

LAPORAN PRAKTIKUM

MODUL IV

LINKED LIST DAN NON CIRCULAR



Disusun oleh:

Reli Gita Nurhidayati

NIM: 2311102025

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

Praktikum ini memiliki tujuan, yaitu:

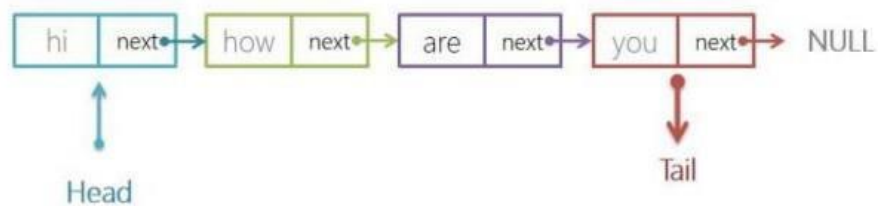
1. Mahasiswa dapat mengetahui dan memahami linked list circular dan non circular.
2. Mahasiswa dapat membuat linked list circular dan non circular.
3. Mahasiswa dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

BAB II

DASAR TEORI

1. Linked List Non Circular

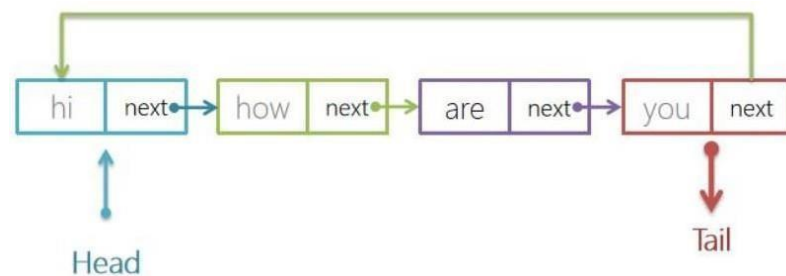
Linked list non circular merupakan *linked list* dengan node pertama (head) dan node terakhir (tail) yang tidak saling terhubung. Pointer terakhir (tail) pada *Linked List* ini selalu bernilai '**NULL**' sebagai pertanda data terakhir dalam *list*-nya. *Linked list non circular* dapat digambarkan sebagai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



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. *Linked list circular* dapat digambarkan sebagai berikut.



Gambar 2 Single Linked List Circular

BAB III

GUIDED

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)
        {
            // This block is empty in the original image
        }
        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;
    }

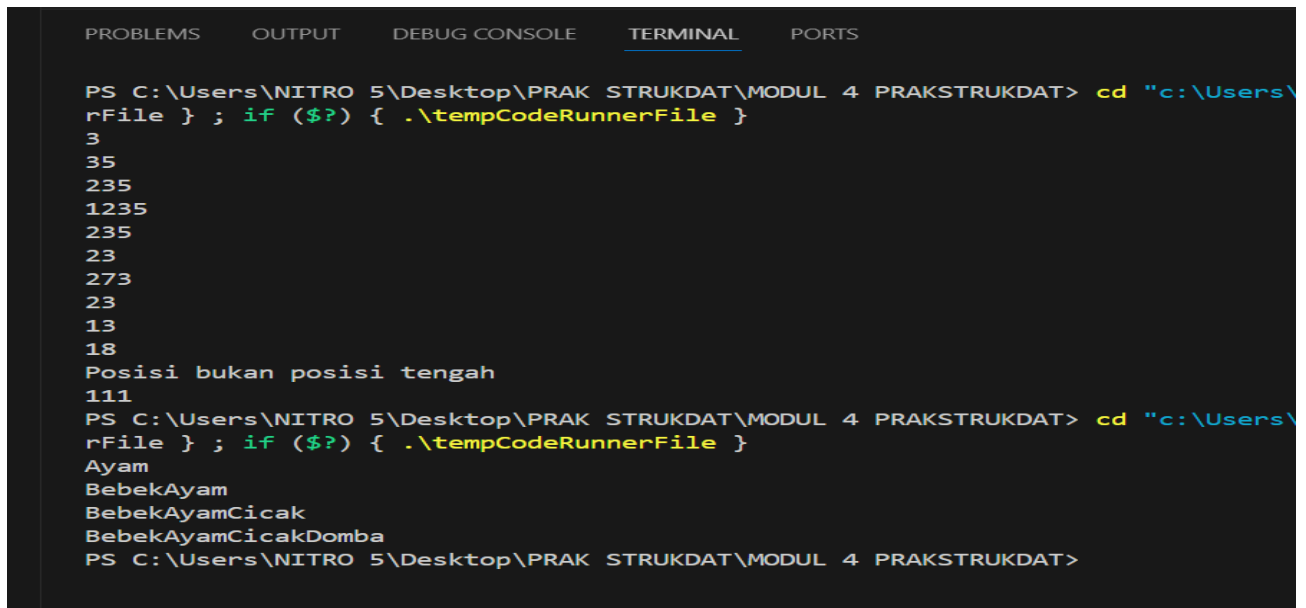
    cout << endl;
}
else
{
    cout << "List masih kosong!" << endl;
}
}

