

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL IV
LINKED LIST CIRCULAR DAN NON CIRCULAR**



Disusun Oleh :

Anisah Syifa Mustika Riyanto
2311102080

Dosen

Wahyu Andi Saputra, S.Pd., M.Eng.

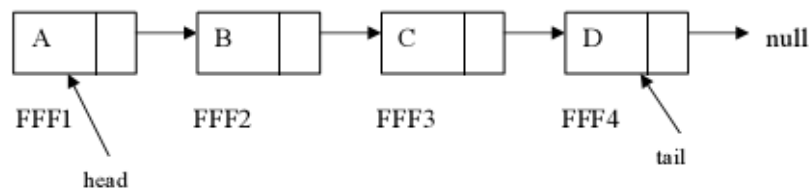
**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. Dasar Teori

Linked List adalah suatu cara untuk menyimpan data dengan struktur sehingga programmer dapat secara otomatis menciptakan suatu tempat baru untuk menyimpan data kapan saja diperlukan. Linked list dikenal juga dengan sebutan senarai berantai adalah stuktur data yang terdiri dari urutan record data dimana setiap record memiliki field yang menyimpan alamat/referensi dari record selanjutnya (dalam urutan). Elemen data yang dihubungkan dengan link pada linked list disebut Node. Biasanya dalam suatu linked list, terdapat istilah head dan tail . 1. Head adalah elemen yang berada pada posisi pertama dalam suatu linked list 2. Tail adalah elemen yang berada pada posisi terakhir dalam suatu linked list. Jenis Linked List (yang akan dipelajari) adalah : 1. Single Linked List 2. Double Linked List

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.

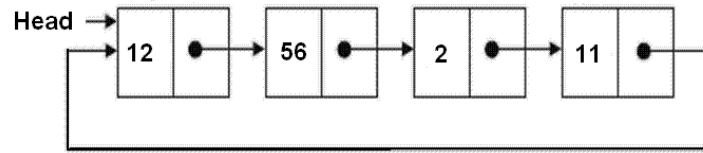


Ilustrasi Linked List Non Circular:

1. Setiap node pada linked list mempunyai field yang berisi data dan berisi pointer ke node berikutnya.
2. Pada akhir linked list node terakhir akan menunjuk ke NULL sehingga linked list berakhir.

Linked List Circular

Circular linked list merupakan sekumpulan node atau simpul yang tidak terdapat nilai NULL pada satupun nodenya. Circular Singly linked list hampir sama dengan single linked list yakni memiliki value dan pointer dalam nodenya. Perbedaannya terletak pada tail next, jika pada single linked list pointer next pada node terakhir (tail) mengarah ke NULL maka pada circular single linked list, pointer mengarah pada head sehingga terjadi susunan elemen yang melingkar. Berikut ini adalah gambaran untuk single circular linked list:



Ilustrasi Circular Single Linked List:

1. Setiap node pada linked list mempunyai field yang berisi data dan berisi pointer ke node berikutnya.
2. Pada akhir linked list node terakhir akan menunjuk ke node terdepan sehingga linked list berputar.

Terdapat beberapa operasi dalam program circular linked list, yaitu:

- Insert/Penambahan: Operasi untuk menyisipkan elemen kedalam list.
- Delete/Penghapusan: Operasi untuk menghapus suatu elemen dalam list.

Dalam implementasinya penambahan dan penghapusan suatu elemen dikembangkan kembali sehingga menghasilkan beberapa variasi dalam penggunaannya, contoh:

- InsertFirst / Penambahan elemen pertama.
- InsertLast / Penambahan elemen terakhir.
- DeleteFirst / Penghapusan elemen pertama.
- DeleteLast / Penghapusan elemen terakhir.

