

LAPORAN PRAKTIKUM
MODUL IV
LINKED LIST CIRCULAR DAN NON CIRCULAR



Disusun oleh:
Raihan Ramadhan
NIM: 2311102040

Dosen Pengampu:
Wahyu Andi Saputra, S.Pd., M.Eng

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

BAB I

TUJUAN PRAKTIKUM

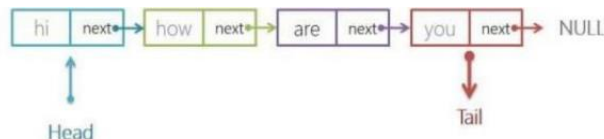
Praktikan dapat mengetahui dan memahami linked list circular dan non circular. Praktikan dapat membuat linked list circular dan non circular. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat. Praktikan juga dapat memahami perbedaan antara penggunaan linked list circular dan non circular dalam konteks pengembangan perangkat lunak. Selain itu, praktikan mampu mengevaluasi keuntungan dan kerugian dari masing-masing jenis linked list untuk memilih solusi yang paling sesuai dalam implementasi program.

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.



OPERASI PADA LINKED LIST NON CIRCULAR

1. Deklarasi Simpul (Node)

```
struct node
{
    int data;
    node *next;
};
```

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
node *head, *tail;
void init()
{
    head = NULL;
    tail = NULL;
};
```

3. Pengecekan Kondisi Linked List

```
bool isEmpty()
{
    if (head == NULL && tail == NULL)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

4. Penambahan Simpul (Node)

```
void insertBelakang(string dataUser)
{
    if (isEmpty() == true)
    {
        node *baru = new node;
        baru->data = dataUser;
        head = baru;
        tail = baru;
        baru->next = NULL;
    }

    else
    {
        node *baru = new node;
        baru->data = dataUser;
        baru->next = NULL;
        tail->next = baru;
        tail = baru;
    }
};
```

5. Penghapusan Simpul (Node)

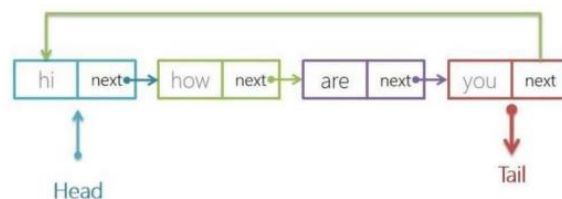
```
void hapusDepan()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        node *helper;
        helper = head;
        if (head == tail)
        {
            head = NULL;
            tail = NULL;
            delete helper;
        }
        else
 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

6. Tampil Data Linked List

```
void tampil()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        node *helper;
        helper = head;
        while (helper != NULL)
        {
            cout << helper->data << ends;
            helper = helper->next;
        }
    }
}
```

2. Linked List Circullar

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.



1. Deklarasi Simpul (Node)

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

2. Membuat dan Menginisialisasi Pointer Head dan Tail

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

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

3. Pengecekan Kondisi Linked List

```
int isEmpty()
{
    if (head == NULL)
        return 1; // true
    else
        return 0; // false
}
```

4. Pembuatan Simpul (Node)

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

5. Penambahan Simpul (Node)

```
// 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;
    }
}
```

6. Penghapusan Simpul (Node)

```
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;
        }
    }
}
```

7. Menampilkan Data Linked List

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

BAB III

GUIDED

1. Guided 1

Source Code

```
#include <iostream>

using namespace std;

// PROGRAM SINGLE LINKED LIST NON-CIRCULAR

// Deklarasi struct node
struct Node
{
    int data;
    Node *next;
};

Node *head; // Deklarasi head
Node *tail; // Deklarasi tail

// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}

// Pengecekan apakah linked list kosong
bool isEmpty()
{
    if (head == NULL)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```



```
}  
  
// Tambah depan  
void insertDepan(int nilai)  
{  
  
    // buat node baru  
    Node *baru = new Node();  
    baru->data = nilai;  
    baru->next = NULL;  
    if (isEmpty() == true)  
    {  
        head = tail = baru;  
        head->next = NULL;  
    }  
    else  
    {  
        baru->next = head;  
        head = baru;  
    }  
}
```

```
// Tambah belakang  
void insertBelakang(int nilai)  
{  
    // buat node baru  
    Node *baru = new Node();  
    baru->data = nilai;  
    baru->next = NULL;  
    if (isEmpty() == true)  
    {  
        head = tail = baru;  
        head->next = NULL;  
    }  
    else  
    {  
        tail->next = baru;  
        tail = baru;  
    }  
}
```

```

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

// Tambah tengah
void insertTengah(int data, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;

        // tranversing
        bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
    }
}