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



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 4 PRAKSTRUKDAT> cd "c:\Users\
rFile } ; if ($?) { .\tempCodeRunnerFile }
3
35
235
1235
235
23
273
23
13
18
Posisi bukan posisi tengah
111
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 4 PRAKSTRUKDAT> cd "c:\Users\
rFile } ; if ($?) { .\tempCodeRunnerFile }
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 4 PRAKSTRUKDAT>
```

Deskripsi Program

Program di atas adalah implementasi dari struktur data Single Linked List non-circular. Dalam program ini, sebuah struktur data `Node` didefinisikan yang memiliki dua anggota: `data` untuk menyimpan nilai integer, dan `next` untuk menunjukkan ke Node berikutnya dalam linked list. Terdapat juga dua pointer global, `head` dan `tail`, yang digunakan untuk menunjukkan awal dan akhir dari linked list. Fungsi-fungsi yang disediakan dalam program ini mencakup inisialisasi linked list (`init`), pengecekan apakah linked list kosong (`isEmpty`), penambahan elemen di depan (`insertDepan`) dan di belakang (`insertBelakang`), perhitungan jumlah elemen dalam linked list (`hitungList`), penambahan elemen di tengah (`insertTengah`), penghapusan elemen di depan (`hapusDepan`), di belakang (`hapusBelakang`), dan di tengah (`hapusTengah`), serta pengubahan nilai elemen di depan (`ubahDepan`), di belakang (`ubahBelakang`), dan di tengah (`ubahTengah`)

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;
    }
}

```

```

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 Depan
}
}
// 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;
        }
    }
}

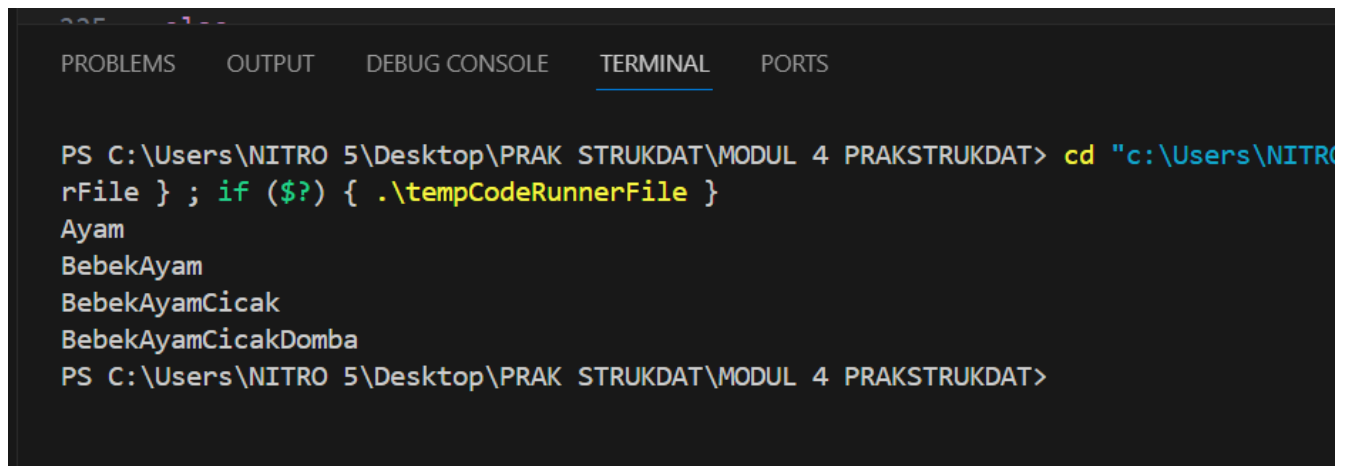
```

```

    }
    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



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 4 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 4 PRAKSTRUKDAT"
rFile } ; if ($?) { .\tempCodeRunnerFile }
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 4 PRAKSTRUKDAT>
```

