

**LAPORAN PRAKTIKUM STRUKTUR**

**DATA DAN ALGORITMA**

**MODUL IV**

**LINKED LIST CIRCULAR DAN NON CIRCULAR**



**Disusun oleh :**

**MUHAMAD NASRULLOH**

**2311102044**

**IF 11-B**

**Dosen :**

**WAHYU ANDI SAPUTRA, S.PD., M.ENG**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2024**

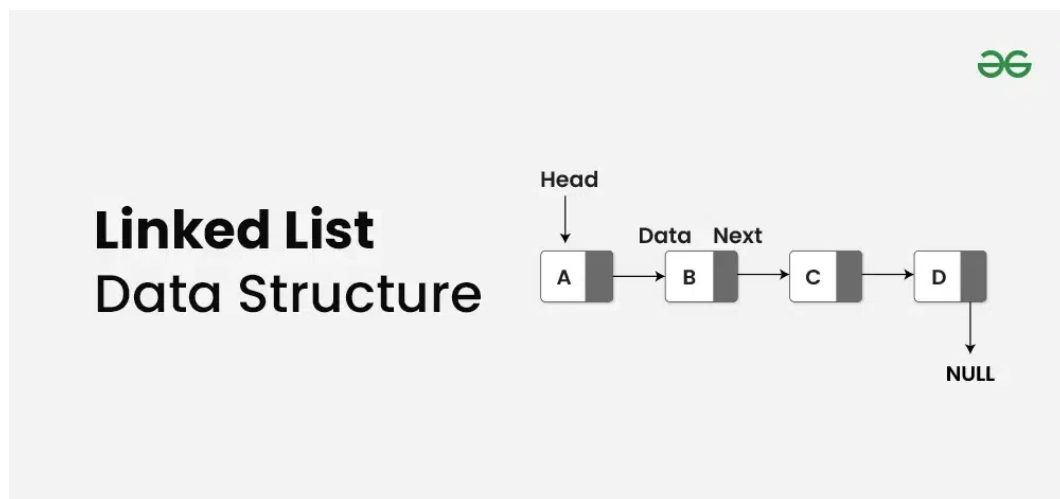
## A. DASAR TEORI

Linked List memiliki pengertian sekumpulan node yang saling terhubung dengan node lain melalui sebuah pointer. Diawali dengan sebuah head untuk menyimpan alamat awal dan diakhiri dengan node yang mengarah pointer ke null.

### 1. LINKED LIST NON CIRCULAR

Linked List yang field pointer nya hanya 1 buah dan 1 arah saja dan node-node didalamnya saling terhubung satu sama lain. Disebut juga sebagai linked list linear dimana simpul-simpul (node) tersusun secara linear atau berurutan dan simpul terakhir menunjuk ke nilai null/tidak menunjuk node apapun sehingga membentuk ujung dari struktur data ini.

Setiap node pada linked list mempunyai field yang berisi data dan pointer ke node berikutnya. Pada akhir linked list, node terakhir akan menunjuk ke NULL yang akan digunakan sebagai kondisi berhenti pada saat pembacaan isi linked list.



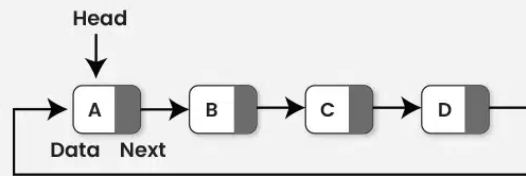
Gb 1.1 Ilustrasi Linked List Non Circular (Geeksforgeeks.org)

### 2. LINKED LIST CIRCULAR

Hampir sama dengan linked list non circular, bedanya node terakhirnya dihubungkan kembali ke node pertama(head) sehingga membentuk lingkaran/cincin. Ini berarti tidak ada node yang menunjuk ke "null" dan iterasi melalui linked list circular akan selalu berputar ke awal setelah mencapai node terakhir.



