

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL IV  
LINKED LIST CIRCULAR DAN NON CIRCULAR**



**Disusun Oleh :**

NAMA : GALIH TRISNA  
NIM : 2311102050

**Dosen**

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

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFROMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## A. Dasar Teori

### a. Linked List Non Circular

Linked list non circular adalah struktur data dimana setiap node terdiri dari dua bagian: data itu sendiri dan pointer yang menunjuk ke node berikutnya dalam urutan. Dalam linked list non circular, node pertama (head) dan node terakhir (tail) tidak saling terhubung. Ini berarti bahwa pointer pada node terakhir (tail) selalu menunjuk ke 'NULL', menandakan akhir dari linked list. Konsep ini membedakan linked list non circular dari yang circular, yang mana node terakhirnya menunjuk kembali ke node pertama. Dengan demikian, gambaran visual dari linked list non circular adalah seperti deretan node yang terhubung satu arah, dimana setiap node memiliki pointer yang menunjuk ke node berikutnya kecuali node terakhir yang menunjuk ke 'NULL'.



### b. Linked List Circular

Linked list circular adalah jenis struktur data dimana setiap node terhubung dengan node berikutnya dalam urutan, dengan node terakhir (tail) menunjuk kembali ke node pertama (head), membentuk sebuah lingkaran. Berbeda dengan linked list non-circular, pada linked list circular, node terakhir tidak memiliki nilai 'NULL'. Untuk menghentikan iterasi atau perulangan, seringkali diperlukan penggunaan dummy node atau node penengah yang disebut sebagai node current. Ini memungkinkan program untuk berhenti menghitung data ketika node current mencapai node pertama (head). Linked list circular umumnya digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu, antrian pesan, atau dalam kasus penggunaan memori berulang dalam suatu aplikasi. Dalam gambaran visual, linked list circular terlihat seperti lingkaran dimana setiap node memiliki pointer yang menunjuk ke node berikutnya, kecuali node terakhir yang mengarah kembali ke node pertama.



## B. Guided

### Guided 1

```
#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;
    }
}

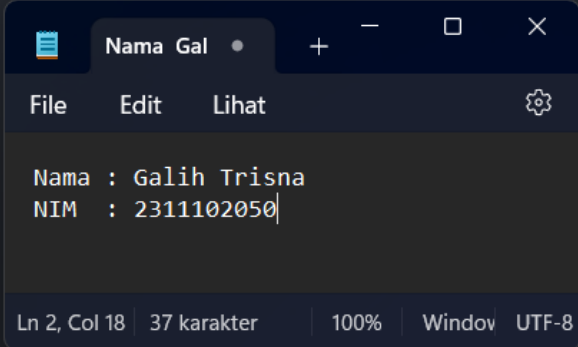
```



```
}  
  
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;  
}
```

Screenshots Output

```
PS C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\030424> g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
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:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\030424>
```



Deskripsi: Program di atas adalah sebuah program linked list yang berisi bilangan bulat. Program ini menambahkan data di depan, belakang, atau di tengah linked list, menghapus data di depan, belakang, atau di tengah linked list, mengubah nilai data di depan, belakang, atau di tengah linked list, menampilkan isi linked list, dan menghapus semua data linked list. Data pada linked list ini terdiri dari sebuah struct Node yang berisi int data untuk menyimpan data bilangan bulat dan Node \*next untuk menunjukkan ke data selanjutnya dalam linked list.