Deskripsi Program

BAB IV UNGUIDED

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)

- 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**

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 (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;

```

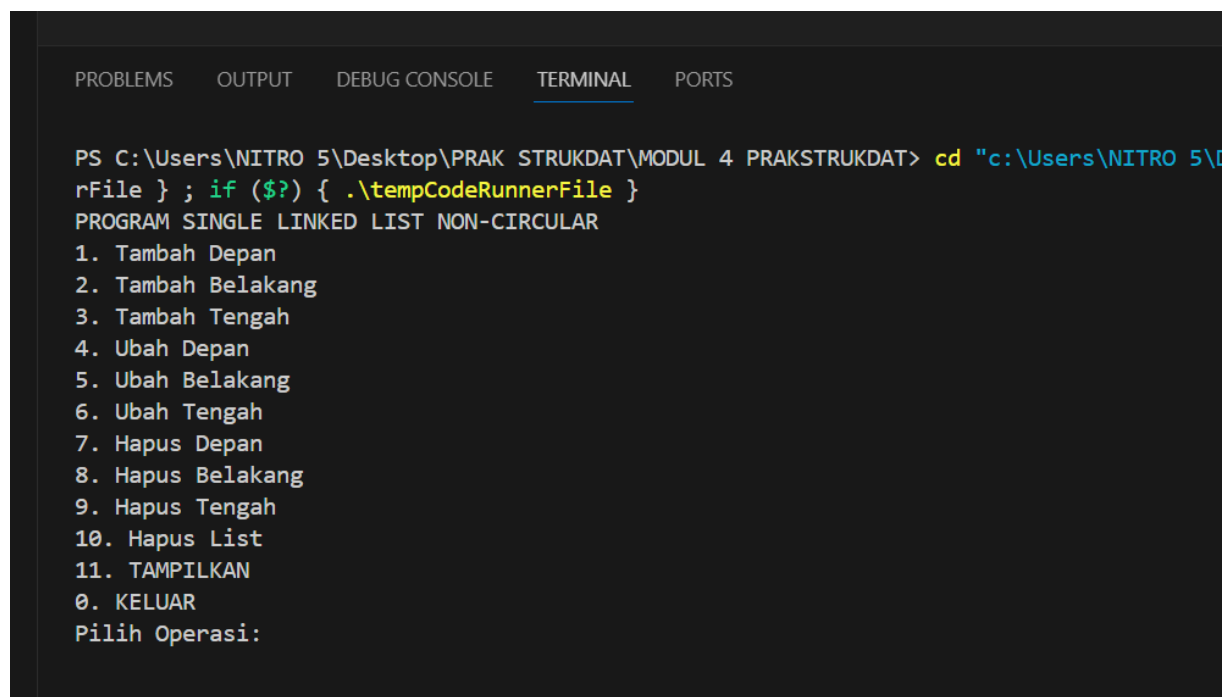
```

        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;
}

```

Screenshoot Program



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 4 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\
rFile } ; if ($?) { .\tempCodeRunnerFile }
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:

```

Deskripsi Program

Program ini adalah implementasi dari struktur data Single Linked List non-circular. Program ini memungkinkan pengguna untuk melakukan sejumlah operasi, seperti menambah, mengubah, dan menghapus data mahasiswa, serta menampilkan isi linked list. Struktur data ini terdiri dari simpul-simpul (Node) yang memiliki dua bagian data, yaitu nama dan NIM mahasiswa, serta sebuah pointer yang menunjukkan ke simpul berikutnya dalam

linked list. Fungsi-fungsi yang disediakan mencakup penambahan data pada bagian depan, belakang, atau posisi tertentu, pengubahan data pada bagian depan, belakang, atau posisi tertentu, penghapusan data pada bagian depan, belakang, atau posisi tertentu, serta fungsi untuk menampilkan isi linked list. Program utama memungkinkan pengguna untuk memilih operasi yang ingin dilakukan terhadap linked list melalui sebuah loop `do-while`. Setiap operasi yang dipilih akan memanggil fungsi yang sesuai dan menampilkan hasilnya. Dengan menggunakan alokasi memori dinamis, program ini memberikan fleksibilitas dalam pengelolaan data mahasiswa secara dinamis sesuai dengan kebutuhan.

B. Tampilan Operasi tambahan

```
Pilih Operasi: 1
-Tambah Depan
Masukkan Nama : Religita
Masukkan NIM : 2311102025
Data telah ditambahkan
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi: 2
-Tambah Belakang
Masukkan Nama: Hidayati
Masukkan NIM: 2311102025
Data telah ditambahkan
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi: 3
-Tambah Tengah
Masukkan Nama : Nur
Masukkan NIM : 2311102025
Masukkan Posisi : 2
Data telah ditambahkan
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

C. Tampilan operasi hapus

```
Pilih Operasi: 8
-Hapus Belakang
Data (nama mahasiswa yang dihapus) berhasil dihapus
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi: 9
-Hapus Tengah
Masukkan posisi : 2
Data (nama mahasiswa yang dihapus) berhasil dihapus
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

D. Tampilan operasi ubah

```
Pilih Operasi: 5
-Ubah Belakang
Masukkan nama : Hidayati
Masukkan NIM : 2311102025
Data (nama lama) telah diganti dengan data (nama baru
)
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

```
Pilih Operasi: 6
Masukkan Nama: Nur
Masukkan NIM: 2311102025
Masukkan posisi: 2
Posisi diluar jangkauan
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

E. Tampilan Operasi Tampil data:

```
Pilih Operasi: 11
=====
DATA MAHASISWA
=====
|  NAMA          |  NIM  |
-----
|  Hidayati      | 2311102025  |
-----
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

2. Unguided 2

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;
}

Node* hapusDepan(Node *head)
{
    if (!isEmpty(head))
    {
        Node *hapus = head;
        head = head->next;
        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;
}

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;

    head = tambahDepan(head, "Jawad", "23300001");
    head = tambahBelakang(head, "Religita", "2311102025");
    head = tambahTengah(head, "Farrel", "23300003", 2);
    head = tambahTengah(head, "Denis", "23300005", 3);
    head = tambahTengah(head, "Anis", "23300008", 4);
    head = tambahBelakang(head, "Bowo", "23300015");
    head = tambahBelakang(head, "Gahar", "23300040");
    head = tambahBelakang(head, "Udin", "23300048");
    head = tambahBelakang(head, "Ucok", "23300050");
    head = tambahBelakang(head, "Budi", "23300099");

    char choice;
    do
    {
        cout << "Menu:" << endl;
        cout << "a) Tambahkan data berikut diantara Farrel dan Denis" << endl;
        cout << "b) Hapus data Denis" << endl;
        cout << "c) Tambahkan data berikut di awal" << endl;
        cout << "d) Tambahkan data berikut di akhir" << endl;
        cout << "e) Ubah data Udin menjadi data berikut" << endl;
        cout << "f) Ubah data terakhir menjadi berikut" << endl;
        cout << "g) Hapus data awal" << endl;
        cout << "h) Ubah data awal menjadi berikut" << endl;
        cout << "i) Hapus data akhir" << endl;
    }
}

```

```

cout << "j) Tampilkan seluruh data" << endl;
cout << "k) Keluar" << endl;
cout << "Pilihan Anda: ";
cin >> choice;

switch (choice)
{
    case 'a':
    {
        string nama, nim;
        cout << "Masukkan nama: ";
        cin >> nama;
        cout << "Masukkan NIM: ";
        cin >> nim;
        head = tambahTengah(head, nama, nim, 3);
        break;
    }
    case 'b':
    {
        head = hapusTengah(head, 4);
        break;
    }
    case 'c':
    {
        string nama, nim;
        cout << "Masukkan nama: ";
        cin >> nama;
        cout << "Masukkan NIM: ";
        cin >> nim;
        head = tambahDepan(head, nama, nim);
        break;
    }
    case 'd':
    {
        string nama, nim;
        cout << "Masukkan nama: ";
        cin >> nama;
        cout << "Masukkan NIM: ";
        cin >> nim;
        head = tambahBelakang(head, nama, nim);
        break;
    }
    case 'e':
    {
        Node *bantu = head;
        while (bantu != nullptr)
        {
            if (bantu->nama == "Udin" && bantu->nim == "23300048")

