

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

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



Disusun Oleh :

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

Dosen

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

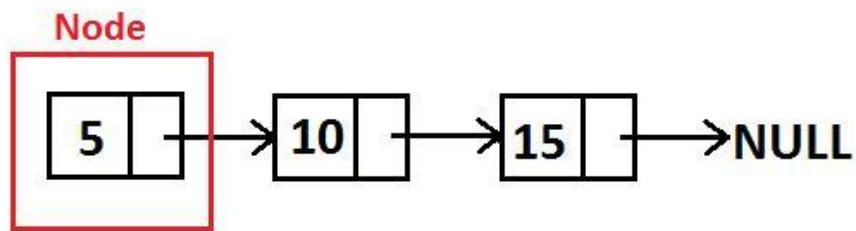
**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. Dasar Teori

a. Linked List Non Circular

Linked List Non Circular adalah linear dari data, yang disebut sebagai nodes, dimana setiap node akan menunjuk pada node lain melalui sebuah pointer. Linked List dapat didefinisikan pula sebagai kumpulan nodes yang merepresentasikan sebuah sequence.

Representasi sebuah linked list Non Circular dapat digambarkan melalui gambar di bawah ini:

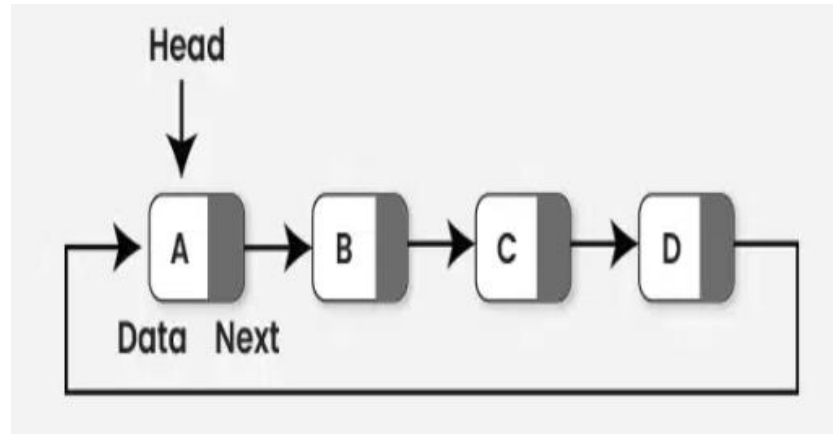


Gambar 1 Linked List Non Circular

Sebuah linked list yang hanya memiliki 1 penghubung ke node lain disebut sebagai single linked list. Di dalam sebuah linked list, ada 1 pointer yang menjadi gambaran besar, yakni pointer HEAD yang menunjuk pada node pertama di dalam linked list itu sendiri. Sebuah linked list dikatakan kosong apabila isi pointer head adalah NULL.

b. Linked List Circular

Linked List Circular merupakan variasi dari linked list dimana semua nodenya terhubung sehingga membentuk lingkaran. Artinya tidak ada NULL di akhir. Node terakhir, alih-alih menunjuk ke NULL, menunjuk ke node pertama. singly linked list or a doubly linked list. Linked List yang semua nodenya terhubung membentuk Circular. Dalam Linked List Circular, simpul pertama dan simpul terakhir dihubungkan satu sama lain sehingga membentuk lingkaran. Tidak ada NULL di akhir.



Gambar 2 Linked List Circular

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\ASUS\Documents\semester 2\struktur da
ak 4\" ; if ($?) { g++ guided1.cpp -o 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

```

+

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

Deskripsi

Program diatas adalah implementasi sederhana dari struktur data linked list. Linked list adalah struktur data linear di mana elemen-elemennya terhubung melalui pointer. mendefinisikan sebuah struktur Node yang memiliki dua anggota: data (bertipe int) dan

pointer next. Head dan tail bertipe pointer ke Node, yang digunakan untuk menandai awal (head) dan akhir (tail) dari linked list. Fungsi main, semua fungsi dalam code program di atas akan memanggilnya secara berurutan.

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;
    }
}

```

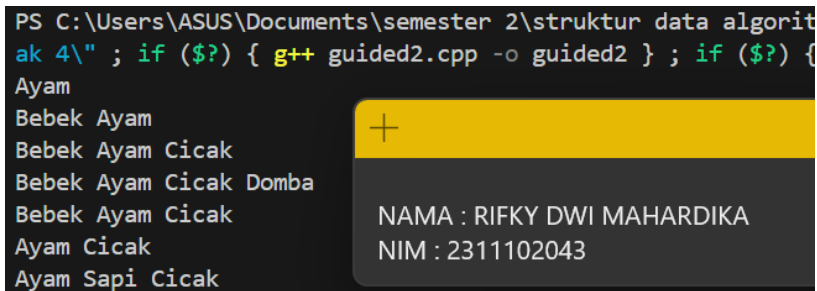


```

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\ASUS\Documents\semester 2\struktur data algorit
ak 4\" ; if ($?) { g++ guided2.cpp -o guided2 } ; if ($?) {
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak

```

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

Deskripsi

Program ini adalah implementasi dari struktur data Linked List Circular. Linked List Circular adalah jenis linked list di mana elemen terakhir menunjuk kembali ke elemen pertama, membentuk sebuah lingkaran. Kode program ini memiliki struktur data Node yang menyimpan string dan pointer ke Node berikutnya. Terdapat fungsi-fungsi untuk inisialisasi, pengecekan apakah linked list kosong, pembuatan Node baru, penambahan elemen di depan, di belakang, atau di tengah linked list, penghapusan elemen di depan, di

belakang, atau di tengah linked list, menghitung jumlah elemen dalam linked list, membersihkan seluruh linked list, dan menampilkan isi linked list. Fungsi main digunakan untuk menguji fungsi-fungsi tersebut dengan menambah, menghapus, dan menampilkan elemen-elemen pada linked list.