```

```

        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Hapus depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

// Hapus belakang
void hapusBelakang()
{
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false)
    {
        if (head != tail)
        {
            hapus = tail;
            bantu = head;
            while (bantu->next != tail)

```

```

        {
            bantu = bantu->next;
        }
        tail = bantu;
        tail->next = NULL;
        delete hapus;
    }
    else
    {
        head = tail = NULL;
    }
}
else
{
    cout << "Linked list masih kosong" << endl;
}
}
// Hapus tengah
void hapusTengah(int posisi)
{
    Node *hapus, *bantu, *sebelum;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        int nomor = 1;
        bantu = head;
        while (nomor <= posisi)
        {
            if (nomor == posisi - 1)
            {
                sebelum = bantu;
            }
            if (nomor == posisi)

```

```

        {
            hapus = bantu;
        }
        bantu = bantu->next;
        nomor++;
    }
    sebelum->next = bantu;
    delete hapus;
}
}

// ubah depan
void ubahDepan(int data)
{
    if (isEmpty() == 0)
    {
        head->data = data;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

// ubah tengah
void ubahTengah(int data, int posisi)
{
    Node *bantu;
    if (isEmpty() == 0)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {

```

```

        int nomor = 1;
        bantu = head;
        while (nomor < posisi)
        {
            bantu = bantu->next;
            nomor++;
        }
        bantu->data = data;
    }
}
else
{
    cout << "Linked list masih kosong" << endl;
}
}

// ubah belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

// Hapus list
void clearList()
{
    Node *bantu, *hapus;
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
}

```

```

    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

// Tampilkan list
void tampilList()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "Linked list masih kosong" << endl;
    }
}

int main()
{
    init();
    insertDepan(3);
    tampilList();
    insertBelakang(5);
    tampilList();
    insertDepan(2);
    tampilList();
    insertDepan(1);
    tampilList();
    hapusDepan();
    tampilList();
    hapusBelakang();
    tampilList();
    insertTengah(7, 2);
}

```

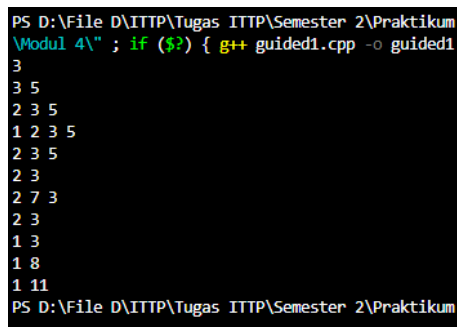
```

    tampilList();
    hapusTengah(2);
    tampilList();
    ubahDepan(1);
    tampilList();
    ubahBelakang(8);
    tampilList();
    ubahTengah(11, 2);
    tampilList();

    return 0;
}

```

Screenshot Program



```

PS D:\File D\ITTP\Tugas ITTP\Semester 2\Praktikum
Modul 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
PS D:\File D\ITTP\Tugas ITTP\Semester 2\Praktikum

```

Deskripsi Program

Program tersebut linked list non-circular menggunakan bahasa pemrograman C++. Dalam program ini, terdapat fungsi-fungsi dasar seperti penambahan (depan, belakang, dan tengah), penghapusan (depan, belakang, dan tengah), serta pembaruan nilai (depan, belakang, dan tengah). Setiap elemen dalam linked list direpresentasikan oleh struktur data node yang memiliki dua bagian: data (nilai) dan pointer yang menunjuk ke node berikutnya.