```

```

        {
            bantu->nama = "Idin";
            bantu->nim = "23300045";
            break;
        }
        bantu = bantu->next;
    }
    break;
}
case 'f':
{
    Node *bantu = head;
    while (bantu->next != nullptr)
    {
        bantu = bantu->next;
    }
    bantu->nama = "Lucy";
    bantu->nim = "23300101";
    break;
}
case 'g':
{
    head = hapusDepan(head);
    break;
}
case 'h':
{
    head->nama = "Bagas";
    head->nim = "2330002";
    break;
}
case 'i':
{
    Node *bantu = head;
    while (bantu->next->next != nullptr)
    {
        bantu = bantu->next;
    }
    delete bantu->next;
    bantu->next = nullptr;
    break;
}
case 'j':
{
    tampil(head);
    break;
}
case 'k':

```

```

        {
            break;
        }
        default:
        {
            cout << "Pilihan tidak valid!" << endl;
        }
    }
} while (choice != 'k');

hapusList(head);

return 0;
}

```

Screenshoot Program

DATA MAHASISWA		
=====		
NAMA	NIM	

jawad	23300001	
reli	2311102025	
farrel	23300003	
denis	23300005	
anis	23300008	
bowo	23300015	
gahar	23300040	
udin	23300040	
ucok	23300050	
budi	23300099	

PROGRAM SINGLE LINKED LIST NON-CIRCULAR		

Deskripsi Program

Program ini merupakan implementasi dari struktur data Single Linked List non-circular. Program ini memungkinkan pengguna untuk melakukan sejumlah operasi,

seperti menambah, mengubah, dan menghapus data mahasiswa, serta menampilkan isi linked list. Struktur data ini terdiri dari simpul-simpul (Node) yang memiliki dua bagian data, yaitu nama dan NIM mahasiswa, serta sebuah pointer yang menunjukkan ke simpul berikutnya dalam linked list. Fungsi-fungsi yang disediakan mencakup penambahan data pada bagian depan, belakang, atau posisi tertentu, pengubahan data pada bagian depan, belakang, atau posisi tertentu, penghapusan data pada bagian depan, belakang, atau posisi tertentu, serta fungsi untuk menampilkan isi linked list. Program utama memungkinkan pengguna untuk memilih operasi yang ingin dilakukan terhadap linked list melalui sebuah loop `do-while`. Setiap operasi yang dipilih akan memanggil fungsi yang sesuai dan menampilkan hasilnya. Dengan menggunakan alokasi memori dinamis, program ini memberikan fleksibilitas dalam pengelolaan data mahasiswa secara dinamis sesuai dengan kebutuhan.

3. Unguided 3

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;
}

```

```

Node* hapusDepan(Node *head)
{
    if (!isEmpty(head))
    {
        Node *hapus = head;
        head = head->next;
        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;
}

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;

    head = tambahDepan(head, "Jawad", "23300001");
    head = tambahBelakang(head, "gita", "2311102025");
    head = tambahTengah(head, "Farrel", "23300003", 2);
    head = tambahTengah(head, "Denis", "23300005", 3);
    head = tambahTengah(head, "Anis", "23300008", 4);
    head = tambahBelakang(head, "Bowo", "23300015");
    head = tambahBelakang(head, "Gahar", "23300040");
    head = tambahBelakang(head, "Udin", "23300048");
    head = tambahBelakang(head, "Ucok", "23300050");

```