# Circular Linked List



**Gb 2.1** Ilustrasi Linked List Circular (geeksforgeeks.org)

## B. GUIDED

### I – Source Code

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

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

Node *head;
Node *tail;

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

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

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

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

int hitungList() {
    Node *hitung = head;
    int jumlah = 0;
```

```

        while (hitung != NULL) {
            jumlah++;
            hitung = hitung->next;
        }
        return jumlah;
    }

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

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

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

```

```

    }
}

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

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

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

```

```

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

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

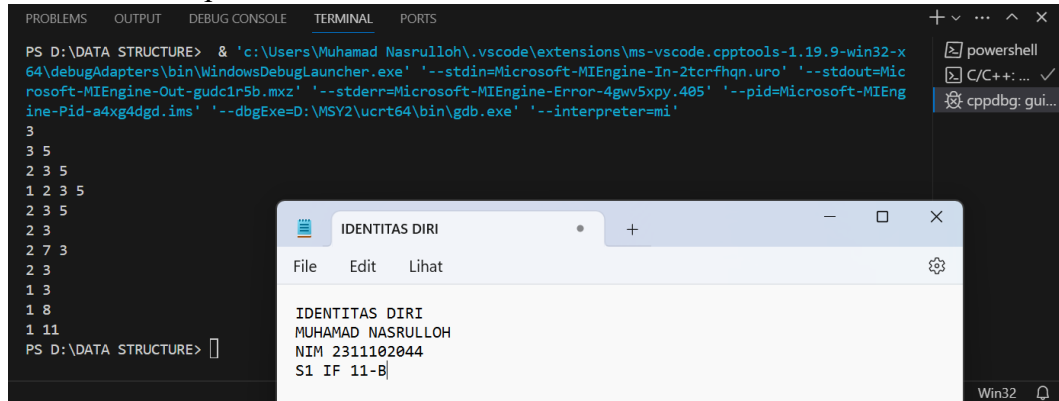
void tampil() {
    Node *bantu = head;
    if (!isEmpty()) {
        while (bantu != NULL) {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

int main() {
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
}

```

```
tampil();
return 0;
}
```

## Screenshot Output



## Deskripsi

Program diatas merupakan contoh implementasi Linked List Non Circular dimana setiap nodenya menyimpan 2 informasi berupa angka (interger) dan alamat memori node berikutnya. Didalam program ini ada 14 fungsi diantaranya : init (menginisialisasi nilai head dan tail dengan NULL/masih kosong), isEmpty (memeriksa apakah linked list tadi kosong), insertDepan (menambahkan node baru didepan linked list), insertBelakang (menambahkan node baru di belakang linked list), hitungList (menghitung jumlah node dan setiap node yang ditemui), insertTengah (menambahkan node baru pada posisi tertentu dalam linked list), hapusDepan (menghapus node pertama), hapusBelakang (menghapus node terakhir), hapusTengah (menghapus node pada posisi tertentu dalam linked list), ubahDepan (mengubah nilai data node pertama), ubahTengah (mengubah nilai data pada posisi tertentu dalam linked list), ubahBelakang (mengubah nilai data node terakhir), clearList (menghapus semua node dalam linked list), tampil (menampilkan data setiap node dalam linked list) yang kesemuanya ini berada dalam fungsi main() untuk menjalankan operasi tertentu.

## II – Source Code

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

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

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

void init() {
    head = NULL;
```



```

        tail = head;
    }

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

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

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

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

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

    void insertTengah(string data, int posisi) {
        if (isEmpty()) {

```

```

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

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

void hapusBelakang() {
    if (!isEmpty()) {
        hapus = head;
        tail = head;
        if (hapus->next == head) {
            head = NULL;
            tail = NULL;
            delete hapus;
        } else {
            while (hapus->next != head) {
                hapus = hapus->next;
            }
            while (tail->next != hapus) {
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    }
}

```

```

    }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

void hapusTengah(int posisi) {
    if (!isEmpty()) {
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

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

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

int main() {
    init();
    insertDepan("Ayam");
    tampil();
    insertDepan("Bebek");
    tampil();
    insertBelakang("Cicak");
}

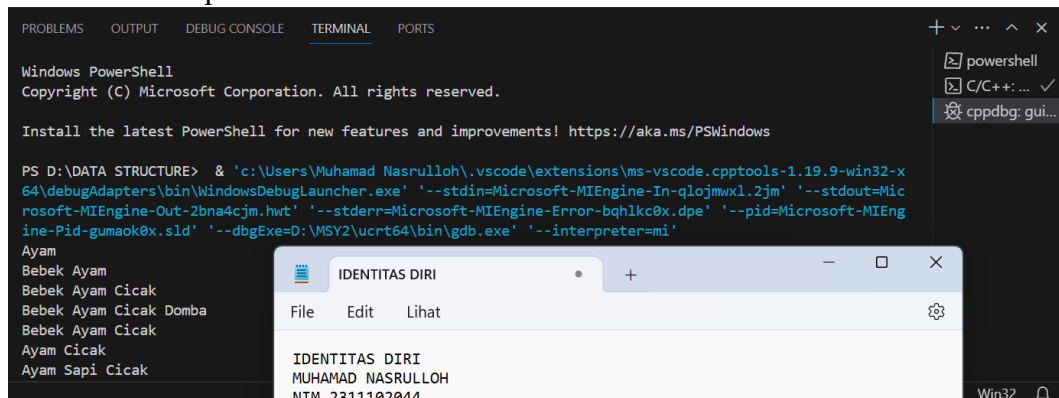
```

```

    tampil();
    insertBelakang("Domba");
    tampil();
    hapusBelakang();
    tampil();
    hapusDepan();
    tampil();
    insertTengah("Sapi", 2);
    tampil();
    hapusTengah(2);
    tampil();
    return 0;
}

