

LAPORAN PRAKTIKUM
MODUL 4
LINKED LIST CIRCULAR DAN NON CIRCULAR



Disusun oleh:
Geranada Saputra Priambudi
NIM : 2311102008

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

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM 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

Linked List artinya node-node tersebut saling terhubung satu sama lain. Double Artinya field pointer-nya dua buah dan dua arah, ke node sebelum dan sesudahnya. Non Circular artinya prev dan next-nya akan menunjuk pada NULL. Jadi, Double Linked List Non Circular (DLLNC) adalah Double Linked List yang memiliki 2 buah pointer yaitu pointer next dan prev. Pointer next menunjuk pada node setelahnya dan pointer prev menunjuk pada node sebelumnya. Setiap node pada linked list mempunyai field yang berisi data dan pointer ke node berikutnya dan ke node sebelumnya. Untuk pembentukan node baru, mulanya pointer next dan prev akan menunjuk ke nilai NULL. Selanjutnya, pointer prev akan menunjuk ke node sebelumnya, dan pointer next akan menunjuk ke node selanjutnya pada list. Penambahan node baru akan dikaitkan di node paling depan, namun pada saat pertama kali (data masih kosong), maka penambahan data dilakukan pada head nya. Pada prinsipnya adalah mengkaitkan data baru dengan head, kemudia head akan menunjuk pada data baru tersebut sehingga head akan tetap selalu menjadi data terdepan. Untuk menghubungkan node terakhir dengan node terdepan dibutuhkan pointer bantu. Penambahan data dilakukan di belakang, namun pada saat pertama kali data langsung ditunjuk pada head-nya. Penambahan di belakang lebih sulit karena kita membutuhkan pointer bantu untuk mengetahui data terbelakang, kemudian dikaitkan dengan data baru.

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;
    }

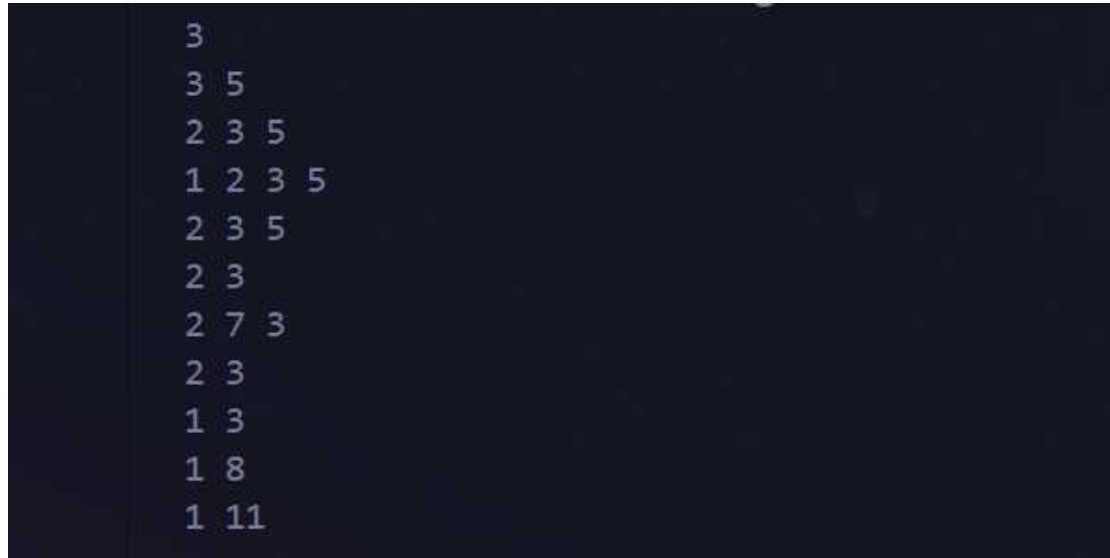
// Tampilkan list
void tampilList()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "Linked list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

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Fungsi insertDepan() adalah menambahkan node baru di depan linked list. Jika list kosong, node baru menjadi head dan tail. Jika tidak, node baru menjadi head dan next dari node baru menjadi node sebelumnya yang menjadi head.