2. Guided 2

Source Code

```
#include <iostream>

using namespace std;

// Deklarasi Struct Node

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

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

// Inisialisasi node head & tail
void init()
{
    head = NULL;
    tail = head;
}

// Pengecekan isi list
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;
}

```

```

}

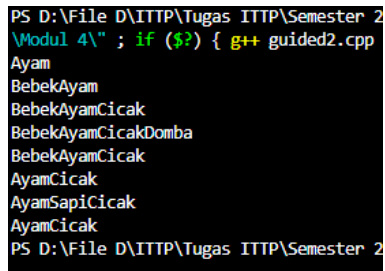
// Tampilkan List
void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
  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

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;
}
```

Screenshot Program



```
PS D:\File D\ITTP\Tugas ITTP\Semester 2
Modul 4\' ; if ($?) { g++ guided2.cpp
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
AyamCicak
PS D:\File D\ITTP\Tugas ITTP\Semester 2
```

Deskripsi Program

Program ini merupakan implementasi dari struktur data linked list sirkular menggunakan bahasa pemrograman C++. Program ini memungkinkan penggunaan operasi dasar seperti penambahan elemen di depan, di belakang, dan di tengah, serta penghapusan elemen dari depan, belakang, dan tengah linked list. Setiap elemen dalam linked list direpresentasikan oleh sebuah node yang memiliki data dan pointer yang menunjuk ke node berikutnya.

BAB IV

UNGUIDED

```
#include <iostream>
#include <iomanip>

using namespace std;

struct mahasiswa
{
    string nama;
    string nim;
};

struct node
{
    mahasiswa identitas;
    node *prev;
    node *next;
};

node *head, *tail;

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

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

mahasiswa mintaData()
{
    system("cls");
    mahasiswa identitas;
```

```
    cout << "\nMasukkan Nama\t: ";
    cin.ignore();
    getline(cin, identitas.nama);
    cout << "Masukkan NIM\t: ";
    cin >> identitas.nim;
    return identitas;
}

void insertDepan(mahasiswa identitas)
{
    node *baru = new node;
    baru->identitas = identitas;
    baru->next = head;
    baru->prev = NULL;
    if (isEmpty())
    {
        tail = baru;
    }
    else
    {
        head->prev = baru;
    }
    head = baru;
}

void insertBelakang(mahasiswa identitas)
{
    node *baru = new node;
    baru->identitas = identitas;
    baru->next = NULL;
    baru->prev = tail;
    if (isEmpty())
    {
        head = baru;
    }
    else
    {
        tail->next = baru;
    }
    tail = baru;
}
```

```

}

void insertTengah(mahasiswa identitastitas, int posisi)
{
    if (posisi <= 1)
    {
        insertDepan(identitastitas);
        return;
    }
    node *baru = new node;
    baru->identitas = identitastitas;
    node *bantu = head;
    for (int i = 1; i < posisi - 1 && bantu != NULL; i++)
    {
        bantu = bantu->next;
    }
    if (bantu == NULL)
    {
        cout << "Posisi diluar jangkauan" << endl;
        return;
    }
    baru->next = bantu->next;
    baru->prev = bantu;
    if (bantu->next != NULL)
    {
        bantu->next->prev = baru;
    }
    bantu->next = baru;
}

void ubahNode(node *target, mahasiswa data)
{
    target->identitas = data;
}

void ubahDepan(mahasiswa data)
{
    if (isEmpty())
    {
        cout << "List kosong" << endl;
    }
}

```

```

        return;
    }
    ubahNode(head, data);
}

void ubahBelakang(mahasiswa data)
{
    if (isEmpty())
    {
        cout << "List kosong" << endl;
        return;
    }
    ubahNode(tail, data);
}

void ubahTengah(mahasiswa data, string nim)
{
    node *bantu = head;
    while (bantu != NULL)
    {
        if (bantu->identitas.nim == nim)
        {
            ubahNode(bantu, data);
            return;
        }
        bantu = bantu->next;
    }
    cout << "Data dengan NIM " << nim << " tidak ditemukan" << endl;
}

void tampil()
{
    system("cls");
    node *bantu = head;
    cout << "Nama "
        << " Nim\n";
    while (bantu != NULL)
    {
        cout << bantu->identitas.nama << " " << bantu->identitas.nim << endl;
        bantu = bantu->next;
    }
}

```

```

    }
}

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

void hapusBelakang()
{
    if (isEmpty())
    {
        cout << "List kosong" << endl;
        return;
    }
    node *hapus = tail;
    if (head == tail)
    {
        head = tail = NULL;
    }
    else
    {
        tail = tail->prev;
        tail->next = NULL;
    }
}