Dalam program ini terdapat beberapa fungsi yaitu:

1. `init()`: Fungsi ini menginisialisasi pointer head dan tail ke NULL, menandakan bahwa linked list masih kosong.
2. `isEmpty()`: Fungsi ini memeriksa apakah linked list kosong. Jika head bernilai NULL, maka linked list kosong dan fungsi akan mengembalikan true, jika tidak, maka linked list tidak kosong dan fungsi akan mengembalikan false.
3. `insertDepan(int nilai)`: Fungsi ini menambahkan sebuah node baru di depan linked list. Jika linked list kosong, maka node baru akan menjadi head dan tail. Namun jika tidak, node baru akan ditambahkan di depan head, dan head diupdate untuk menunjuk ke node baru.
4. `insertBelakang(int nilai)`: Fungsi ini menambahkan sebuah node baru di belakang linked list. Jika linked list kosong, maka node baru akan menjadi head dan tail. Namun jika tidak, node baru akan ditambahkan setelah tail, dan tail diupdate untuk menunjuk ke node baru.
5. `hitungList()`: Fungsi ini menghitung jumlah node di dalam linked list dengan melakukan iterasi dari head sampai tail dan menghitung setiap node yang ditemui.
6. `insertTengah(int data, int posisi)`: Fungsi ini menambahkan node baru pada posisi tertentu di dalam linked list. Fungsi ini memeriksa

apakah posisi yang dimasukkan valid, kemudian melakukan iterasi hingga posisi yang ditentukan dan menyisipkan node baru di antara node sebelum dan sesudah posisi yang ditentukan.

7. hapusDepan(): Fungsi ini menghapus node pertama dalam linked list. Jika linked list tidak kosong, head akan diupdate untuk menunjuk ke node berikutnya, dan node pertama akan dihapus dari memori.
8. hapusBelakang(): Fungsi ini menghapus node terakhir dalam linked list. Jika linked list tidak kosong, tail akan diupdate untuk menunjuk ke node sebelumnya, dan node terakhir akan dihapus dari memori.
9. hapusTengah(int posisi): Fungsi ini menghapus node pada posisi tertentu dalam linked list. Fungsi ini memeriksa apakah posisi yang dimasukkan valid, kemudian melakukan iterasi hingga posisi yang ditentukan dan menghapus node dari memori.
10. ubahDepan(int data): Fungsi ini mengubah nilai data node pertama dalam linked list.
11. ubahTengah(int data, int posisi): Fungsi ini mengubah nilai data pada posisi tertentu dalam linked list.
12. ubahBelakang(int data): Fungsi ini mengubah nilai data node terakhir dalam linked list.
13. clearList(): Fungsi ini menghapus semua node dalam linked list dengan melakukan iterasi dari head hingga tail, dan menghapus setiap node dari memori.
14. tampil(): Fungsi ini menampilkan nilai data dari setiap node dalam linked list dengan melakukan iterasi dari head hingga tail, dan menampilkan nilai data setiap node.

Semua fungsi di atas digunakan di dalam fungsi main() untuk menjalankan berbagai perintah pada linked list seperti menambahkan, menghapus, mengubah, dan menampilkan data-data dalam linked list.

## Guided 2

```
#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;
}

```

### Screenshots Output

```

PS C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\030424>
g++ guided2.cpp -o guided2 } ; if ($?) { .\guided2 }
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak
PS C:\Users\galih\OneDrive\Dokume

```

Overlay window content:

```

Nama : Galih Trisna
NIM  : 2311102050

```

Footer: Ln 2, Col 18 | 37 karakter | 100% | Window UTF-8

Deskripsi: program diatas adalah sebuah contoh implementasi dari circular linked list. Circular linked list ini memiliki beberapa fungsi seperti isEmpty, buatNode, init, hitungList, insertDepan, insertBelakang, insertTengah, hapusDepan, hapusBelakang, hapusTengah, clearList, dan tampil. Program ini mirip dengan linked list versi non-circular pada contoh sebelumnya, hanya saja pada circular linked list, pointer next di tail mengarah kembali kepada head lagi. Artinya, jika kita ingin menampilkan data dari node paling awal sampai akhir, node paling akhir tidak lagi memiliki nilai null pada pointer next di



variabel tail. Sebaliknya, nilai head pada pointer next di tail yang menandakan bahwa linked list ini membentuk suatu lingkaran atau loop.