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;
        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()

```

```

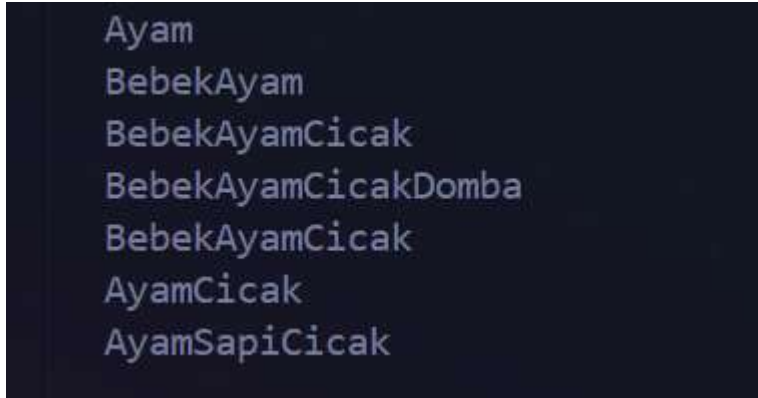
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

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
```

DESKRIPSI PROGRAM

Fungsi buatNode() adalah Fungsi ini membuat node baru dengan data yang diberikan dan menetapkan next dari node baru menjadi NULL.

UNGUIDED

1. UNGUIDED 1

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

SOURCE CODE

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

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

class linkedList {
private:
    Node* head;

public:
    linkedList() {
        head = nullptr;
    }

    void tambahDepan(string nama, string nim) {
        Node* newNode = new Node;
        newNode->nama = nama;
```

```

        newNode->nim = nim;
        newNode->next = head;
        head = newNode;
        cout << "Data telah ditambahkan" << endl;
    }

void tambahBelakang(string nama, string nim) {
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->nim = nim;
    newNode->next = nullptr;
    if (head == nullptr) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next != nullptr) {
        temp = temp->next;
    }
    temp->next = newNode;
    cout << "Data telah ditambahkan" << endl;
}

void tambahTengah(string nama, string nim, int posisi) {
    if (posisi <= 0) {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->nim = nim;
    Node* temp = head;
    for (int i = 0; i < posisi - 1; i++) {
        if (temp == nullptr) {

```



```

        cout << "Posisi tidak valid" << endl;
        return;
    }
    temp = temp->next;
}
if (temp == nullptr) {
    cout << "Posisi tidak valid" << endl;
    return;
}
newNode->next = temp->next;
temp->next = newNode;
cout << "Data telah ditambahkan" << endl;
}

void hapusDepan() {
    if (head == nullptr) {
        cout << "Linked list kosong" << endl;
        return;
    }
    Node* temp = head;
    head = head->next;
    delete temp;
    cout << "Data berhasil dihapus" << endl;
}

void hapusBelakang() {
    if (head == nullptr) {
        cout << "Linked list kosong" << endl;
        return;
    }
    if (head->next == nullptr) {
        delete head;
        head = nullptr;
        cout << "Data berhasil dihapus" << endl;
    }
}

```

```

        return;
    }
    Node* temp = head;
    while (temp->next->next != nullptr) {
        temp = temp->next;
    }
    delete temp->next;
    temp->next = nullptr;
    cout << "Data berhasil dihapus" << endl;
}

void hapusTengah(int posisi) {
    if (posisi <= 0 || head == nullptr) {
        cout << "Linked list kosong atau posisi tidak valid"
<< endl;
        return;
    }
    if (posisi == 1) {
        hapusDepan();
        return;
    }
    Node* temp = head;
    for (int i = 0; i < posisi - 2; i++) {
        if (temp->next == nullptr) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        temp = temp->next;
    }
    if (temp->next == nullptr) {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    Node* nodeToDelete = temp->next;

```

```

        temp->next = temp->next->next;
        delete nodeToDelete;
        cout << "Data berhasil dihapus" << endl;
    }

    void ubahDepan(string namaBaru, string nimBaru) {
        if (head == nullptr) {
            cout << "Linked list kosong" << endl;
            return;
        }
        head->nama = namaBaru;
        head->nim = nimBaru;
        cout << "Data berhasil diubah" << endl;
    }

    void ubahBelakang(string namaBaru, string nimBaru) {
        if (head == nullptr) {
            cout << "Linked list kosong" << endl;
            return;
        }
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->nama = namaBaru;
        temp->nim = nimBaru;
        cout << "Data berhasil diubah" << endl;
    }

    void ubahTengah(string namaBaru, string nimBaru, int posisi)
    {
        if (posisi <= 0 || head == nullptr) {
            cout << "Linked list kosong atau posisi tidak valid"
<< endl;

```

```

        return;
    }
    Node* temp = head;
    for (int i = 0; i < posisi - 1; i++) {
        if (temp == nullptr) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        temp = temp->next;
    }
    if (temp == nullptr) {
        cout << "Posisi tidak valid" << endl;
        return;
    }
    temp->nama = namaBaru;
    temp->nim = nimBaru;
    cout << "Data berhasil diubah" << endl;
}

void hapusList() {
    Node* current = head;
    Node* next;
    while (current != nullptr) {
        next = current->next;
        delete current;
        current = next;
    }
    head = nullptr;
    cout << "Linked list berhasil dihapus" << endl;
}

void tampilkanData() {
    Node* temp = head;
    cout << "DATA MAHASISWA" << endl;

