LAPORAN PRAKTIKUM

MODUL 4

LINKED LIST CIRCULAR DAN NON CIRCULAR

Disusun oleh:
Nabila Shasya Sabrina
NIM: 2311102039

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

a. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.

b. Praktikan dapat membuat linked list circular dan non circular.

c. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

BAB II
DASAR TEORI

1. **Linked List Non Circular**

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

2. **Linked List Circular**

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

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

BAB III

GUIDED

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;
Node *tail;
// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}
// Pengecekan
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;
        tail->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;
        tail->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 diluar 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 << "List 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 << "List kosong!" << endl;
    }
}
// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *bantu, *hapus, *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 << "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)
        {
        }
        else
        {
            cout << "Posisi bukan posisi tengah" << endl;
            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }
    else
    {
        cout << "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 tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << ends;
            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;
}
```

Screenshoot Program:

```
3
35
235
1235
235
23
273
23
13
18
Posisi bukan posisi tengah
111
PS D:\Data\Kuliah\Semester 2\Praktikum Struktur Data dan Algoritma\Modul 4>
```

Deskripsi program:

Program tersebut adalah implementasi dari single linked list non-circular. Ini adalah struktur data yang terdiri dari node-node yang memiliki dua anggota, yaitu `'data'` untuk menyimpan nilai, dan `'next'` untuk menunjukkan ke node berikutnya dalam linked list. Fungsi-fungsi ini memungkinkan manipulasi data pada linked list, seperti penambahan, penghapusan, dan pengubahan nilai, serta penampilan isi linked list. Program ini menggunakan tiga variabel global `'head'`, `'tail'`, dan `'next'` untuk mengatur operasi-operasi tersebut.

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;

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

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

```
}

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

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

// Tambah Depan
void insertDepan(string data)
{
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        while (tail->next != head)
        {
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}

// Tambah Belakang
void insertBelakang(string data)
{
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        while (tail->next != head)
        {
```

```
            tail = tail->next;
        }
        tail->next = baru;
        baru->next = head;
    }
}

// Tambah Tengah
void insertTengah(string data, int posisi)
{
    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        baru->data = data;
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Hapus Depan
void hapusDepan()
{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;
            delete hapus;
        }
        else
        {
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            head = head->next;
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
```

```
// Hapus Belakang
void hapusBelakang()
{
    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;
            delete hapus;
        }
        else
        {
            while (hapus->next != head)
            {
                hapus = hapus->next;
            }
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;
            delete hapus;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus Tengah
void hapusTengah(int posisi)
{
    if (isEmpty() == 0)
    {
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus List
void clearList()
{
    if (head != NULL)
    {
        hapus = head->next;
```

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

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

int main()
{
    init();
    insertDepan("Ayam");
    tampil();
    insertDepan("Bebek");
    tampil();
    insertBelakang("Cicak");
    tampil();
    insertBelakang("Domba");
    tampil();
    hapusBelakang();
    tampil();
    hapusDepan();
    tampil();
    insertTengah("Sapi", 2);
    tampil();
    hapusTengah(2);
    tampil();
    return 0;
}
```

Screenshoot program:

```
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
AyamCicak
PS D:\Data\Kuliah\Semester 2\Praktikum Struktur Data dan Algoritma\Modul 4>
```

Deskripsi Program:

Program di atas adalah implementasi dari single linked list circular. Single linked list adalah struktur data yang terdiri dari sejumlah node yang saling terhubung satu sama lain melalui pointer. Setiap node memiliki dua bagian yaitu data untuk menyimpan nilai, dan next untuk menunjukkan ke node berikutnya dalam linked list.

Pada program ini, `'struct Node'` dideklarasikan dengan dua anggota, yaitu `'data'` bertipe string dan `'next'` bertipe pointer Node. Selain itu, program juga mendeklarasikan beberapa variabel global, seperti `'head'`, `'tail'`, `'baru'`, `'bantu'`, dan `'hapus'`.