### C. Unguided/Tugas

#### Unguided 1

```
#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
         << "===== " <<
endl;
    cout << "  PROGRAM SINGLE LINKED LIST NON-CIRCULAR  " <<
endl;
    cout << "===== " <<
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 << endl
                     << "=== Tambah Depan ===" << endl
                     << endl;
                cout << "Masukkan Nama : ";
                cin.ignore();
                getline(cin, nama);
                cout << "Masukkan NIM : ";
                cin >> nim;
                insertDepan(nama, nim);
                cout << endl

```

```

        << endl
        << "Data telah ditambahkan." << endl
        << endl;
    break;
case 2:
    cout << endl
        << "=== Tambah Belakang ===" << endl
        << endl;
    cout << "Masukkan Nama : ";
    cin.ignore();
    getline(cin, nama);
    cout << "Masukkan NIM : ";
    cin >> nim;
    insertBelakang(nama, nim);
    cout << endl
        << endl
        << "Data telah ditambahkan." << endl
        << endl;
    break;
case 3:
    cout << endl
        << "=== Tambah Tengah ===" << endl
        << 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
        << endl
        << "Data telah ditambahkan." << endl
        << endl;
    break;
case 4:
    cout << endl
        << "=== Ubah Depan ===" << endl
        << endl;
    cout << "Masukkan Nama : ";
    cin.ignore();
    getline(cin, nama);
    cout << "Masukkan NIM : ";

```

```

        cin >> nim;
        ubahDepan(nama, nim);
        break;
    case 5:
        cout << endl
            << "=== Ubah Belakang ===" << endl
            << endl;
        cout << "Masukkan Nama : ";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan NIM : ";
        cin >> nim;
        ubahBelakang(nama, nim);
        break;
    case 6:
        cout << endl
            << "=== Ubah Tengah ===" << endl
            << endl;
        cout << "Masukkan Nama : ";
        cin.ignore();
        getline(cin, nama);
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        ubahTengah(nama, nim, posisi);
        break;
    case 7:
        cout << endl
            << "=== Hapus Depan ===" << endl
            << endl;
        hapusDepan();
        break;
    case 8:
        cout << endl
            << "=== Hapus Belakang ===" << endl
            << endl;
        hapusBelakang();
        break;
    case 9:
        cout << endl
            << "=== Hapus Tengah ===" << endl
            << endl;
        cout << "Masukkan Posisi : ";

```

```

        cin >> posisi;
        hapusTengah(posisi);
        break;
    case 10:
        cout << endl
              << "=== Hapus List ===" << endl
              << endl;
        clearList();
        break;
    case 11:
        tampil();
        break;

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

```

## Screenshots Output

- Menu

```

PS C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\030424> cd "c:
g++ unguided1.cpp -o unguided1 } ; if ($?) { .\unguided1 }

=====
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
=====
1. Tambah Data di Depan
2. Tambah Data di Belakang
3. Tambah Data di Tengah
4. Ubah Data di Depan
5. Ubah Data di Belakang
6. Ubah Data di Tengah
7. Hapus Data di Depan
8. Hapus Data di Belakang
9. Hapus Data di Tengah
10. Hapus Seluruh Data
11. Tampilkan Data
0. Keluar
=====
Pilih Operasi: 

```

Nama : Galih Trisna  
 NIM : 2311102050

Ln 2, Col 18 | 37 karakter | 100% | Window UTF-8

- Masukkan data sesuai urutan berikut

=====

Pilih Operasi: 2

=== Tambah Belakang ===

Masukkan Nama : Jawad

Masukkan NIM : 23300001

Data telah ditambahkan.

Pilih Operasi: 2

=== Tambah Belakang ===

Masukkan Nama : Galih Trisna

Masukkan NIM : 2311102050

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.