```

## Screenshot Output



## Deskripsi

Program diatas merupakan contoh implementasi Circular Linked List. Disini ada 12 fungsi. Tetapi ada sedikit perbedaan dibandingkan Guided 1 sebelumnya yaitu tidak adanya ubah data (update) baik depan, tengah maupun belakang. Kemudian pada looping, jika data sekarang (pointer) nya tidak sama dengan head maka looping akan berjalan terus dan jika sama dengan head maka looping akan berhenti, karena pointer sudah kembali ke titik awal (node tail selalu menunjuk ke head). Inilah yang membedakan dengan Linked List Non Circular pada Guided 1 sebelumnya.

### C. UNGUIDED

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

#### Source Code

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

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

string tmp_nama;
Node *head;
Node *tail;

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

void insertTengah(string nama, string nim, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru = new Node();
        baru->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;
        tmp_nama = head->nama;
        if (head->next != NULL)

```

```

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

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

void hapusTengah(int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
}

```

```

else if (posisi == 1)
{
    cout << "Posisi bukan posisi tengah" << endl;
}
else
{
    Node *bantu = head;
    Node *hapus;
    Node *sebelum = NULL;
    int nomor = 1;

    while (nomor < posisi)
    {
        sebelum = bantu;
        bantu = bantu->next;
        nomor++;
    }
    hapus = bantu;
    string tmp_nama = bantu->nama;
    if (sebelum != NULL)
    {
        sebelum->next = bantu->next;
    }
    else
    {
        head = bantu->next;
    }
    delete hapus;
    cout << "Data " << tmp_nama << " telah dihapus." <<
endl;
}
}

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

void ubahTengah(string nama, string nim, int posisi)
{
    if (!isEmpty())
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;

```



```

    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi)
        {
            bantu = bantu->next;
            nomor++;
        }
        tmp_nama = bantu->nama;
        bantu->nama = nama;
        bantu->nim = nim;
        cout << "Data " << tmp_nama << " telah diganti
dengan data " << nama << endl;
    }
}
else
{
    cout << "List masih kosong!" << endl;
}
}

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

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

```

```

void tampil()
{
    Node *bantu = head;
    if (!isEmpty())
    {
        cout << "\nDATA MAHASISWA\n\n";
        cout << setw(37) << setfill('-') << "-" << setfill(' ')
<< endl;
        cout << "| " << setw(20) << left << "Nama"
            << " | " << setw(10) << "NIM"
            << " |" << endl;
        cout << setw(37) << setfill('-') << "-" << setfill(' ')
<< endl;
        while (bantu != NULL)
        {
            cout << "| " << setw(20) << left << bantu->nama << "
| " << setw(10) << bantu->nim << " |" << endl;
            bantu = bantu->next;
        }
        cout << setw(37) << setfill('-') << "-" << setfill(' ')
<< endl;
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    int operasi, posisi;
    string nama;
    string nim;

    cout << endl;
    cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << endl;
    cout << "1. Tambah Data di Depan" << endl;
    cout << "2. Tambah Data di Belakang" << endl;
    cout << "3. Tambah Data di Tengah" << endl;
    cout << "4. Ubah Data di Depan" << endl;
    cout << "5. Ubah Data di Belakang" << endl;
    cout << "6. Ubah Data di Tengah" << endl;
    cout << "7. Hapus Data di Depan" << endl;
    cout << "8. Hapus Data di Belakang" << endl;
    cout << "9. Hapus Data di Tengah" << endl;
    cout << "10. Hapus Seluruh Data" << endl;
    cout << "11. Tampilkan Data" << endl;
    cout << "0. Keluar" << endl;
    cout << endl;
    do
    {
        cout << "Pilih Operasi: ";
        cin >> operasi;
        switch (operasi)

```