BAB IV

UNGUIDED

Source code:

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

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

bool isEmpty(Node *head)
{
    return head == nullptr;
}

Node* buatNode(string nama, string nim)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = nullptr;
    return baru;
}

Node* tambahDepan(Node *head, string nama, string nim)
{
    Node *baru = buatNode(nama, nim);
    if (isEmpty(head))
    {
        return baru;
    }
    baru->next = head;
    return baru;
}

Node* tambahBelakang(Node *head, string nama, string nim)
{
    Node *baru = buatNode(nama, nim);
    if (isEmpty(head))
    {
        return baru;
    }
    Node *tail = head;
    while (tail->next != nullptr)
    {
        tail = tail->next;
    }
    tail->next = baru;
    return head;
}

Node* tambahTengah(Node *head, string nama, string nim, int posisi)
{
    if (posisi < 1)
    {
```

```
        cout << "Posisi tidak valid" << endl;
        return head;
    }
    if (posisi == 1)
    {
        cout << "Gunakan tambahDepan untuk menambahkan pada posisi
pertama" << endl;
        return tambahDepan(head, nama, nim);
    }
    Node *baru = buatNode(nama, nim);
    Node *bantu = head;
    for (int i = 1; i < posisi - 1 && bantu != nullptr; i++)
    {
        bantu = bantu->next;
    }
    if (bantu == nullptr)
    {
        cout << "Posisi diluar jangkauan" << endl;
        return head;
    }
    baru->next = bantu->next;
    bantu->next = baru;
    return head;
}

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

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

void ubahTengah(Node *head, string nama, string nim, int posisi)
{
    if (!isEmpty(head))
    {
        if (posisi < 1)
        {
            cout << "Posisi tidak valid" << endl;
```

```
            return;
        }
        Node *bantu = head;
        for (int i = 1; i < posisi && bantu != nullptr; i++)
        {
            bantu = bantu->next;
        }
        if (bantu == nullptr)
        {
            cout << "Posisi diluar jangkauan" << endl;
            return;
        }
        bantu->nama = nama;
        bantu->nim = nim;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

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

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

Node* hapusTengah(Node *head, int posisi)
```

```
{
    if (!isEmpty(head))
    {
        if (posisi < 1)
        {
            cout << "Posisi tidak valid" << endl;
            return head;
        }
        if (posisi == 1)
        {
            return hapusDepan(head);
        }
        Node *bantu = head;
        for (int i = 1; i < posisi - 1 && bantu != nullptr; i++)
        {
            bantu = bantu->next;
        }
        if (bantu == nullptr || bantu->next == nullptr)
        {
            cout << "Posisi diluar jangkauan" << endl;
            return head;
        }
        Node *hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
    return head;
}

void hapusList(Node *&head)
{
    while (!isEmpty(head))
    {
        head = hapusDepan(head);
    }
    cout << "List berhasil terhapus!" << endl;
}

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

void tampil(Node *head)
{
    if (!isEmpty(head))
    {
        cout << "=====================================" << endl;
        cout << "            DATA MAHASISWA" << endl;
        cout << "=====================================" << endl;
        cout << "|   No. |       Nama       |    NIM    |" << endl;
        cout << "-------------------------------------" << endl;
```

```
        Node *bantu = head;
        int no = 1;
        while (bantu != nullptr)
        {
             cout << "|" << setw(7) << no << " | " << setw(15) <<
bantu->nama << " | " << setw(9) << bantu->nim << " |" << endl;
            bantu = bantu->next;
            no++;
        }
        cout << "--------------------------------------" << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    Node *head = nullptr;
    int choice, posisi;
    string nama, nim;
    do
    {
        cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << endl;
        cout << "1. Tambah Depan" << endl;
        cout << "2. Tambah Belakang" << endl;
        cout << "3. Tambah Tengah" << endl;
        cout << "4. Ubah Depan" << endl;
        cout << "5. Ubah Belakang" << endl;
        cout << "6. Ubah Tengah" << endl;
        cout << "7. Hapus Depan" << endl;
        cout << "8. Hapus Belakang" << endl;
        cout << "9. Hapus Tengah" << endl;
        cout << "10. Hapus List" << endl;
        cout << "11. TAMPILKAN" << endl;
        cout << "0. KELUAR" << endl;
        cout << "Pilih Operasi: ";
        cin >> choice;

        switch (choice)
        {
            case 1:
                cout << "-Tambah Depan" << endl;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                head = tambahDepan(head, nama, nim);
                cout << "Data telah ditambahkan" << endl;
                break;
            case 2:
                cout << "-Tambah Belakang" << endl;
                cout << "Masukkan Nama: ";
                cin >> nama;
                cout << "Masukkan NIM: ";
                cin >> nim;
                head = tambahBelakang(head, nama, nim);
                cout << "Data telah ditambahkan" << endl;
                break;
            case 3:
                cout << "-Tambah Tengah" << endl;
                cout << "Masukkan Nama : ";
```