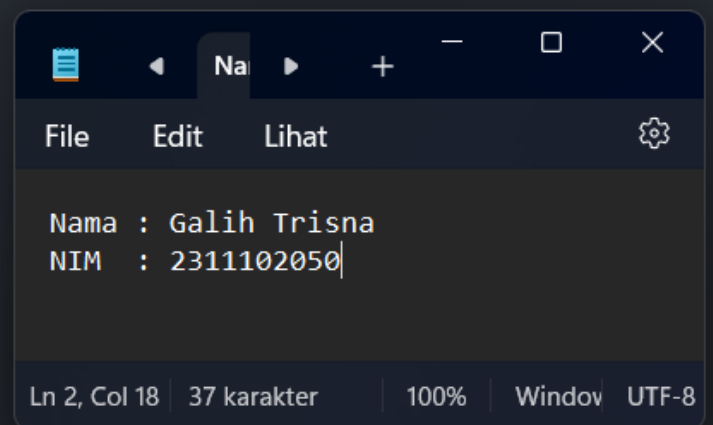
Pilih Operasi: 2

=== 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

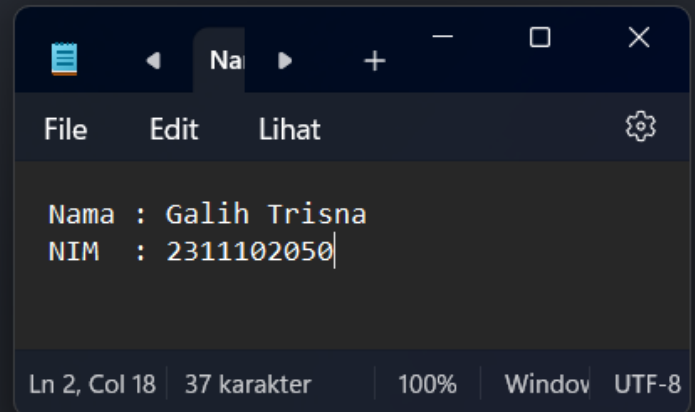
Masukkan NIM : 23300099

Data telah ditambahkan.

Pilih Operasi: 11

DATA MAHASISWA

-----	
Nama	NIM
-----	
Jawad	23300001
Galih Trisna	2311102050
Farrel	23300003
Denis	23300005
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099
-----	



- Tambahkan data berikut diantara Farrel dan Denis: Wati 2330004

```
Pilih Operasi: 3

=== Tambah Tengah ===

Masukkan Nama : Wati
Masukkan NIM : 2330004
Masukkan Posisi : 4

Data telah ditambahkan.

Pilih Operasi: 11
```

Na
 File Edit Lihat
 Nama : Galih Trisna
 NIM : 2311102050
 Ln 2, Col 18 | 37 karakter | 100% | Window UTF-8

- Hapus data Denis

```
Pilih Operasi: 9

=== Hapus Tengah ===

Masukkan Posisi : 5
Data Denis telah dihapus.
Pilih Operasi: 
```

Na
 File Edit Lihat
 Nama : Galih Trisna
 NIM : 2311102050

- Tambahkan data berikut di awal: Owi 2330000

```
Pilih Operasi: 1

=== Tambah Depan ===

Masukkan Nama : Owi
Masukkan NIM : 2330000

Data telah ditambahkan.
```

Na
 File Edit Lihat
 Nama : Galih Trisna
 NIM : 2311102050

- Tambahkan data berikut di akhir: David 23300100

```
Pilih Operasi: 2

=== Tambah Belakang ===

Masukkan Nama : David
Masukkan NIM : 23300100

Data telah ditambahkan.
```

File	Edit	Lihat
Nama : Galih Trisna NIM : 2311102050		

- Ubah data Udin menjadi data berikut: Idin 23300045

```
Pilih Operasi: 6

=== Ubah Tengah ===

Masukkan Nama : Idin
Masukkan NIM : 23300045
Masukkan Posisi : 9
Data Udin telah diganti dengan data Idin
Pilih Operasi: 
```

