

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

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



**Disusun Oleh :
WIDARI DWI HAYATI
2311102060**

**Dosen :
WAHYU ANDI SAPUTRA, S.Pd., M.Eng.**

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

A. Dasar Teori

a. **Linked List Non Circular**

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

OPERASI PADA LINKED LIST NON CIRCULAR

1. Deklarasi Simpul (Node)

```
struct Node {  
    int data;  
    Node *next;  
};
```

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
Node *head;  
Node *tail;
```

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

3. Pengecekan Kondisi Linked List

```
bool isEmpty()  
{  
    if (head == NULL && tail == NULL) {  
        return true;  
    }  
    else  
    {  
        return false;  
    }  
}
```

4. Penambahan Simpul (Node)

```
void insertBelakang(string dataUser) {  
    if (isEmpty() == true)  
    {  
        node *baru = new node;  
        baru->data = dataUser;  
        head = baru;  
        tail = baru;  
        baru->next = NULL;  
    }  
    else  
    {  
        node *baru = new node;  
        baru->data = dataUser;
```

```

        baru->next = NULL;
        tail->next = baru;
        tail = baru;
    }
};

```

5. Penghapusan Simpul (Node)

```

void hapusDepan()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl; }
    else
    {
        node *helper;
        helper = head;
        if (head == tail)
        {
            head = NULL;
            tail = NULL;
            delete helper;
        }
        else
            head = head->next;
        helper->next = NULL;
        delete helper;
    }
}

```

6. Tampil Data Linked List

```

void tampil()
{
    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    else
    {
        node *helper;
        helper = head;
        while (helper != NULL)

```

```

        {
            cout << helper->data << ends;
            helper = helper->next;
        }
    }
}

```

b. Linked List Circular

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

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

OPERASI PADA LINKED LIST CIRCULAR

1. Deklarasi Simpul (Node)

```

struct Node
{
    string data;
    Node *next;
};

```

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```

Node *head, *tail, *baru, *bantu, *hapus;
void init()
{
    head = NULL;
    tail = head;
}

```

3. Pengecekan Kondisi Linked List

```

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

```

4. Pembuatan Simpul (Node)

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

5. Penambahan Simpul (Node)

```
// 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;
    }
}
```

6. Penghapusan Simpul (Node)

```
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;
    }
}

```

7. Menampilkan Data Linked List

```

void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
}

```

B. Guided

Guided 1

Linked List Non Circular

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

struct Node {
    int data;
    Node *next;
};

Node *head;
Node *tail;

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

bool isEmpty() {
    return head == NULL;
}

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

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

int hitungList() {
    Node *hitung = head;
```

```

    int jumlah = 0;
    while (hitung != NULL) {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

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 = new Node();
        baru->data = data;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

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

void hapusBelakang() {
    if (!isEmpty()) {
        Node *hapus = tail;
        if (head != tail) {
            Node *bantu = head;
            while (bantu->next != tail) {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
        }
    }
}

```



```

    } else {
        head = tail = NULL;
    }
    delete hapus;
} else {
    cout << "List kosong!" << endl;
}
}

void hapusTengah(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 *bantu = head;
        Node *hapus;
        Node *sebelum = NULL;
        int nomor = 1;
        while (nomor < posisi) {
            sebelum = bantu;
            bantu = bantu->next;
            nomor++;
        }
        hapus = bantu;
        if (sebelum != NULL) {
            sebelum->next = bantu->next;
        } else {
            head = bantu->next;
        }
        delete hapus;
    }
}

void ubahDepan(int data) {
    if (!isEmpty()) {
        head->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

void ubahTengah(int data, int posisi) {
    if (!isEmpty()) {
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi di luar jangkauan" << endl;
        } else if (posisi == 1) {
            cout << "Posisi bukan posisi tengah" << endl;
        } else {
            Node *bantu = head;
            int nomor = 1;

```

```

        while (nomor < posisi) {
            bantu = bantu->next;
            nomor++;
        }
        bantu->data = data;
    }
} else {
    cout << "List masih kosong!" << endl;
}
}

void ubahBelakang(int data) {
    if (!isEmpty()) {
        tail->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

void clearList() {
    Node *bantu = head;
    Node *hapus;
    while (bantu != NULL) {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

void tampil() {
    Node *bantu = head;
    if (!isEmpty()) {
        while (bantu != NULL) {
            cout << bantu->data << " ";
            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

```

3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS C:\Users\ADVAN>