B. Guided

Guided 1 Linked List Non Circular

Source code:

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

// Deklarasi Struct Node
struct Node
{
    int data;
    Node *next;
};
Node *head;
Node *tail;
// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}
```

```

// Pengecekan
bool isEmpty()
{
    if (head == NULL)
        return true;
    else
        return false;
}

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

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

// Hitung Jumlah List
int hitungList()
{
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
    }
}

```

```

        hitung = hitung->next;
    }
    return jumlah;
}
// Tambah Tengah
void insertTengah(int data, int posisi)
{
    if (posisi < 1 || posisi >
        hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;
        // tranversing
        bantu = head;
        int nomor = 1;
        while (nomor < posisi -
            1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru
            ->next =
                bantu->next;
        bantu->next = baru;
    }
}
// Hapus Depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;
            head = head->next;
        }
        else
        {
            {
            }
        }
    }
    else
    {
        delete hapus;
        head = tail =

```

```

        NULL;
        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;
        }
        else
        {
        }
    }
    else
    {
        delete hapus;
        head = tail = NULL;
        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)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {
            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;
}

```

Screenshots Output

```

PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC> & 'c:\Users\hp151\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-h2zrglag.h53' '--stdout=Microsoft-MIEngine-Out-pobxhkzr.nlm' '--stderr=Microsoft-MIEngine-Error-yrf4cazy.ocq' '--pid=Microsoft-MIEngine-Pid-2tnsvs1v.4py' '--dbgExe=C:\Program Files\CodeBlocks\MinGW\bin\gdb.exe' '--interpreter=mi'
3
35
235
1235
235
23
273
23
13
18
111
PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC>

```

Deskripsi:

Program di atas menggunakan linked list non circular. Terdapat beberapa operasi dalam program, yakni tambah, hapus, dan ubah yang telah dikembangkan ke dalam pilihan node depan, node pada posisi tertentu atau node tengah, dan node belakang. Selain itu terdapat pilihan operasi tampilkan dan hapus seluruh list. Data merupakan inputan dari pengguna.

Guided 2 Linked List Circular

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;
}

```

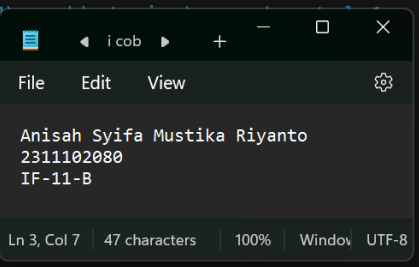
```

}
// Tampilkan List
void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << " ";
            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;
}

```

Screenshots Output

```
PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC> & 'c:\Users\hp157\OneDrive\Desktop\praktikum\19.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-Out-dsuupra0.lgg' '--stderr=Microsoft-MIEngine-Error-ciyfhqk0.wnd
de' '--dbgExe=C:\Program Files\CodeBlocks\MinGW\bin\gdb.exe' '--interpreter=mi
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak
█
```



Deskripsi:

Program di atas adalah contoh implementasi dari struktur data linked list non-circular dalam bahasa C++. Dalam program, terdapat struct node, variable global, fungsi inisialisasi linked list, fungsi untuk memeriksa list kosong, membuat node baru, hingga memanipulasi node dengan menambah, mengganti, atau menghapus list.

C. 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

-Hapus Belakang

Data (nama mahasiswa yang dihapus) berhasil dihapus

-Hapus Tengah

Masukkan posisi :

Data (nama mahasiswa yang dihapus) berhasil dihapus

• Tampilan Operasi Ubah:

-Ubah Belakang

Masukkan nama :
Masukkan NIM :

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

-Ubah Belakang

Masukkan nama :
Masukkan NIM :
Masukkan posisi :

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

• Tampilan Operasi Tampil Data:

DATA MAHASISWA

NAMA NIM
Nama1 NIM1
Nama2 NIM2

Source code:

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

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

class LinkedList {
private:
    identitasMhs* head;

public:
    LinkedList() {
        head = NULL;
    }

    void addFirst(string nama, string nim) {
        identitasMhs* newNode = new identitasMhs();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = head;
        head = newNode;
        cout << endl << "Data telah ditambahkan"<<endl;
    }

    void addLast(string nama, string nim) {
        identitasMhs* newNode = new identitasMhs();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = NULL;
        if (head == NULL) {
            head = newNode;
        } else {
            identitasMhs* temp = head;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
        cout << endl << "Data telah ditambahkan"<<endl;
    }

    void addMiddle(string nama, string nim, int posisi) {
        if (posisi < 1) {
            cout << endl << "Posisi tidak valid"<<endl;
            return;
        }
        identitasMhs* newNode = new identitasMhs();
```



```

newNode->nama = nama;
newNode->nim = nim;
if (posisi == 1) {
    newNode->next = head;
    head = newNode;
} else {
    identitasMhs* temp = head;
    for (int i = 1; i < posisi - 1; i++) {
        if (temp == NULL) {
            cout << endl << "Posisi tidak valid"<<endl;
            return;
        }
        temp = temp->next;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}
cout << endl << "Data telah ditambahkan"<<endl;
}

void changeFirst(string nama, string nim) {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    head->nama = nama;
    head->nim = nim;
    cout << endl << "Data di depan telah diubah"<<endl;
}

void changeLast(string nama, string nim) {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    identitasMhs* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->nama = nama;
    temp->nim = nim;
    cout << endl << "Data di belakang telah diubah"<<endl;
}

void changeMiddle(string nama, string nim, int posisi) {
    if (posisi < 1) {
        cout << endl << "Posisi tidak valid"<<endl;
        return;
    }
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    identitasMhs* temp = head;
    for (int i = 1; i < posisi; i++) {

```

```

        if (temp == NULL) {
            cout << endl << "Posisi tidak valid"<<endl;
            return;
        }
        temp = temp->next;
    }
    temp->nama = nama;
    temp->nim = nim;
    cout << endl << "Data di posisi " << posisi << " telah diubah"<<endl;
}

void deleteFirst() {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    identitasMhs* temp = head;
    head = head->next;
    delete temp;
    cout << endl << "Data di depan telah dihapus"<<endl;
}

void deleteLast() {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    if (head->next == NULL) {
        delete head;
        head = NULL;
        cout << endl << "Data di belakang telah dihapus"<<endl;
        return;
    }
    identitasMhs* temp = head;
    while (temp->next->next != NULL) {
        temp = temp->next;
    }
    delete temp->next;
    temp->next = NULL;
    cout << endl << "Data di belakang telah dihapus"<<endl;
}

void deleteMiddle(int posisi) {
    if (posisi < 1) {
        cout << endl << "Posisi tidak valid"<<endl;
        return;
    }
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    identitasMhs* temp = head;
    if (posisi == 1) {
        head = head->next;
        delete temp;
    }

```

```

        cout << endl << "Data di posisi " << posisi << " telah dihapus"<<endl;
        return;
    }
    for (int i = 1; i < posisi - 1; i++) {
        if (temp == NULL || temp->next == NULL) {
            cout << endl << "Posisi tidak valid"<<endl;
            return;
        }
        temp = temp->next;
    }
    identitasMhs* nextNode = temp->next->next;
    delete temp->next;
    temp->next = nextNode;
    cout << endl << "Data di posisi " << posisi << " telah dihapus"<<endl;
}

void print() {
    if (head == NULL) {
        cout << endl << "Linked List kosong"<<endl;
        return;
    }
    cout << "=====DATA MAHASISWA===== "<< endl;
    cout << "-----" << endl;
    cout << setw(20) << left << "NAMA" << setw(20) << left << "NIM" << endl;
    identitasMhs* temp = head;
    while (temp != NULL) {
        cout << setw(20) << left << temp->nama << setw(20) << left << temp->nim
<< endl;
        temp = temp->next;
    }
}

void deleteAll() {
    while (head != NULL) {
        identitasMhs* temp = head;
        head = head->next;
        delete temp;
    }
    cout << endl << "Semua data mahasiswa telah dihapus"<<endl;
}