```

{
case 1:
    cout << "=== Tambah Depan ===" << endl;
    cout << "Masukkan Nama : ";
    cin.ignore();
    getline(cin, nama);
    cout << "Masukkan NIM : ";
    cin >> nim;
    insertDepan(nama, nim);
    cout << endl
        << "Data telah ditambahkan." << endl
        << endl;
    break;
case 2:
    cout << "=== Tambah Belakang ===" << endl;
    cout << "Masukkan Nama : ";
    cin.ignore();
    getline(cin, nama);
    cout << "Masukkan NIM : ";
    cin >> nim;
    insertBelakang(nama, nim);
    cout << endl
        << "Data telah ditambahkan." << endl
        << endl;
    break;
case 3:
    cout << "=== Tambah Tengah ===" << endl;
    cout << "Masukkan Nama : ";
    cin.ignore();
    getline(cin, nama);
    cout << "Masukkan NIM : ";
    cin >> nim;
    cout << "Masukkan Posisi : ";
    cin >> posisi;
    insertTengah(nama, nim, posisi);
    cout << endl
        << "Data telah ditambahkan." << endl
        << endl;
    break;
case 4:
    cout << "=== Ubah Depan ===" << endl;
    cout << "Masukkan Nama : ";
    cin.ignore();
    getline(cin, nama);
    cout << "Masukkan NIM : ";
    cin >> nim;
    ubahDepan(nama, nim);
    break;
case 5:
    cout << "=== Ubah Belakang ===" << endl;
    cout << "Masukkan Nama : ";
    cin.ignore();
    getline(cin, nama);
    cout << "Masukkan NIM : ";
    cin >> nim;
    ubahBelakang(nama, nim);

```

```

        cout << endl;
        break;
    case 6:
        cout << "=== Ubah Tengah ===" << endl;
        cout << "Masukkan Nama : ";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        ubahTengah(nama, nim, posisi);
        cout << endl;
        break;
    case 7:
        cout << endl
            << "=== Hapus Depan ===" << endl;
        hapusDepan();
        cout << endl;
        break;
    case 8:
        cout << "=== Hapus Belakang ===" << endl;
        hapusBelakang();
        break;
    case 9:
        cout << "=== Hapus Tengah ===" << endl;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        hapusTengah(posisi);
        cout << endl;
        break;
    case 10:
        cout << "=== Hapus List ===" << endl;
        clearList();
        break;
    case 11:
        tampil();
        break;

    default:
        break;
}
} while (operasi != 0);
}

```

**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 :

-Hapus Belakang

Data (nama mahasiswa yang dihapus) berhasil dihapus

-Hapus Tengah

Masukkan posisi :

Data (nama mahasiswa yang dihapus) berhasil dihapus

• Tampilan Operasi Tambah:

-Tambah Depan

Masukkan Nama :

Masukkan NIM :

Data telah ditambahkan

• Tampilan Operasi Ubah:

-Ubah Belakang

Masukkan nama :

Masukkan NIM :

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

-Tambah Tengah

Masukkan Nama :

Masukkan NIM :

Masukkan Posisi :

Data telah ditambahkan

-Ubah Belakang

Masukkan nama :

Masukkan NIM :

Masukkan posisi :

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

• Tampilan Operasi Tampil Data:

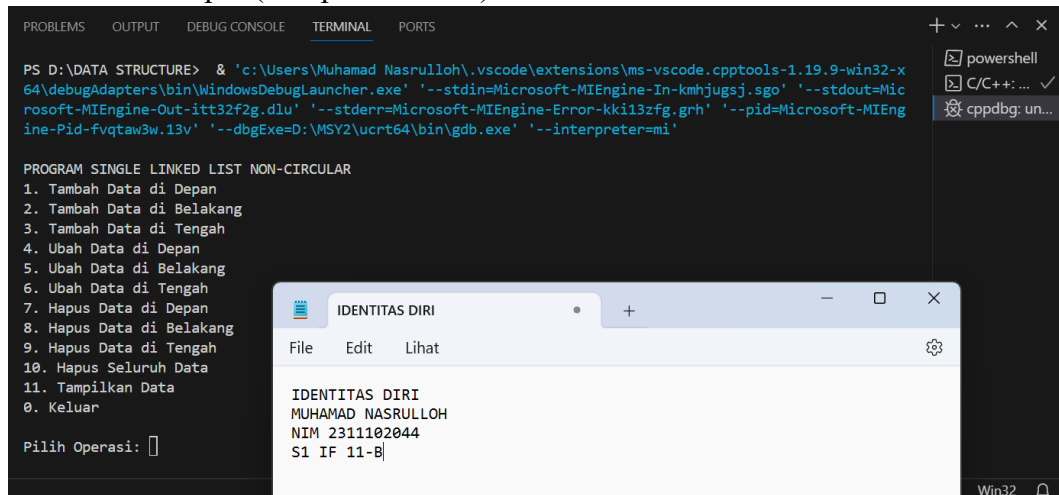
DATA MAHASISWA

NAMA NIM

Nama1 NIM1

Nama2 NIM2


## Screenshot Output (Tampilkan menu)



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
[Nama Anda]	[NIM Anda]
Farrel	23300003
Denis	23300005
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

## Screenshot Output (Menampilkan Urutan Data)