C. Unguided

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\ASUS\Documents\semester 2\struktur data algoritma\l
ak 4\" ; if ($?) { g++ unguided1.cpp -o unguided1 } ; if ($?) {

=====
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 : RIFKY DWI MAHARDIKA
NIM : 2311102043

➤ Masukkan data sesuai urutan

```
Pilih Operasi: 2

=== Tambah Belakang ===

Masukkan Nama : Jawad
Masukkan NIM : 23300001

Data telah ditambahkan.

Pilih Operasi: 2

=== Tambah Belakang ===

Masukkan Nama : Rifky Dwi Mahardika
Masukkan NIM : 2311102043

Data telah ditambahkan.
```

+

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043 |

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.



NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043 |

Pilih Operasi: 2

=== Tambah Belakang ===

Source Control (Ctrl+Shift+G)

Masukkan Nama : Bowo
Masukkan NIM : 23300015

Data telah ditambahkan.

Pilih Operasi: 2

=== Tambah Belakang ===

Masukkan Nama : Gahar
Masukkan NIM : 23300040

Data telah ditambahkan.



NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

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.

+

NAMA : RIFKY DWI MAHARDIKA

NIM : 2311102043 |

Pilih Operasi: 2

=== Tambah Belakang ===

Masukkan Nama : Budi

Masukkan NIM : 23300099

Data telah ditambahkan.

+

NAMA : RIFKY DWI MAHARDIKA

NIM : 2311102043 |

Pilih Operasi: 11

DATA MAHASISWA

Nama	NIM

Jawad	23300001
Rifky Dwi Mahardika	2311102043
Farrel	23300003
Denis	23300005
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

+

NAMA : RIFKY DWI MAHARDIKA

NIM : 2311102043 |

- **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.

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

- **Hapus data Denis**

Pilih Operasi: 9

=== Hapus Tengah ===

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

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

- **Tambahkan data berikut di awal: Owi 2330000**

Pilih Operasi: 1

=== Tambah Depan ===

Masukkan Nama : Owi
Masukkan NIM : 2330000

Data telah ditambahkan.

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

- **Tambahkan data berikut di akhir: David 23300100**

Pilih Operasi: 2

=== Tambah Belakang ===

Masukkan Nama : David
Masukkan NIM : 23300100

Data telah ditambahkan.

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

- 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
```

+

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

- Ubah data terkahir menjadi berikut: Lucy 23300101

```
Pilih Operasi: 5

=== Ubah Belakang ===

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

+

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

- Hapus data awal

```
Pilih Operasi: 7

=== Hapus Depan ===

Data Owi telah dihapus.
Pilih Operasi: 
```

+

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

- Ubah data awal menjadi berikut: Bagus 2330002

```
Pilih Operasi: 4

=== Ubah Depan ===

Masukkan Nama : Bagus
Masukkan NIM : 2330002
Data Jawad telah diganti dengan data Bagus
```

+

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

➤ **Hapus data akhir**

```
Pilih Operasi: 8

=== Hapus Belakang ===

Data Lucy telah dihapus.
Pilih Operasi: 
```

+

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

➤ **Tampilkan seluruh data**

```
Pilih Operasi: 11

DATA MAHASISWA

-----
| Nama          | NIM          |
-----
| Bagus         | 2330002      |
| Rifky Dwi Mahardika | 2311102043  |
| Farrel        | 23300003     |
| Wati          | 23300004     |
| Anis          | 23300008     |
| Bowo          | 23300015     |
| Gahar         | 23300040     |
| Idin          | 23300045     |
| Ucok          | 23300050     |
| Budi          | 23300099     |
-----
```

+

NAMA : RIFKY DWI MAHARDIKA
NIM : 2311102043

Deskripsi

Code program tersebut adalah implementasi dari struktur data linked list non-circular. Struktur data ini digunakan untuk menyimpan data mahasiswa, dengan setiap node menyimpan nama dan NIM. Terdapat fungsi-fungsi untuk melakukan operasi seperti penambahan data di depan, di belakang, atau di tengah linked list, penghapusan data di depan, di belakang, atau di tengah linked list, pengubahan data di depan, di belakang, atau di tengah linked list, serta menampilkan seluruh data yang tersimpan. Program ini menggunakan menu operasi untuk memungkinkan pengguna memilih operasi yang diinginkan. Setiap operasi diimplementasikan dengan menggunakan fungsi-fungsi yang telah didefinisikan sebelumnya. Saat pengguna memilih operasi keluar (0), program akan berhenti (keluar).

D. Kesimpulan

Linked list non-circular adalah tipe linked list yang memiliki simpul terakhir (tail) yang bernilai NULL. Sedangkan Linked list circular adalah tipe linked list di mana semua simpul (node) terhubung membentuk sebuah lingkaran. Linked list Circular cocok untuk situasi di mana kita perlu mengulang daftar secara terus-menerus. Linked list Non-circular lebih umum digunakan dan lebih sederhana, tetapi tidak memiliki sifat lingkaran.

E. Referensi

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

RINI WONGSO,S.KOM.,M.T.I. Single Linked List. Di akses pada 16 April 2024
<https://socs.binus.ac.id/2017/03/15/single-linked-list/>

Geeksforgeeks. (2024).Introduction to Circular Linked List. Di akses pada 16 April 2024
<https://www.geeksforgeeks.org/circular-linked-list/>