```

```

    delete hapus;
}

void hapusTengah(string nim)
{
    if (isEmpty())
    {
        cout << "List kosong" << endl;
        return;
    }
    node *bantu = head;
    while (bantu != NULL)
    {
        if (bantu->identitas.nim == nim)
        {
            if (bantu == head)
            {
                hapusDepan();
            }
            else if (bantu == tail)
            {
                hapusBelakang();
            }
            else
            {
                bantu->prev->next = bantu->next;
                bantu->next->prev = bantu->prev;
                delete bantu;
            }
            return;
        }
        bantu = bantu->next;
    }
    cout << "Data dengan NIM " << nim << " tidak ditemukan" << endl;
}

void hapusList()
{
    while (!isEmpty())
    {

```

```

        hapusDepan();
    }
    cout << "Semua data berhasil dihapus" << endl;
}

int main()
{
    init();
    mahasiswa identitas;
    int operasi, posisi;

    do
    {
        cout << "PROGRAM DOUBLE 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.Exit" << endl;
        cout << "\nPilih Operasi : ";
        cin >> operasi;

        switch (operasi)
        {
            case 1:
                cout << "Tambah Depan\n";
                insertDepan(mintaData());
                cout << endl;
                break;
            case 2:
                cout << "Tambah Belakang\n";
                insertBelakang(mintaData());
                cout << endl;

```

```
        break;
    case 3:
        cout << "Tambah Tengah\n";
        cout << "Nama : ";
        cin >> identitas.nama;
        cout << "NIM : ";
        cin >> identitas.nim;
        cout << "Posisi : ";
        cin >> posisi;
        insertTengah(identitas, posisi);
        cout << endl;
        break;
    case 4:
        cout << "Ubah Depan\n";
        ubahDepan(mintaData());
        cout << endl;
        break;
    case 5:
        cout << "Ubah Belakang\n";
        ubahBelakang(mintaData());
        cout << endl;
        break;
    case 6:
        cout << "Ubah Tengah\n";
        cout << "NIM : ";
        cin >> identitas.nim;
        ubahTengah(mintaData(), identitas.nim);
        cout << endl;
        break;
    case 7:
        cout << "Hapus Depan\n";
        hapusDepan();
        cout << endl;
        break;
    case 8:
        cout << "Hapus Belakang\n";
        hapusBelakang();
        cout << endl;
        break;
    case 9:
```



```
        cout << "Hapus Tengah\n";
        cout << "NIM : ";
        cin >> identitas.nim;
        hapusTengah(identitas.nim);
        cout << endl;
        break;

    case 10:
        cout << "Hapus List\n";
        hapusList();
        cout << endl;
        break;
    case 11:
        cout << "Tampilkan\n";
        tampil();
        cout << endl;
        break;
    case 0:
        cout << "Exit Program\n";
        break;
    default:
        cout << "Salah input operasi\n";
        cout << endl;
        break;
    }
} while (operasi != 0);

return 0;
}
```

Outpout Program

Screenshoot Program Nomer 1 :

Tampil Menu

```
PS D:\File D\ITTP\Tugas ITI
goritma\Modul 4\" ; if ($?)
PROGRAM DOUBLE LINKED LIST

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.Exit

Pilih Operasi :
```

Tambah Depan

```
Nama Nim
Nadhif 2311102070
Raihan 2311102040
Rizal 2311102060
Tegar 2311102050
Denny 23111030

PROGRAM DOUBLE 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.Exit

Pilih Operasi : 1
```

Tambah Tengah

```
Pilih Operasi : 3
Tambah Tengah
Nama : Alul
NIM : 2311102080
Posisi : 3
```

```
Nama Nim
Nadhif 2311102070
Raihan 2311102040
Alul 2311102080
Rizal 2311102060
Tegar 2311102050
Denny 23111030
```

Hapus Belakang

```
Pilih Operasi : 8
Hapus Belakang
```

```
Nama Nim
Nadhif 2311102070
Raihan 2311102040
Alul 2311102080
Rizal 2311102060
Tegar 2311102050
```

Hapus Tengah

```
Pilih Operasi : 9
Hapus Tengah
NIM : 2311102080
```

```
Nama Nim
Nadhif 2311102070
Raihan 2311102040
Rizal 2311102060
Tegar 2311102050
```

Ubah Belakang

```
Masukkan Nama : Suryo
Masukkan NIM : 231110700
```

```
Nama Nim
Nadhif 2311102070
Raihan 2311102040
Rizal 2311102060
Suryo 231110700
```