};

int main() {
    LinkedList linkedList;
    int pilihan;
    string nama, nim;
    int posisi;
    while (true) {
        cout << endl;
        linkedList.print();
        cout << endl;
        cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR"<<endl;
        cout << "===== " << endl;
        cout << 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 << endl;

cout << "Pilih Operasi: ";
cin >> pilihan;

cout << endl;

switch (pilihan) {
    case 1:
        cout << "-Tambah Depan"<<endl;
        cout << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.addFirst(nama, nim);
        break;
    case 2:
        cout << "-Tambah Belakang"<<endl;
        cout << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        linkedList.addLast(nama, nim);
        break;
    case 3:
        cout << "-Tambah Tengah"<<endl;
        cout << endl;
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.addMiddle(nama, nim, posisi);
        break;
    case 4:
        cout << "-Ubah Depan"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";

```

```

        cin >> nim;
        linkedList.changeFirst(nama, nim);
        break;
    case 5:
        cout << "-Ubah Belakang"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        linkedList.changeLast(nama, nim);
        break;
    case 6:
        cout << "-Ubah Tengah"<<endl;
        cout << endl;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.changeMiddle(nama, nim, posisi);
        break;
    case 7:
        cout << "-Hapus Depan"<<endl;
        cout << endl;
        linkedList.deleteFirst();
        break;
    case 8:
        cout << "-Hapus Belakang"<<endl;
        cout << endl;
        linkedList.deleteLast();
        break;
    case 9:
        cout << "-Hapus Tengah"<<endl;
        cout << endl;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        linkedList.deleteMiddle(posisi);
        break;
    case 10:
        cout << "-Hapus List"<<endl;
        cout << endl;
        linkedList.deleteAll();
        break;
    case 11:
        cout << "-Tampilkan"<<endl;
        cout << endl;
        linkedList.print();
        break;
    case 0:
        exit(0);
    default:
        cout << "Pilihan tidak valid"<<endl;
}

```

```

    }

    return 0;
}

```

Screenshots Output

```

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: 1

-Tambah Depan

Masukkan Nama : Anisah
Masukkan NIM : 2311102080

Data telah ditambahkan

=====DATA MAHASISWA=====
-----
NAMA          NIM
Anisah        2311102080

```

```

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: 4

-Ubah Depan

Masukkan Nama Baru : Syifa
Masukkan NIM Baru : 2311102080

Data di depan telah diubah

=====DATA MAHASISWA=====
-----
NAMA          NIM
Syifa         2311102080

```

```
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: 7

-Hapus Depan

Data di depan telah dihapus

Linked List kosong
```

```
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: 0

PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC\Modul 4>
```

```
Anisah Syifa Mustika Riyanto
2311102080
IF-11-B
```

Deskripsi:

Program menggunakan linked list non circular untuk mendata identitas mahasiswa secara terurut. Terdapat 11 operasi yang terdapat pada program, yakni tambah, hapus, dan ubah yang telah dikembangkan ke dalam pilihan node depan, node pada posisi tertentu atau node tengah, dan node belakang. Selain itu terdapat pilihan operasi tampilkan dan keluar. Data merupakan inputan dari pengguna.

- 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

Screenshots Output

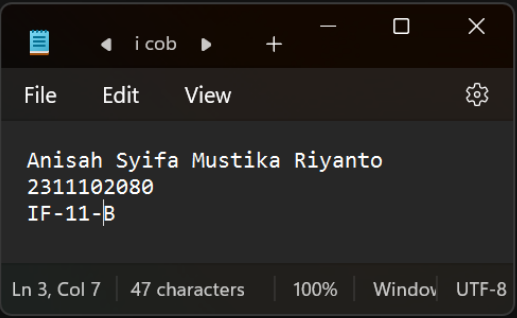
```
=====DATA MAHASISWA=====
-----
NAMA          NIM
Jawad          23300001
Anisah          2311101080
Farrel          23300003
Denis          23200005
Anis            23300008
Bowo            23300015
Bahar           23300040
Udin            23300048
Ucok            23300050
Budi            23300099