```
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                cout << "Masukkan Posisi : ";
                cin >> posisi;
                head = tambahTengah(head, nama, nim, posisi);
                cout << "Data telah ditambahkan" << endl;
                break;
            case 4:
                cout << "Masukkan Nama: ";
                cin >> nama;
                cout << "Masukkan NIM: ";
                cin >> nim;
                ubahDepan(head, nama, nim);
                break;
            case 5:
                cout << "-Ubah Belakang" << endl;
                cout << "Masukkan nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                ubahBelakang(head, nama, nim);
                cout << "Data (nama lama) telah diganti dengan data
(nama baru)" << endl;
                break;
            case 6:
                cout << "Masukkan Nama: ";
                cin >> nama;
                cout << "Masukkan NIM: ";
                cin >> nim;
                cout << "Masukkan posisi: ";
                cin >> posisi;
                ubahTengah(head, nama, nim, posisi);
                break;
            case 7:
                head = hapusDepan(head);
                break;
            case 8:
                cout << "-Hapus Belakang" << endl;
                head = hapusBelakang(head);
                cout << "Data (nama mahasiswa yang dihapus) berhasil
dihapus" << endl;
                break;
            case 9:
                cout << "-Hapus Tengah" << endl;
                cout << "Masukkan posisi : ";
                cin >> posisi;
                head = hapusTengah(head, posisi);
                cout << "Data (nama mahasiswa yang dihapus) berhasil
dihapus" << endl;
                break;
            case 10:
                hapusList(head);
                break;
            case 11:
                tampil(head);
                break;
            case 0:
                cout << "Terima kasih!" << endl;
                break;
            default:
                cout << "Pilihan tidak valid!" << endl;
                break;
```

```
        }
    } while (choice != 0);

    return 0;
}
```

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

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 :
```

- Tampilan Operasi Tambah:

```
-Tambah Depan-

Masukkan Nama :
Masukkan NIM  :

Data telah ditambahkan
```

```
-Tambah Tengah-

Masukkan    Nama     :
Masukkan     NIM     :
Masukkan Posisi :

Data telah ditambahkan
```

- Tampilan Operasi Hapus:

```
-Hapus Belakang-

Data (nama mahasiswa yang dihapus) berhasil dihapus
```

```
-Hapus Tengah- Masukkan

posisi :

Data (nama mahasiswa yang dihapus) berhasil dihapus
```

- Tampilan Operasi Ubah:

```
-Ubah Belakang-

Masukkan nama :
Masukkan NIM  :

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

```
-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
```

***Buat tampilan output sebagus dan secantik mungkin sesuai kreatifitas anda masing-masing, jangan terpaku pada contoh output yang diberikan.**

Screenshot Output:

- 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:
```

- Tampilan Operasi Tambah:

```
-Tambah Depan
Masukkan Nama : Nabila
Masukkan NIM : 2311102039
Data telah ditambahkan
```

```
-Tambah Tengah
Masukkan Nama : Sabrina
Masukkan NIM : 2311102066
Masukkan Posisi : 2
Data telah ditambahkan
```

```
-Tambah Belakang
Masukkan Nama: Shasya
Masukkan NIM: 2311102006
Data telah ditambahkan
```

- Tampilan Operasi Ubah:

```
Pilih Operasi: 4
Masukkan Nama: Alel
Masukkan NIM: 2211171002
```

```
Pilih Operasi: 6
Masukkan Nama: Yudha
Masukkan NIM: 2211101777
Masukkan posisi: 2
```