```
Pilih Operasi: 2
=== Tambah Belakang ===
Masukkan Nama : Jawad
Masukkan NIM : 23300008

Data telah ditambahkan.

Pilih Operasi: 2
=== Tambah Belakang ===
Masukkan Nama : Muhamad Nasrulloh
Masukkan NIM : 2311102044

Data telah ditambahkan.

Pilih Operasi: 2
=== Tambah Belakang ===
Masukkan Nama : Farrel
Masukkan NIM : 23300003

Data telah ditambahkan.

Pilih Operasi: 2
=== Tambah Belakang ===
Masukkan Nama : Denis
Masukkan NIM : 23300005

Data telah ditambahkan.

Pilih Operasi: 2
=== Tambah Belakang ===
Masukkan Nama : Anis
Masukkan NIM : 23300008

Data telah ditambahkan.

Pilih Operasi: 2
=== Tambah Belakang ===
Masukkan Nama : Bowo
Masukkan NIM : 23300015

Data telah ditambahkan.

Pilih Operasi: 2
=== Tambah Belakang ===
Masukkan Nama : Gahar
Masukkan NIM : 23300040

Data telah ditambahkan.

Pilih Operasi: 2
=== Tambah Belakang ===
Masukkan Nama : Udin
Masukkan NIM : 23300048

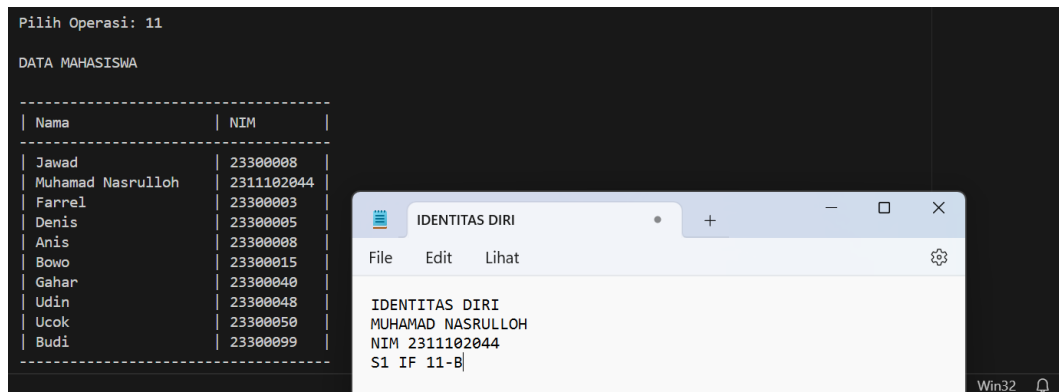
Data telah ditambahkan.

Pilih Operasi: 2
=== Tambah Belakang ===
Masukkan Nama : Ucok
Masukkan NIM : 23300050

Data telah ditambahkan.

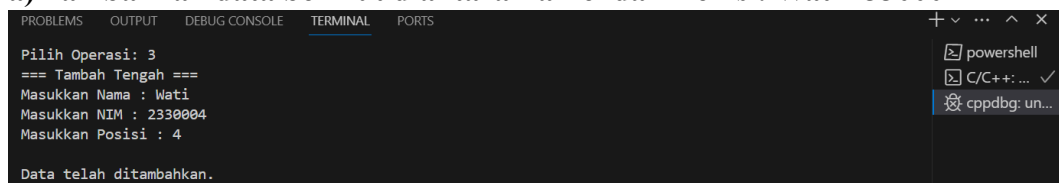
Pilih Operasi: 2
=== Tambah Belakang ===
Masukkan Nama : Budi
Masukkan NIM : 23300099

