamCicak  
AyamCicak  
AyamSapiCicak
```

## Deskripsi

Program di atas merupakan implementasi dari linked list circular. Program diatas mendeklarasikan sebuah Struct Node yang memiliki dua anggota yaitu 'string data' untuk menyimpan data dan 'node \*next' sebagai pointer yang menunjukan ke node selanjutnya. Lalu program juga mendeklarasikan variable global head, tail, baru, bantu, hapus sebagai pointer ke node dalam linked list. Program diatas juga memiliki fungsi untuk menambahkan elemen baru di depan, menambahkan elemen baru di belakang, menambahkan elemen baru di posisi tertentu, menghapus elemen pertama, menghapus elemen terakhir, menghapus elemen di posisi tertentu, menghapus semua elemen, menampilkan semua elemen.

## BAB IV

### UNGUIDED

#### Unguided 1

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

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

Node *head = NULL;
Node *tail = NULL;

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

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

void insertDepan(string nama, string nim)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

void insertBelakang(string nama, string nim)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
```

```

    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

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

void insertTengah(string nama, string nim, int posisi)
{
    if (posisi < 1 || posisi > hitungList() + 1)
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        insertDepan(nama, nim);
    }
    else if (posisi == hitungList() + 1)
    {
        insertBelakang(nama, nim);
    }
    else
    {
        Node *baru = new Node();
        baru->nama = nama;
        baru->nim = nim;
        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;
        head = head->next;
        delete hapus;
        if (head == NULL)
            tail = NULL;
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}

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

void hapusTengah(int posisi)
{
    if (isEmpty() || posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan atau list kosong" <<
endl;
    }
    else if (posisi == 1)
    {
        hapusDepan();
    }
    else if (posisi == hitungList())
    {
        hapusBelakang();
    }
    else

```

```

    {
        Node *bantu = head;
        for (int nomor = 1; nomor < posisi - 1; ++nomor)
        {
            bantu = bantu->next;
        }
        Node *hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

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

void ubahDepan(string nama, string nim)
{
    if (!isEmpty())
    {
        head->nama = nama;
        head->nim = nim;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void ubahTengah(string nama, string nim, int posisi)
{
    if (isEmpty() || posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan atau list kosong" <<
endl;
    }
    else
    {
        Node *bantu = head;
        for (int nomor = 1; nomor < posisi; ++nomor)
        {
            bantu = bantu->next;
        }
        bantu->nama = nama;
        bantu->nim = nim;
    }
}

```

```

}

void ubahBelakang(string nama, string nim)
{
    if (!isEmpty())
    {
        tail->nama = nama;
        tail->nim = nim;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void tampil()
{
    if (!isEmpty())
    {
        Node *current = head;
        cout << "\nDATA MAHASISWA\n\nNama\t\tNIM\n";
        while (current != NULL)
        {
            cout << current->nama << "\t\t" << current->nim <<
endl;
            current = current->next;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    while (true)
    {
        cout << "\nPROGRAM SINGLE LINKED LIST NON-
CIRCULAR\n\n";
        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;

        int choice;
        cout << "\npilih Operasi : ";
        cin >> choice;
    }
}

```



```

switch (choice)
{
case 1:
{
    string nama_2311102013;
    string nim_2311102013;
    cout << "\n<<< Tambah Depan >>>\n";
    cout << "Masukan Nama : ";
    cin >> nama_2311102013;
    cout << "Masukan Nim : ";
    cin >> nim_2311102013;
    insertDepan(nama_2311102013, nim_2311102013);
    cout << "\nData telah ditambahkan " << endl;
    break;
}
case 2:
{
    string nama_2311102013;
    string nim_2311102013;
    cout << "\n<<< Tambah Belakang >>>\n";
    cout << "Masukan Nama : ";
    cin >> nama_2311102013;
    cout << "Masukan Nim : ";
    cin >> nim_2311102013;
    insertBelakang(nama_2311102013, nim_2311102013);
    cout << "\nData telah ditambahkan " << endl;
    break;
}
case 3:
{
    string nama_2311102013;
    string nim_2311102013;
    int posisi_2311102013;
    cout << "\n<<< Tambah Tengah >>>\n";
    cout << "Masukan Nama : ";
    cin >> nama_2311102013;
    cout << "Masukan Nim : ";
    cin >> nim_2311102013;
    cout << "Masukan Posisi : ";
    cin >> posisi_2311102013;
    insertTengah(nama_2311102013, nim_2311102013,
posisi_2311102013);
    cout << "\nData telah ditambahkan " << endl;
    break;
}
case 4:
{
    string nama_2311102013;
    string nim_2311102013;
    cout << "\n<<< Ubah Depan >>>\n";
    cout << "Masukan Nama : ";
    cin >> nama_2311102013;
    cout << "Masukan Nim : ";
    cin >> nim_2311102013;
    ubahDepan(nama_2311102013, nim_2311102013);