File	Edit	Lihat
Nama : Galih Trisna NIM : 2311102050		

- Ubah data terkahir menjadi berikut: Lucy 23300101

```
=== Ubah Belakang ===

Masukkan Nama : Lucy
Masukkan NIM : 23300101
Data David telah diganti dengan data Lucy
Pilih Operasi: 
```

File	Edit	Lihat
Nama : Galih Trisna NIM : 2311102050		

- Hapus data awal

```
=== Hapus Depan ===

Data Owi telah dihapus.
Pilih Operasi: 
```

File	Edit	Lihat
Nama : Galih Trisna NIM : 2311102050		



- 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
Pilih Operasi: 
```

File Edit Lihat

Nama : Galih Trisna  
NIM : 2311102050

- Hapus data akhir

```
Pilih Operasi: 8

=== Hapus Belakang ===

Data Lucy telah dihapus.
Pilih Operasi: 
```

File Edit Lihat

Nama : Galih Trisna  
NIM : 2311102050

- Tampilkan seluruh data

```
Pilih Operasi: 11

DATA MAHASISWA

-----
| Nama          | NIM          |
-----
| Bagas         | 2330002      |
| Galih Trisna  | 2311102050   |
| Farrel        | 23300003     |
| Wati          | 23300004     |
| Anis          | 23300008     |
| Bowo          | 23300015     |
| Gahar         | 23300040     |
| Idin          | 23300045     |
| Ucok          | 23300050     |
| Budi          | 23300099     |
-----

Pilih Operasi: 
```

File Edit Lihat

Nama : Galih Trisna  
NIM : 2311102050

Ln 2, Col 18 | 37 karakter | 100%

- Deskripsi

Program di atas merupakan hasil modifikasi dari program Guided 1. Sama seperti program Guided 1, program ini menggunakan single linked list non-circular, namun program ini dapat menyimpan dua data, yaitu nama dan NIM dari mahasiswa. Secara keseluruhan program ini hampir sama dengan Guided 1, namun ada beberapa perbedaan dalam program kali ini. Perbedaan yang terdapat pada fungsi ubahDepan(), ubahTengah() ubahBelakang() karena pada setiap fungsi dijalankan maka akan menampilkan nama mahasiswa sebelum diubah dan setelah diubah. Kemudian terdapat perbedaan juga pada fungsi hapusDepan() hapusBelakang() hapusTengah() yang Ketika dijalankan akan menampilkan nama mahasiswa yang datanya telah dihapus. Perbedaan juga dapat ditemukan pada saat program dijalankan. Pada program ini hanya menampilkan menu sebanyak 1 kali. Namun untuk pilih menu (pilih operasi) tetap terus tampil selama pengguna tidak memilih menu keluar.

#### D. Kesimpulan

Linked list non circular adalah linked list yang pointer next node terakhirnya mengarah kepada null. Sedangkan untuk double linked list, pointer next pada node terakhir dan prev pada node paling awal akan mengarah ke null. Namun berbeda lagi dengan circular, pada single linked list circular, pointer next pada node terakhir mengarah ke head. Sedangkan untuk double linked list, pointer next pada node terakhir mengarah ke head atau node paling awal dan prev pada node paling awal mengarah ke tail atau node paling terakhir.

#### E. Referensi

Mikirin Kode. (n.d.). Single Linked List. Diakses pada 8 April 2024, dari <https://mikirinkode.com/single-linked-list/>

Asisten Praktikum. (2024). Modul IV : Linked List Circular dan Non Circular

GfG. Linked List data structure. GeeksforGeeks. Diakses pada 8 april 2024, dari <https://www.geeksforgeeks.org/data-structures/linked-list>

Programiz. (n.d.). Circular Linked List. Diakses pada 8 april 2024, dari <https://www.programiz.com/dsa/circular-linked-list>