```

```

        cout << "NAMA\tNIM" << endl;
        while (temp != nullptr) {
            cout << temp->nama << "\t" << temp->nim << endl;
            temp = temp->next;
        }
    }
};

int main() {
    linkedList linkedList;
    int option;
    string nama, nim;
    int posisi;

    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 >> option;

        switch (option) {
            case 1:

```

```
        cout << "-Tambah Depan-" << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.tambahDepan(nama, nim);
        break;
    case 2:
        cout << "-Tambah Belakang-" << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.tambahBelakang(nama, nim);
        break;
    case 3:
        cout << "-Tambah Tengah-" << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.tambahTengah(nama, nim, posisi);
        break;
    case 4:
        cout << "-Ubah Depan-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.ubahDepan(nama, nim);
        break;
    case 5:
```

```
        cout << "-Ubah Belakang-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.ubahBelakang(nama, nim);
        break;
    case 6:
        cout << "-Ubah Tengah-" << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.ubahTengah(nama, nim, posisi);
        break;
    case 7:
        linkedList.hapusDepan();
        break;
    case 8:
        linkedList.hapusBelakang();
        break;
    case 9:
        cout << "-Hapus Tengah-" << endl;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.hapusTengah(posisi);
        break;
    case 10:
        linkedList.hapusList();
        break;
    case 11:
        linkedList.tampilkanData();
```

```
        break;
    default:
        cout << "pilihan tidak ada" << endl;
    }
} while (option != 12);

return 0;
}
```


SCREENSHOOT PROGRAM

NAMA	NIM
Jawad	23300001
Geranada	2311102008
Farrel	23300003
Wati	23300004
Denis	23300005
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

Tambah wati tengah

DATA MAHASISWA	
NAMA	NIM
Jawad	23300001
Geranada	2311102008
Farrel	23300003
Wati	23300004
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

Hapus data denis

NAMA	NIM
Owi	2330000
Jawad	23300001
Geranada	2311102008
Farrel	23300003
Wati	23300004
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

Tambah owi

NAMA	NIM
Owi	2330000
Jawad	23300001
Geranada	2311102008
Farrel	23300003
Wati	23300004
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099
David	23300100

Tambah David

NAMA	NIM
Owi	2330000
Jawad	23300001
Geranada	2311102008
Farrel	23300003
Wati	2330004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099
David	23300100

Edit udin menjadi idin

NAMA	NIM
Owi	2330000
Jawad	23300001
Geranada	2311102008
Farrel	23300003
Wati	2330004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099
Lucy	23300101

Ubah data terakhir

NAMA	NIM
Jawad	23300001
Geranada	2311102008
Farrel	23300003
Wati	23300004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099
Lucy	23300101

Hapus depan

DATA MAHASISWA	
NAMA	NIM
Bagas	23300002
Geranada	2311102008
Farrel	23300003
Wati	23300004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099
Lucy	23300101

Ubah depan

NAMA	NIM
Bagas	2330002
Geranada	2311102008
Farrel	23300003
Wati	2330004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099

Hapus akhir

DESKRIPSI PROGRAM

Fungsi main() adalah titik masuk program. Di sini, fungsi-fungsi yang telah didefinisikan sebelumnya dipanggil untuk melakukan operasi pada linked list. Program ini menyediakan menu interaktif untuk memilih operasi yang akan dilakukan pada linked list, seperti menambahkan, mengubah, dan menghapus node, serta menampilkan data mahasiswa yang disimpan dalam list.

BAB IV

KESIMPULAN

Setelah melakukan pembelajaran mengenai tipe data di Bahasa Pemrograman C++ berikut poin utama yang telah dipelajari :

1. Struktur Data: Linked list adalah struktur data yang terdiri dari elemen-elemen yang terhubung secara langsung oleh tautan. Setiap elemen disebut sebagai node, yang berisi data dan referensi ke node berikutnya dalam list.
2. Linked list adalah struktur data yang efisien untuk penambahan dan penghapusan elemen, tetapi akses ke elemen tertentu memerlukan traversal dari awal list.
3. Linked list sirkular menawarkan keuntungan dalam traversal karena tidak memerlukan pengecekan kondisi akhir list, tetapi penambahan dan penghapusan node di tengah list masih memerlukan traversal dari awal list.

DAFTAR PUSTAKA

Susanti. 2013. Single LinkList, <http://susanti.ilearning.me/2013/10/03/single-link-list/>, diakses pada tanggal 14 April 2024.