Ubah Tengah & Tampilan Kondisi Data Terakhir

```
Masukkan Nama : Daus
Masukkan NIM : 23111020100
```

```
Nama Nim
Nadhif 2311102070
Raihan 2311102040
Daus 23111020100
Suryo 231110700
```

Screenshoot Program Nomer 2 :

```
Nama  Nim
Jawad 23300001
Raihan Ramadhan 2311102040
Farrel 23300003
Denis 23300005
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099
```

Screenshoot Program Nomer 3 :

A. Tambah data Wati 23300004 diantara Farrel dan Denis

```
Nama  Nim
Jawad 23300001
Raihan Ramadhan 2311102040
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
NIM : 23300005
```

```
Nama  Nim
Jawad 23300001
Raihan Ramadhan 2311102040
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099
```

C. Tambahkan Data Berikut Diawal

```
Masukkan Nama : Owi
Masukkan NIM : 2330000
```

```
Nama Nim
Owi 2330000
Jawad 23300001
Raihan Ramadhan 2311102040
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099
```

D. Tambahkan Data Berikut Diakhir

```
Masukkan Nama : David
Masukkan NIM : 23300100
```

```
Nama Nim
Owi 2330000
Jawad 23300001
Raihan Ramadhan 2311102040
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099
David 23300100
```

E. Ubah Data Udin menjadi Idin

```
Masukkan Nama : Idin
Masukkan NIM : 23300045
```

```
Nama Nim
Owi 2330000
Jawad 23300001
Raihan Ramadhan 2311102040
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
David 23300100
```

F. Ubah Data Terakhir Menjadi Lucy

```
Masukkan Nama : Lucy
Masukkan NIM : 23300101
```

```
Nama Nim
Owi 2330000
Jawad 23300001
Raihan Ramadhan 2311102040
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
Hapus Depan
```

```
Nama Nim
Jawad 23300001
Raihan Ramadhan 2311102040
Farrel 23300003
Wati 2330004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
Lucy 23300101
```

H. Ubah Data Awal Menjadi Bagus

```
Masukkan Nama : Bagus
Masukkan NIM : 2330002
```

```
Nama Nim
Bagas 2330002
Raihan Ramadhan 2311102040
Farrel 23300003
Wati 2330004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
Lucy 23300101
```

I. Hapus Data Akhir

```
Pilih Operasi : 8  
Hapus Belakang
```

```
Nama  Nim  
Bagas 2330002  
Raihan Ramadhan 2311102040  
Farrel 23300003  
Wati 2330004  
Anis 23300008  
Bowo 23300015  
Gahar 23300040  
Idin 23300045  
Ucok 23300050  
Budi 23300099
```

J. Tampilkan Seluruh Data

```
Nama  Nim  
Bagas 2330002  
Raihan Ramadhan 2311102040  
Farrel 23300003  
Wati 2330004  
Anis 23300008  
Bowo 23300015  
Gahar 23300040  
Idin 23300045  
Ucok 23300050  
Budi 23300099
```

Deskripsi Program

Program ini merupakan double linked list non-circular menggunakan bahasa pemrograman C++. Program ini memungkinkan pengguna untuk melakukan operasi dasar seperti penambahan, penghapusan, dan pengubahan data mahasiswa. Operasi yang disediakan antara lain adalah penambahan data di depan, di belakang, atau di tengah list, pengubahan data di depan, di belakang, atau di tengah list berdasarkan NIM, penghapusan data di depan, di belakang, atau di tengah list berdasarkan NIM, dan juga penghapusan seluruh data.

BAB IV

KESIMPULAN

Dari praktikum Linked List Non-Circular, dapat disimpulkan bahwa struktur data ini memberikan keleluasaan dalam penyimpanan serta pengelolaan data, terutama ketika data yang dimasukkan memiliki panjang atau jenis yang bervariasi. Dengan menggunakan linked list, kita dapat dengan mudah menambah, mengubah, atau menghapus data tanpa harus terikat pada urutan tertentu, sehingga proses tersebut menjadi lebih efisien.

DAFTAR PUSTAKA

Asisten Praktikum “Modul 4 LINKED LIST CIRCULAR DAN NON CIRCULAR”. Learning Management System, 14 Maret 2024.