

LAPORAN PRAKTIKUM

MODUL IV LINKED LIST CIRCULAR DAN NON CIRCULAR



Disusun oleh:
Anita Nurazizah Agussalim
NIM: 2311102017

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

1. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
2. Praktikan dapat membuat linked list circular dan non circular.
3. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

BAB II

DASAR TEORI

a) **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.

b) **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;
        }

        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;
    }
}

// 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;  
}
```

Screenshot Output

```
3
35
235
1235
235
23
273
23
13
18
Posisi bukan posisi tengah
111
```

Deskripsi program

Program di atas adalah implementasi dari sebuah single linked list non-circular. Dalam program ini, terdapat definisi dari struktur Node yang memiliki dua anggota: data dan pointer ke Node selanjutnya. Terdapat fungsi-fungsi untuk menginisialisasi, mengecek apakah linked list kosong, menambahkan elemen di depan, belakang, atau di posisi tengah, menghapus elemen di depan, belakang, atau di posisi tengah, serta mengubah nilai data pada elemen di depan, belakang, atau di posisi tengah. Program ini juga memiliki fungsi untuk menghitung jumlah elemen dalam linked list dan menampilkan isi dari linked list. Pada fungsi `main()`, dilakukan pengujian terhadap berbagai operasi yang telah didefinisikan sebelumnya untuk memanipulasi linked list.

2. Guided 2

Source code

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST CIRCULAR

// 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;
        }
    }
}
```

```

    }
    else
    {
        delete hapus;

        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;
        }
    }
    else

```

```

        {
            delete hapus;

            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

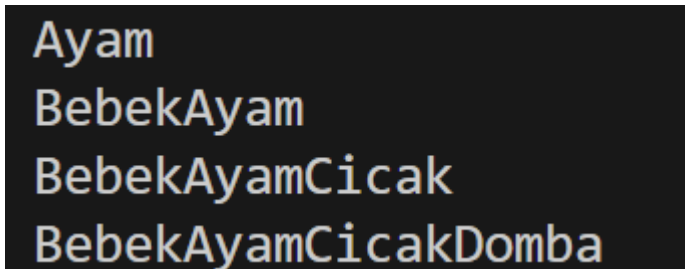
```

```
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 Output



```
Ayam  
BebekAyam  
BebekAyamCicak  
BebekAyamCicakDomba
```

Deskripsi program

Program di atas merupakan implementasi dari sebuah single linked list circular. Struktur data ini terdiri dari sebuah struktur Node yang menyimpan data string dan pointer ke Node selanjutnya. Terdapat fungsi-fungsi untuk menginisialisasi linked list, mengecek apakah linked list kosong, menambahkan elemen di depan, belakang, atau di posisi tengah, menghapus elemen di depan, belakang, atau di posisi tengah, menghitung jumlah elemen dalam linked list, membersihkan seluruh isi linked list, dan menampilkan isi linked list. Pada fungsi `'main()'`, dilakukan pengujian terhadap berbagai operasi yang telah didefinisikan sebelumnya untuk memanipulasi linked list.

LATIHAN KELAS - UNGUIDED

Unguided

Source code

```
#include <iostream>
#include <string>
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))
    {
        Node *bantu = head;
        cout << "=====" << endl;
        cout << "    DATA MAHASISWA" << endl;
        cout << "=====" << endl;
        cout << "|          NAMA          |    NIM    |" << endl;
        cout << "-----" << endl;
        while (bantu != nullptr)
        {
            cout << "|    " << bantu->nama << "    |    " <<
bantu->nim << "    |" << endl;
            bantu = bantu->next;
        }
        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 telah diganti dengan data yang
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);
        cout << "Data telah diganti dengan data yang
baru" << endl;

        break;
    case 7:
        head = hapusDepan(head);

        cout << "Data telah berhasil dihapus" << endl;
        break;
    case 8:
        cout << "-Hapus Belakang" << endl;
        head = hapusBelakang(head);
        cout << "Data telah berhasil dihapus" << endl;
        break;
    case 9:
        cout << "-Hapus Tengah" << endl;
        cout << "Masukkan posisi : ";
        cin >> posisi;
        head = hapusTengah(head, posisi);
        cout << "Data " << nama << "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;
}

```

1. Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM mahasiswa, berikut 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 : anita
Masukkan NIM : 2311102017
Data telah ditambahkan