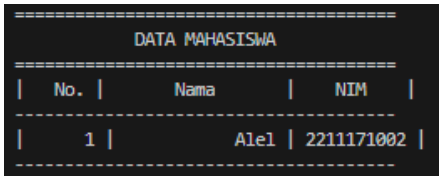
```
Masukkan nama : Aryaputra
Masukkan NIM : 2211177021
Data (nama lama) telah diganti dengan data (nama baru)
```

- Tampilan Operasi Hapus:

```
-Hapus Tengah
Masukkan posisi : 2
Data (nama mahasiswa yang dihapus) berhasil dihapus
```

```
-Hapus Belakang
Data (nama mahasiswa yang dihapus) berhasil dihapus
```

- Tampilan Operasi Tampil Data:

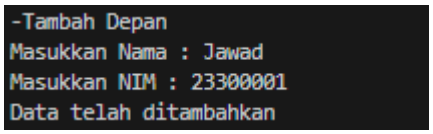


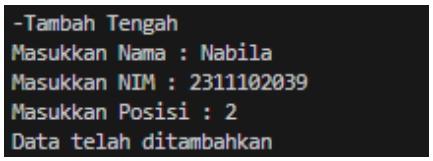
2. Setelah membuat menu tersebut, masukkan data sesuai urutan berikut, lalu tampilkan data yang telah dimasukkan. (Gunakan insert depan, belakang atau tengah)

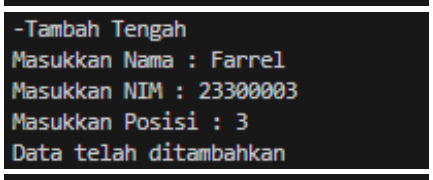
Nama	**NIM**
Jawad	**23300001**
[Nama Anda]	**[NIM Anda]**
Farrel	**23300003**
Denis	**23300005**
Anis	**23300008**
Bowo	**23300015**
Gahar	**23300040**
Udin	**23300048**
Ucok	**23300050**
Budi	**23300099**

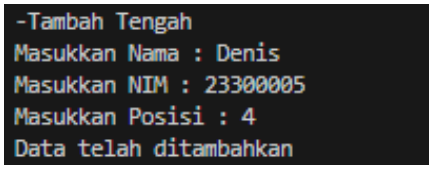
Screenshot Output:

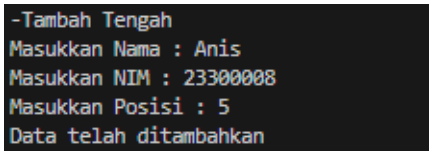
- Tampilan Tambah Data











```
-Tambah Tengah
Masukkan Nama : Bowo
Masukkan NIM : 23300015
Masukkan Posisi : 6
Data telah ditambahkan
```

```
-Tambah Tengah
Masukkan Nama : Gahar
Masukkan NIM : 23300040
Masukkan Posisi : 7
Data telah ditambahkan
```

```
-Tambah Tengah
Masukkan Nama : Udin
Masukkan NIM : 23300048
Masukkan Posisi : 8
Data telah ditambahkan
```

```
-Tambah Tengah
Masukkan Nama : Ucok
Masukkan NIM : 23300050
Masukkan Posisi : 9
Data telah ditambahkan
```

```
-Tambah Belakang
Masukkan Nama: Budi
Masukkan NIM: 23300099
Data telah ditambahkan
```

- Tampilan Data Mahasiswa:

```
====================================
            DATA MAHASISWA
====================================
|   No. |       Nama       |    NIM    |
------------------------------------
|     1 |            Jawad |  23300001 |
|     2 |           Nabila | 2311102039 |
|     3 |           Farrel |  23300003 |
|     4 |            Denis |  23300005 |
|     5 |             Anis |  23300008 |
|     6 |             Bowo |  23300015 |
|     7 |            Gahar |  23300040 |
|     8 |             Udin |  23300048 |
|     9 |             Ucok |  23300050 |
|    10 |             Budi |  23300099 |
------------------------------------
```

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 terkahir menjadi berikut:

Lucy 23300101

g) Hapus data awal

h) Ubah data awal menjadi berikut:

Bagas 2330002

i) Hapus data akhir

j) Tampilkan seluruh data

Screenshot Output:

a) Tambahkan data berikut diantara Farrel dan Denis:

Wati 2330004

```
-Tambah Tengah
Masukkan Nama : Wati
Masukkan NIM : 2330004
Masukkan Posisi : 4
Data telah ditambahkan
```

b) Hapus data Denis

```
-Hapus Tengah
Masukkan posisi : 5
Data (nama mahasiswa yang dihapus) berhasil dihapus
```

c) Tambahkan data berikut di awal:

Owi 2330000

```
-Tambah Depan
Masukkan Nama : Owi
Masukkan NIM : 2330000
Data telah ditambahkan
```

d) Tambahkan data berikut di akhir:

David 23300100

```
-Tambah Belakang
Masukkan Nama: David
Masukkan NIM: 23300100
Data telah ditambahkan
```

e) Ubah data Udin menjadi data berikut:

Idin 23300045

```
Pilih Operasi: 6
Masukkan Nama: Idin
Masukkan NIM: 23300045
Masukkan posisi: 9
```

f) Ubah data terkahir menjadi berikut:

Lucy 23300101

```
-Ubah Belakang
Masukkan nama : Lucy
Masukkan NIM : 23300101
```

g) Hapus data awal

```
Pilih Operasi: 7
```

h) Ubah data awal menjadi berikut:
Bagas 2330002

```
Pilih Operasi: 4
Masukkan Nama: Bagas
Masukkan NIM: 2330002
```

i) Hapus data akhir

```
Pilih Operasi: 8
-Hapus Belakang
Data (nama mahasiswa yang dihapus) berhasil dihapus
```

j) Tampilkan seluruh data

```
=====================================
            DATA MAHASISWA
=====================================
|   No. |       Nama       |    NIM    |
-------------------------------------
|      1 |          Bagas |   2330002 |
|      2 |         Nabila | 2311102039 |
|      3 |         Farrel |  23300003 |
|      4 |           Wati |   2330004 |
|      5 |           Anis |  23300008 |
|      6 |           Bowo |  23300015 |
|      7 |          Gahar |  23300040 |
|      8 |           Idin |  23300045 |
|      9 |           Ucok |  23300050 |
|     10 |           Budi |  23300099 |
-------------------------------------
```

Deskripsi Program:

Program di atas adalah sebuah implementasi dari struktur data linked list satu arah (single linked list) yang tidak bersirkular. Program ini menyediakan beberapa operasi dasar untuk mengelola data mahasiswa, antara lain:

1. `Tambah Depan`: Menambahkan data mahasiswa baru di awal linked list.
2. `Tambah Belakang`: Menambahkan data mahasiswa baru di akhir linked list.
3. `Tambah Tengah`: Menambahkan data mahasiswa baru di posisi tertentu dalam linked list.
4. `Ubah Depan`: Mengubah data mahasiswa di awal linked list.
5. `Ubah Belakang`: Mengubah data mahasiswa di akhir linked list.
6. `Ubah Tengah`: Mengubah data mahasiswa pada posisi tertentu dalam linked list.
7. `Hapus Depan`: Menghapus data mahasiswa di awal linked list.
8. `Hapus Belakang`: Menghapus data mahasiswa di akhir linked list.
9. `Hapus Tengah`: Menghapus data mahasiswa pada posisi tertentu dalam linked list.
10. `Hapus List`: Menghapus seluruh data mahasiswa dari linked list.
11. `Tampilkan`: Menampilkan seluruh data mahasiswa yang ada dalam linked list.

BAB V
KESIMPULAN

Linked list non-circular memiliki elemen terakhir yang menunjuk pada NULL, sedangkan linked list circular memiliki elemen terakhir yang menunjuk kembali pada elemen pertama, membentuk lingkaran. Operasi penambahan dan penghapusan elemen dalam circular linked list sering kali lebih efisien karena tidak perlu menyesuaikan pointer terakhir. Namun, implementasi circular linked list memerlukan perhatian ekstra untuk menghindari looping tak terbatas. Pemilihan antara keduanya bergantung pada kebutuhan spesifik aplikasi dan preferensi desain.

DAFTAR PUSTAKA

Asisten Praktikum. 2024. "*MODUL 4 LINKED LIST CIRCULAR DAN NON CIRCULAR*". Learning Management System.