```

File Edit View

Nama : Widari Dwi Hayati
 NIM : 2311102060
 Kelas : IF-11-B

Ln 1, Col 1 | 60 characters | 100% | Window UTF-8

Deskripsi:

Program ini menggunakan Single Linked List. Di dalamnya terdapat beberapa fungsi, seperti `init()` untuk menginisialisasi head dan tail, `isEmpty()` untuk mengecek apakah Linked List kosong, `insertDepan()` dan `insertBelakang()` untuk menambahkan node di depan dan di belakang, `hitungList()` untuk menghitung jumlah node, `insertTengah()` untuk menambahkan node di posisi tertentu, `hapusDepan()` dan `hapusBelakang()` untuk menghapus node di depan dan di belakang, `hapusTengah()` untuk menghapus node di posisi tertentu, serta `ubahDepan()`, `ubahTengah()`, dan `ubahBelakang()` untuk mengubah nilai data node di depan, tengah, dan belakang. Selain itu, terdapat juga fungsi `clearList()` untuk menghapus semua node, dan `tampil()` untuk menampilkan semua node yang ada.

Guided 2

Linked List Circular

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

struct Node {
    string data;
    Node *next;
};

Node *head, *tail, *baru, *bantu, *hapus;

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

int isEmpty() {
    return head == NULL;
}

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

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

void insertDepan(string data) {
    buatNode(data);
    if (isEmpty()) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        while (tail->next != head) {
            tail = tail->next;
        }
        tail->next = baru;
    }
}
```

```

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

void insertBelakang(string data) {
    buatNode(data);
    if (isEmpty()) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        tail->next = baru;
        tail = baru;
        tail->next = head;
    }
}

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

void hapusDepan() {
    if (!isEmpty()) {
        if (head == tail) {
            delete head;
            head = NULL;
            tail = NULL;
        } else {
            hapus = head;
            head = head->next;

```

```

        tail->next = head;
        delete hapus;
    }
} else {
    cout << "List masih kosong!" << endl;
}
}

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

void hapusTengah(int posisi) {
    if (!isEmpty()) {
        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;
    }
}

void clearList() {
    if (head != NULL) {

```

```

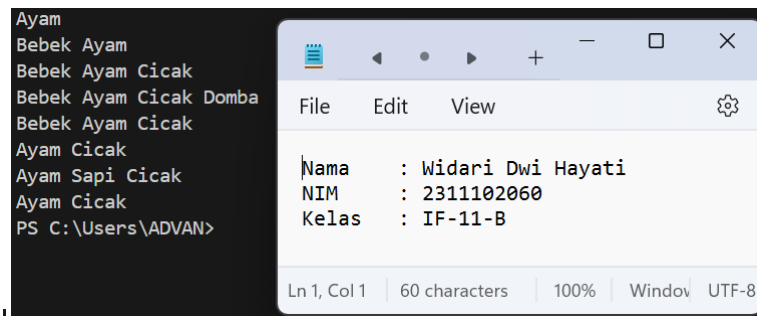
        bantu = head;
        while (bantu != NULL) {
            hapus = bantu;
            bantu = bantu->next;
            delete hapus;
        }
        head = NULL;
        tail = NULL;
    }
    cout << "List berhasil terhapus!" << endl;
}

void tampil() {
    if (!isEmpty()) {
        bantu = head;
        do {
            cout << bantu->data << " ";
            bantu = bantu->next;
        } while (bantu != 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



The screenshot shows a Java IDE with two windows. The left window displays the output of a program, which is a linked list containing the following elements: Ayam, Bebek Ayam, Bebek Ayam Cicak, Bebek Ayam Cicak Domba, Bebek Ayam Cicak, Ayam Cicak, Ayam Sapi Cicak, and Ayam Cicak. The right window is a text editor showing the following text: Nama : Widari Dwi Hayati, NIM : 2311102060, and Kelas : IF-11-B. The text editor also shows the status bar with the text: Ln 1, Col 1, 60 characters, 100%, Window, UTF-8.