```

head = tambahBelakang(head, "Budi", "23300099");

char choice;
do
{
    cout << "Menu:" << endl;
    cout << "a) Tambahkan data berikut diantara Farrel dan Denis" << endl;
    cout << "b) Hapus data Denis" << endl;
    cout << "c) Tambahkan data berikut di awal" << endl;
    cout << "d) Tambahkan data berikut di akhir" << endl;
    cout << "e) Ubah data Udin menjadi data berikut" << endl;
    cout << "f) Ubah data terakhir menjadi berikut" << endl;
    cout << "g) Hapus data awal" << endl;
    cout << "h) Ubah data awal menjadi berikut" << endl;
    cout << "i) Hapus data akhir" << endl;
    cout << "j) Tampilkan seluruh data" << endl;
    cout << "k) Keluar" << endl;
    cout << "Pilihan Anda: ";
    cin >> choice;

    switch (choice)
    {
        case 'a':
        {
            string nama, nim;
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            head = tambahTengah(head, nama, nim, 3);
            break;
        }
        case 'b':
        {
            head = hapusTengah(head, 4);
            break;
        }
        case 'c':
        {
            string nama, nim;
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            head = tambahDepan(head, nama, nim);
            break;
        }
        case 'd':

```

```

{
    string nama, nim;
    cout << "Masukkan nama: ";
    cin >> nama;
    cout << "Masukkan NIM: ";
    cin >> nim;
    head = tambahBelakang(head, nama, nim);
    break;
}
case 'e':
{
    Node *bantu = head;
    while (bantu != nullptr)
    {
        if (bantu->nama == "Udin" && bantu->nim == "23300048")
        {
            bantu->nama = "Idin";
            bantu->nim = "23300045";
            break;
        }
        bantu = bantu->next;
    }
    break;
}
case 'f':
{
    Node *bantu = head;
    while (bantu->next != nullptr)
    {
        bantu = bantu->next;
    }
    bantu->nama = "Lucy";
    bantu->nim = "23300101";
    break;
}
case 'g':
{
    head = hapusDepan(head);
    break;
}
case 'h':
{
    head->nama = "Bagas";
    head->nim = "2330002";
    break;
}
case 'i':
{

```

```

        Node *bantu = head;
        while (bantu->next->next != nullptr)
        {
            bantu = bantu->next;
        }
        delete bantu->next;
        bantu->next = nullptr;
        break;
    }
    case 'j':
    {
        tampil(head);
        break;
    }
    case 'k':
    {
        break;
    }
    default:
    {
        cout << "Pilihan tidak valid!" << endl;
    }
}
} while (choice != 'k');

hapusList(head);

return 0;
}

```

Screenshoot Program

A. Tambahkan data wati

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERM
DATA MAHASISWA
=====
|  NAMA      |  NIM  |
-----
|  Jawad     | 23300001 |
|  Farrel    | 23300003 |
|  wati      | 23300004 |
|  Denis     | 23300005 |
|  Anis      | 23300008 |
|  gita      | 2311102025 |
|  Bowo      | 23300015 |
|  Gahar     | 23300040 |
|  Udin      | 23300048 |
|  Ucok      | 23300050 |
|  Budi      | 23300099 |
-----
Menu:
```

B. Hapus data denis

```
=====
DATA MAHASISWA
=====
|  NAMA      |  NIM  |
-----
|  Jawad     | 23300001 |
|  Farrel    | 23300003 |
|  wati      | 23300004 |
|  Anis      | 23300008 |
|  gita      | 2311102025 |
|  Bowo      | 23300015 |
|  Gahar     | 23300040 |
|  Udin      | 23300048 |
|  Ucok      | 23300050 |
|  Budi      | 23300099 |
-----
Menu:
```

C. Tambah data owi di awal

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

NAMA	NIM
owi	2330000
Jawad	23300001
Farrel	23300003
wati	2330004
Anis	23300008
gita	2311102025
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

```
-----
Menu:
```

D. Tambah data david di akhir

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

NAMA	NIM
owi	2330000
Jawad	23300001
Farrel	23300003
wati	2330004
Anis	23300008
gita	2311102025
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099
david	23300100

E. Ubah data udin menjadi idin

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

NAMA	NIM
owi	2330000
Jawad	23300001
Farrel	23300003
wati	2330004
Anis	23300008
gita	2311102025
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099
in	23300045

```
-----
Menu:
```