```

```
-Tambah Tengah  
Masukkan Nama : Nurazizah  
Masukkan NIM : 23111020177  
Masukkan Posisi : 2  
Data telah ditambahkan
```

```
-Tambah Belakang  
Masukkan Nama: Agussalim  
Masukkan NIM: 231110201777  
Data telah ditambahkan
```

- Tampilan Operasi Ubah:

```
Pilih Operasi: 4  
Masukkan Nama: tyler  
Masukkan NIM: 5550153
```

```
Pilih Operasi: 6  
Masukkan Nama: marla  
Masukkan NIM: 5550143  
Masukkan posisi: 2
```

```
-Ubah Belakang  
Masukkan nama : narrator  
Masukkan NIM : 5550153
```

- Tampilan Operasi Hapus:

```
Pilih Operasi: 8  
-Hapus Belakang  
Data telah berhasil dihapus
```

```
-Hapus Tengah  
Masukkan posisi : 2  
Data narrator berhasil dihapus
```

- Tampilan Operasi Tampil data:

```
Pilih Operasi: 11
=====
DATA MAHASISWA
=====
|  NAMA          |  NIM  |
-----
|  tyler         | 5550153 |
-----
```

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

```
=====
DATA MAHASISWA
=====
|  NAMA          |  NIM  |
-----
|  Jawad         | 23300001 |
|  Anita         | 2311102017 |
|  Farrel        | 23300003 |
|  Denis         | 23300005 |
|  Anis          | 23300008 |
|  Bowo          | 23300015 |
|  Gahar         | 23300040 |
|  Udin          | 23300048 |
|  Ucok          | 23300050 |
|  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 terakhir menjadi berikut:
Lucy 23300101
- g) Hapus data awal
- h) Ubah data awal menjadi berikut:
Bagas 2330002
- i) Hapus data akhir
- j) Tampilkan seluruh data

```
=====
DATA MAHASISWA
=====
```

NAMA	NIM
Bagas	2330002
Anita	2311102017
Farrel	23300003
Wati	2330004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099

```
=====
```

BAB IV

KESIMPULAN

Linked list non-circular, juga dikenal sebagai singly linked list, memiliki simpul pertama dan terakhir yang tidak saling terhubung. Setiap simpul hanya memiliki satu pointer yang menunjuk ke simpul berikutnya, dan simpul terakhir menunjuk ke null, menandakan akhir dari daftar. Linked list non-circular digunakan untuk mengelola data secara berurutan dengan efisien, seperti dalam daftar kontak, daftar tugas, dan daftar putar musik.

Di sisi lain, linked list circular membentuk struktur lingkaran di mana simpul terakhir terhubung kembali ke simpul pertama (head). Linked list ini berguna untuk situasi di mana data perlu diakses berulang kali. Contohnya adalah dalam buffer lingkaran, implementasi antrian, dan alokasi memori berkelanjutan. Saat menggunakan linked list circular, kita memerlukan simpul palsu atau simpul saat ini untuk menghentikan perhitungan saat melintasi struktur lingkaran. Kesimpulannya, kedua jenis linked list memiliki tujuan yang berbeda dan berperan penting dalam pemrograman. Memahami karakteristik dan kasus penggunaannya sangat penting untuk manipulasi data yang efisien dan desain algoritma.

DAFTAR PUSTAKA

Asisten Praktikum. 2024. MODUL 4: LINKED LIST CIRCULAR DAN NON CIRCULAR.