```

```

        cout << "\nData telah diubah " << endl;
        ;
        break;
    }
    case 5:
    {
        string nama_2311102013;
        string nim_2311102013;
        cout << "\n<<< Ubah Belakang >>>\n";
        cout << "Masukan Nama : ";
        cin >> nama_2311102013;
        cout << "Masukan Nim : ";
        cin >> nim_2311102013;
        ubahBelakang(nama_2311102013, nim_2311102013);
        cout << "\nData telah diubah " << endl;
        break;
    }
    case 6:
    {
        string nama_2311102013;
        string nim_2311102013;
        int posisi_2311102013;
        cout << "\n<<< Ubah Tengah >>>\n";
        cout << "Masukan Nama : ";
        cin >> nama_2311102013;
        cout << "Masukan Nim : ";
        cin >> nim_2311102013;
        cout << "Masukan Posisi : ";
        cin >> posisi_2311102013;
        ubahTengah(nama_2311102013, nim_2311102013,
posisi_2311102013);
        cout << "\nData telah diubah " << endl;
        break;
    }
    case 7:
        hapusDepan();
        cout << "Data Berhasil dihapus " << endl;
        break;
    case 8:
        hapusBelakang();
        cout << "Data Berhasil dihapus " << endl;
        break;
    case 9:
    {
        int posisi_2311102013;
        cout << "<<< Hapus Tengah >>>\n";
        cout << "Masukkan posisi : ";
        cin >> posisi_2311102013;
        hapusTengah(posisi_2311102013);
        cout << "Data Berhasil dihapus " << endl;
        break;
    }
    case 10:
        clearList();
        break;

```

```

        case 11:
            tampil();
            break;
        case 0:
            return 0;
        default:
            cout << "Pilihan tidak valid!" << endl;
        }
    }
    return 0;
}

```

Screenshots Output:

Insert:

```

pilih Operasi : 1

<<< Tambah Depan >>>
Masukan Nama : Jawad
Masukan Nim   : 23300001

Data telah ditambahkan

```

Tampilkan semua:

```

pilih Operasi : 11

DATA MAHASISWA

Nama          NIM
Jawad         23300001
Radit         2311102013
Farrel        23300003
Denis         23300005
Anis          23300008
Bowo          23300015
Gahar         23300040
Udin          23300048
Ucok          23300050
Budi          23300099

```

- a. Tambahkan data berikut diantara Farrel dan Denis:

**Wati 2330004**

```
pilih Operasi : 3

<<< Tambah Tengah >>>
Masukan Nama   : Wati
Masukan Nim     : 2330004
Masukan Posisi  : 4

Data telah ditambahkan
```

```
pilih Operasi : 11

DATA MAHASISWA

Nama      NIM
Jawad     23300001
Radit     2311102013
Farrel    23300003
Wati      23300004
Denis     23300005
Anis      23300008
Bowo      23300015
Gahar     23300040
Udin      23300048
Ucok      23300050
Budi      23300099
```

- b. Hapus data Denis

```
pilih Operasi : 9
<<< Hapus Tengah >>>
Masukkan posisi : 5
Data Berhasil dihapus
```

DATA MAHASISWA	
Nama	NIM
Jawad	23300001
Radit	2311102013
Farrel	23300003
Wati	23300004
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

c. Tambah data berikut di awal:

Owi 2330000

```
pilih Operasi : 1

<<< Tambah Depan >>>
Masukan Nama : Owi
Masukan Nim  : 2330000

Data telah ditambahkan
```

DATA MAHASISWA	
Nama	NIM
Owi	2330000
Jawad	23300001
Radit	2311102013
Farrel	23300003
Wati	23300004
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

d. Tambahkan data berikut di akhir:

David 23300100

```
pilih Operasi : 2

<<< Tambah Belakang >>>
Masukan Nama : David
Masukan Nim  : 23300100

Data telah ditambahkan
```

DATA MAHASISWA	
Nama	NIM
Owi	2330000
Jawad	23300001
Radit	2311102013
Farrel	23300003
Wati	23300004
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099
David	23300100

- e. Ubah data Udin menjadi data berikut:

Idin 23300045

```
pilih Operasi : 6

<<< Ubah Tengah >>>
Masukan Nama   : Idin
Masukan Nim     : 23300045
Masukan Posisi  : 9

Data telah diubah
```

DATA MAHASISWA	
Nama	NIM
Owi	2330000
Jawad	23300001
Radit	2311102013
Farrel	23300003
Wati	23300004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099
David	23300100

- f. Ubah data terakhir menjadi berikut:

Lucy 23300101

```
pilih Operasi : 5

<<< Ubah Belakang >>>
Masukan Nama : Lucy
Masukan Nim   : 23300101

Data telah diubah
```

```
DATA MAHASISWA

Nama      NIM
Owi       2330000
Jawad     23300001
Radit     2311102013
Farrel    23300003
Wati      2330004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
Lucy      23300101
```

g. Hapus data awal

```
pilih Operasi : 7
Data Berhasil dihapus
```

```
DATA MAHASISWA

Nama      NIM
Jawad     23300001
Radit     2311102013
Farrel    23300003
Wati      2330004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
Lucy      23300101
```

- h. Ubah data awal menjadi Berikut

Bagas 2330002

```
pilih Operasi : 4

<<< Ubah Depan >>>
Masukan Nama : Bagas
Masukan Nim   : 2330002

Data telah diubah
```

```
DATA MAHASISWA

Nama           NIM
Bagas          2330002
Radit          2311102013
Farrel         23300003
Wati           23300004
Anis           23300008
Bowo           23300015
Gahar          23300040
Idin           23300045
Ucok           23300050
Budi           23300099
Lucy           23300101
```

- i. Hapus Data akhir

```
pilih Operasi : 8
Data Berhasil dihapus
```



#### DATA MAHASISWA

Nama	NIM
Bagas	2330002
Radit	2311102013
Farrel	23300003
Wati	2330004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099

- j. Tampilkan seluruh data

#### DATA MAHASISWA

Nama	NIM
Bagas	2330002
Radit	2311102013
Farrel	23300003
Wati	2330004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099

### Deskripsi:

Program di atas merupakan program menu yang mengimplementasikan dari linked list non-circular. Program akan mendeklarasikan Struct Node yang memiliki dua string ('nama' dan 'nim') sebagai data dan sebuah pointer 'next' yang menunjuk ke node berikutnya. Program akan mendeklarasi variable Global head dan tail yang menunjukkan node pertama dan terakhir. Kemudian program akan menginisialisasi head dan tail menjadi NULL. Program akan memeriksa apakah linked list kosong atau tidak.

Program juga memiliki fungsi untuk menambahkan elemen di depan, menambahkan elemen di tengah, menambahkan elemen di belakang, menghitung jumlah node, menghapus elemen pertama, menghapus elemen terakhir, menghapus elemen pada posisi tertentu, menghapus semua elemen, mengubah elemen pertama, mengubah elemen pada posisi tertentu, mengubah elemen terakhir, menampilkan semua elemen dalam linked list. Program juga menggunakan looping yang terus berjalan sehingga user dapat terus memilih operasi dalam menu.

## **BAB V**

### **KESIMPULAN**

Kesimpulan yang dapat saya ambil, Linked list non-circular memiliki node pertama (head) dan node terakhir (tail) yang tidak saling terhubung, di mana pointer terakhir (tail) selalu bernilai 'NULL'. Sebaliknya, linked list circular tidak memiliki akhir karena node terakhir (tail) terhubung dengan node pertama (head), sehingga tidak ada elemen yang menunjuk ke 'NULL'. Keduanya memiliki kegunaan yang berbeda, dengan non-circular cocok untuk data dengan awal dan akhir jelas, sedangkan circular cocok untuk penggunaan berulang.

## **BAB VI**

### **DAFTAR PUSTAKA**

[1] Asisten Praktikum. 2024. Modul 4 “Circular dan Non-Circular”. Diakses 13 April 2024, 15:00 WIB. <https://lms.ittelkom-pwt.ac.id/>