F. Ubah data terakhir menjadi lucy

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

NAMA	NIM
y	23300101
owi	2330000
Jawad	23300001
Farrel	23300003
wati	2330004
Anis	23300008
gita	2311102025
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099
Lucy	23300101

```
-----
Menu:
```

G. Hapus data awal

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

NAMA	NIM
owi	2330000
Jawad	23300001
Farrel	23300003
wati	2330004
Anis	23300008
gita	2311102025
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099
Lucy	23300101

```
-----
```

H. Ubah data awal menjadi bagas

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

NAMA	NIM
Bagas	2330002
Jawad	23300001
gas	2330002
Farrel	23300003
Anis	23300008
gita	2311102025
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099
Lucy	23300101

```
-----
Menu:
```

I. Hapus data akhir

```
PROBLEMS OUTPUT DEBUG CONSOLE TERM
Pilihan Anda: j
=====
DATA MAHASISWA
=====
|  NAMA      |  NIM  |
|-----|
|  Bagus     | 2330002 |
|  Jawad     | 23300001 |
|  gas       | 2330002 |
|  Farrel    | 23300003 |
|  Anis      | 23300008 |
|  gita      | 2311102025 |
|  Bowo      | 23300015 |
|  Gahar     | 23300040 |
|  Idin      | 23300045 |
|  Ucok      | 23300050 |
|  Budi      | 23300099 |
|-----|
Menu:
a) Tambahkan data berikut di antara ...
```

J. Tampilkan seluruh data

```
Pilihan Anda: j
=====
DATA MAHASISWA
=====
|  NAMA      |  NIM  |
|-----|
|  Bagus     | 2330002 |
|  Jawad     | 23300001 |
|  gas       | 2330002 |
|  Farrel    | 23300003 |
|  Anis      | 23300008 |
|  gita      | 2311102025 |
|  Bowo      | 23300015 |
|  Gahar     | 23300040 |
|  Idin      | 23300045 |
|  Ucok      | 23300050 |
|  Budi      | 23300099 |
|-----|
Menu:
```

Deskripsi program

Program ini memberikan pengalaman praktis dalam pengelolaan linked list non-circular dan memberikan pemahaman yang lebih baik tentang bagaimana struktur data ini dapat dimanfaatkan untuk mengelola data dengan efisien dalam konteks aplikasi pemrograman.

BAB V

KESIMPULAN

Mempelajari materi linked list non-circular memberikan pemahaman akan struktur data secara mendalam akan pemrograman. Berbeda dengan array yang memiliki keterbatasan dalam ukuran dan fleksibilitas, linked list memungkinkan penyimpanan data secara dinamis dan efisien. Dengan konsep simpul (node) yang saling terhubung, linked list memungkinkan pengelolaan data yang lebih fleksibel.

Dalam linked list non-circular, setiap node terhubung ke node berikutnya secara linear, dan tidak ada simpul yang mengarah kembali ke simpul sebelumnya, sehingga tidak terdapat siklus tertutup. Hal ini memungkinkan operasi-operasi seperti penambahan, penghapusan, dan pencarian data dengan efisiensi yang tinggi.

Memahami linked list non-circular memperluas wawasan tentang bagaimana data disimpan dan diakses dalam struktur non-sequential. Dengan menguasai konsep ini, pengembang dapat mengimplementasikan berbagai algoritma dan struktur data yang lebih kompleks dalam pengembangan perangkat lunak. Kemampuan untuk mengelola data secara fleksibel dan efisien menjadi kunci penting dalam pengoptimalan kinerja aplikasi dan sistem yang dikembangkan.

Selain itu, mempelajari linked list non-circular juga membantu dalam pengembangan keterampilan pemecahan masalah dan pemikiran algoritma. Kemampuan untuk merancang, mengimplementasikan, dan mengelola struktur data yang kompleks seperti linked list memperkaya kemampuan pemrogram dalam menyelesaikan berbagai tantangan dalam pengembangan perangkat lunak.

Dengan demikian, pemahaman yang mendalam tentang linked list non-circular tidak hanya memperluas pengetahuan teknis pemrogram, tetapi juga memperkaya kemampuan analisis, perancangan, dan implementasi solusi perangkat lunak yang efisien dan handal.