Data telah ditambahkan.
```



3. Lakukan perintah berikut :

a) Tambahkan data berikut diantara Farrel dan Denis : Wati 2330004



b) Hapus data Denis



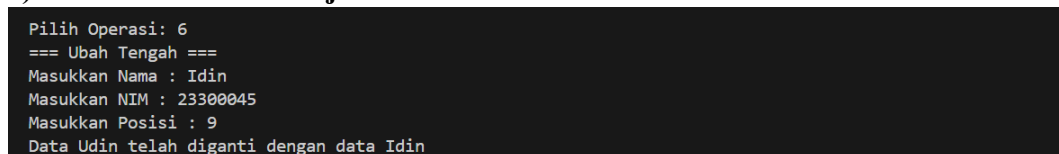
c) Tambahkan data berikut di awal : Owi 2330000



d) Tambahkan data berikut di akhir : David 23300100



e) Ubah data Udin menjadi data berikut : Idin 23300045



f) Ubah data terakhir menjadi berikut : Lucy 23300101



g) Hapus data awal





#### h) Ubah data awal menjadi berikut : Bagas 2330002

```
Pilih Operasi: 4
=== Ubah Depan ===
Masukkan Nama : Bagas
Masukkan NIM : 2330002
Data Jawad telah diganti dengan data Bagas
```

#### i) Hapus data akhir

```
Pilih Operasi: 8
=== Hapus Belakang ===
Data Lucy telah dihapus.
```

#### j) Tampilkan seluruh data

```
Pilih Operasi: 11

DATA MAHASISWA

-----
| Nama          | NIM          |
-----
| Bagas         | 2330002      |
| Muhamad Nasrulloh | 2311102044  |
| Farrel        | 23300003     |
| Wati          | 23300004     |
| Anis          | 23300008     |
| Bowo          | 23300015     |
| Gahar         | 23300040     |
| Idin          | 23300045     |
| Ucok          | 23300050     |
| Budi          | 23300099     |
-----

Pilih Operasi: 0
PS D:\DATA STRUCTURE>
```

IDENTITAS DIRI  
MUHAMAD NASRULLOH  
NIM 2311102044  
S1 IF 11-B

#### Deskripsi

Program diatas merupakan program hasil modifikasi dari Guided 1, jadi sama-sama menggunakan Linked List Non Circular, tetapi pada program ini dapat menyimpan 2 data : nama dan NIM mahasiswa. Yang menjadi pembeda ada di fungsi ubahDepan, ubahTengah, ubahBelakang dimana ketika fungsi ini dijalankan akan menampilkan output berupa nama mahasiswa sebelum dan setelah diubah. Begitu juga pada fungsi hapusDepan, hapusTengah, hapusBelakang yang menampilkan output nama mahasiswa yang telah dihapus datanya. Program akan menampilkan menu 1x sesuai pilihan dari user, dan “pilih operasi” akan terus ditampilkan sampai user memilih opsi 0. Keluar sehingga program baru berhenti.

#### **D. KESIMPULAN**

Berdasar dari yang telah saya ketahui, untuk kode program baik Linked List Non Circular dengan Linked List Circular itu hampir sama. Perbedaannya ada pada penunjuk akhir, iterasi, dan mungkin implementasi (kegunaannya). Untuk penunjuk akhir, pada Linked List Non Circular (LLNC) menunjuk ke NULL sedangkan Linked List Circular (LLC) menunjuk ke node pertama (head). Iterasinya, LLNC berhenti ketika mencapai NULL sedangkan LLC akan terus berputar melalui node-nodenya. Dan implementasi, LLNC lebih umum untuk struktur data yang sederhana sedangkan LLC untuk implementasi yang memerlukan akses berulang.

#### **E. REFERENSI**

[1] Prepinsta, Linked List in C++.

<https://prepinsta.com/cpp-program/linked-list/>. Diakses pada 16 April 2024

[2] GeeksforGeeks (2024), Linked List Data Structure.

<https://www.geeksforgeeks.org/data-structures/linked-list/>. Diakses pada 16 April 2024

[3] GeeksforGeeks (2024), Introduction to Circular Linked List.

<https://www.geeksforgeeks.org/circular-linked-list/>. Diakses pada 16 April 2024

[4] Nadia Apriliani Hamidah (2020), Membedakan Single Linked List Circular dan Non Circular.

<https://nadiaaprilianihamidah04.blogspot.com/2020/05/membedakan-single-linked-list-circular.html>. Diakses pada 16 April 2024