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: 0

PS D:\Huru Hara Semester 2\Praktikum Strukdat\Praktikum VSC\Modul 4> |
```



Deskripsi:

Program menggunakan 2 pilihan operasi, yaitu operasi tambah depan untuk menambah data pertama, dan operasi no 2 untuk menambah data ke-2 hingga terakhir. Pilihan menu tampil tidak perlu digunakan karena saya memrogram agar tampilan otomatis keluar ketika data selesai diinput.

3. Lakukan perintah berikut:

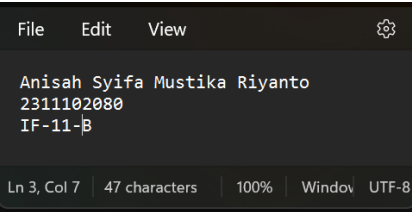
a) Tambahkan data berikut diantara Farrel dan Denis:

Wati 23300004

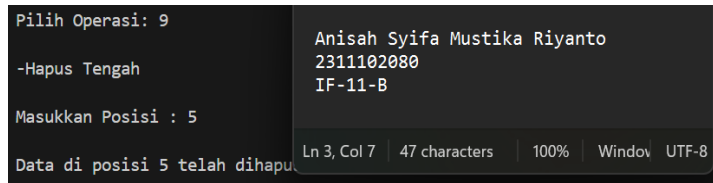
```
Pilih Operasi: 3
-Tambah Tengah

Masukkan Nama : Wati
Masukkan NIM : 23300004
Masukkan Posisi : 4

Data telah ditambahkan
```

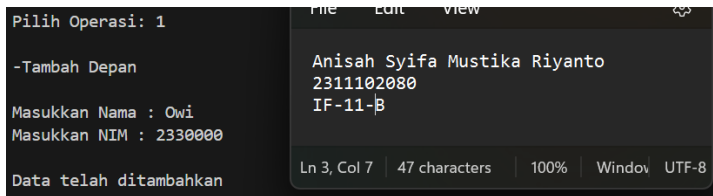


b) Hapus data Denis



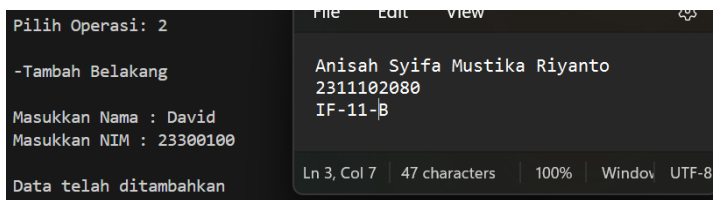
c) Tambahkan data berikut di awal:

Owi 2330000



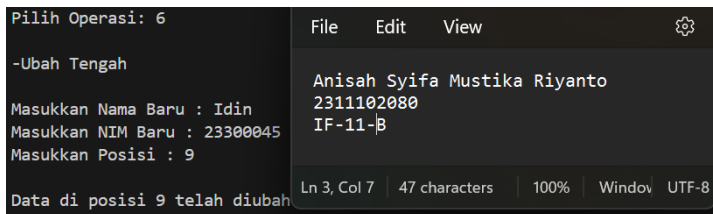
d) Tambahkan data berikut di akhir:

David 23300100



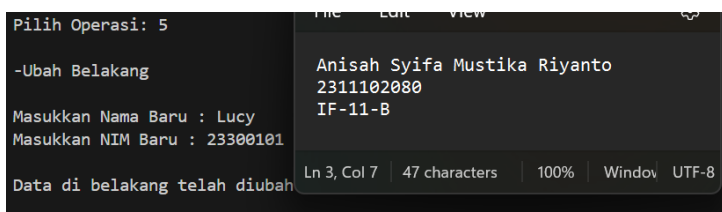
e) Ubah data Udin menjadi data berikut:

Idin 23300045

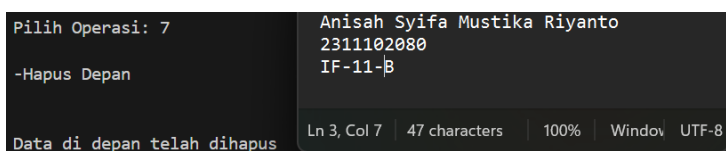


f) Ubah data terakhir menjadi berikut:

Lucy 23300101



g) Hapus data awal



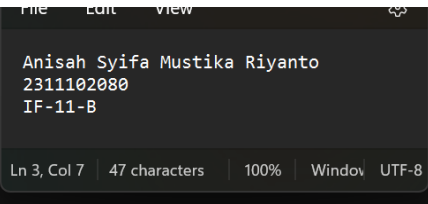
h) Ubah data awal menjadi berikut:

Bagas 2330002

```
Pilih Operasi: 4
-Ubah Depan

Masukkan Nama Baru : Bagas
Masukkan NIM Baru : 2330002

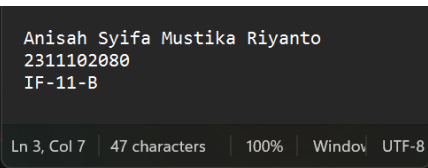
Data di depan telah diubah
```



i) Hapus data akhir

```
Pilih Operasi: 8
-Hapus Belakang

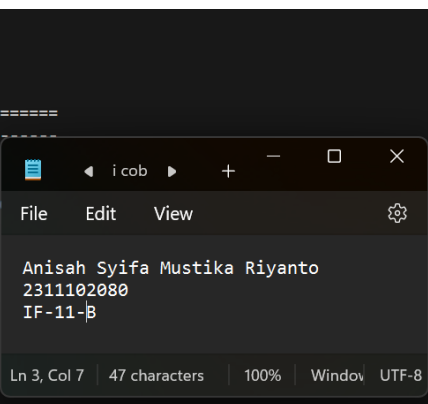
Data di belakang telah dihapus
```



j) Tampilkan seluruh data

```
Pilih Operasi: 11
-Tampilkan

=====DATA MAHASISWA=====
-----
NAMA      NIM
Bagas     23300002
Anisah    231110108
Farrel    23300005
Wati      23300004
Anis      23300008
Bowo      23300015
Gahar     23300040
Idin      23300045
Ucok      23300050
Budi      23300099
```



Deskripsi:

Tampilan di atas merupakan hasil dari keseluruhan pilihan operasi yang telah digunakan.

D. Kesimpulan

Terdapat banyak jenis linked list yang bisa digunakan sesuai fungsi dan kebutuhan program. Pada praktikum ini linked list yang telah dipelajari adalah Linked list Non Circular dan Circular Linked List. Keduanya hampir mirip, masing- masing memiliki node yang saling terhubung dengan tiap node memiliki field untuk menyimpan data dan pointer. Selain itu terdapat node yang menjadi head pada awal list dan tail pada akhir list. Perbedaannya terdapat pada bagian tail. Pointer tail pada linked list circular menunjuk ke head sehingga linked list berputar, sedangkan pointer pada linked list non circular menunjuk ke NULL sehingga linked list berhenti.

E. Referensi

Geeksforgeeks. Introduction to Circular Linked List.(2024) Diakses 6 April 2024 dari <https://www.geeksforgeeks.org/circular-linked-list/>

Malik, D. S. (2023). *C++ programming*. Cengage Learning, EMEA.

Repository.dinus. Single Linked List Non Circular. Diakses pada 6 April 2024 dari https://repository.dinus.ac.id/docs/ajar/Pertemuan_11_Single_Linked_List_Circular.pdf

Stroustrup, B. (2022). *A Tour of C++*. Addison-Wesley Professional.

Ulum, M. Bahrul. Modul Kuliah Struktur Data. Linked List. Diakses pada 6 April 2024 dari https://lmsparalel.esaunggul.ac.id/pluginfile.php?file=%2F86227%2Fmod_resource%2Fcontent%2F1%2FModul%20Struktur%20Data-Linked%20List.pdf