Deskripsi:

Program ini menggunakan struktur data linked list. Program ini memiliki beberapa fungsi dasar seperti inialisasi linked list, menambahkan node di awal, akhir, atau tengah linked list, menghapus node pertama, terakhir, atau tengah linked list, menghitung jumlah node pada linked list, dan menampilkan semua data pada linked list. Program ini menggunakan beberapa variabel global seperti head, tail, baru, bantu, dan hapus. Selain itu, program ini juga menggunakan beberapa fungsi lain seperti buatNode, isEmpty, hitungList, insertDepan, insertBelakang, insertTengah, hapusDepan, hapusBelakang, hapusTengah, clearList, dan tampil. Program utama akan menginisialisasi linked list, menambahkan beberapa node, menghapus beberapa node, dan menampilkan data pada linked list.

C. Unguided/Tugas

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:

- Tampilan Menu:

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. TAMPILKAN
0. KELUAR
Pilih Operasi :
```

- Tampilan Operasi Tambah:

-Tambah Depan

Masukkan Nama :
Masukkan NIM :

Data telah ditambahkan

-Tambah Tengah

Masukkan Nama :
Masukkan NIM :
Masukkan Posisi :

Data telah ditambahkan

- Tampilan Operasi Hapus:

-Hapus Belakang

Data (nama mahasiswa yang dihapus) berhasil dihapus

-Hapus Tengah

Masukkan posisi :

Data (nama mahasiswa yang dihapus) berhasil dihapus

- Tampilan Operasi Ubah:

-Ubah Belakang

Masukkan nama :

Masukkan NIM :

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

-Ubah Tengah

Masukkan nama :

Masukkan NIM :

Masukkan posisi :

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

- Tampilan Operasi Tampil Data:

DATA MAHASISWA
NAMA NIM
Nama1 NIM1
Nama2 NIM2

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

- 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

- Lakukan perintah berikut:

- Tambahkan data berikut diantara Farrel dan Denis:

Wati 23300004

- Hapus data Denis

- Tambahkan data berikut di awal:

Owi 23300000

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 23300002

i) Hapus data akhir

j) Tampilkan seluruh data

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

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

Node *head;
Node *tail;

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

bool isEmpty() {
    return head == NULL;
}

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

```

        baru->next = NULL;
        if (isEmpty()) {
            head = tail = baru;
        } else {
            baru->next = head;
            head = baru;
        }
    }
}

void insertBelakang(string nama, string nim) {
    Node *baru = new Node;
    baru->nama = nama;
    baru->nim = nim;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        tail->next = baru;
        tail = baru;
    }
}

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

void insertTengah(string nama, string nim, 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 = new Node();
        baru->nama = nama;
        baru->nim = nim;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
    }
}

```

```

        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan() {
    if (!isEmpty()) {
        string namaLama = head->nama;
        Node *hapus = head;
        if (head->next != NULL) {
            head = head->next;
        } else {
            head = tail = NULL;
        }
        delete hapus;
        cout << "Data (" << namaLama << ") berhasil dihapus" << endl;
    } else {
        cout << "List kosong!" << endl;
    }
    cout << endl;
}

void hapusBelakang() {
    if (!isEmpty()) {
        string namaLama = tail->nama;
        Node *hapus = tail;
        if (head != tail) {
            Node *bantu = head;
            while (bantu->next != tail) {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
        } else {
            head = tail = NULL;
        }
        delete hapus;
        cout << "Data (" << namaLama << ") berhasil dihapus" << endl;
    } else {
        cout << "List kosong!" << endl;
    }
    cout << endl;
}

void hapusTengah(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 *bantu = head;
        Node *hapus;
        Node *sebelum = NULL;
        int nomor = 1;
        while (nomor < posisi) {
            sebelum = bantu;
            bantu = bantu->next;
            nomor++;
        }
        hapus = bantu;
        if (sebelum != NULL) {
            sebelum->next = bantu->next;
        } else {
            head = bantu->next;
        }
        string namaLama = bantu->nama;
        delete hapus;
        cout << "Data (" << namaLama << ") berhasil dihapus" << endl;
    }
    cout << endl;
}

void ubahDepan(string nama, string nim) {
    if (!isEmpty()) {
        string namaLama = head->nama;
        head->nama = nama;
        head->nim = nim;
        cout << "\nData (" << namaLama << ") telah diganti dengan data (" << head-
>nama << ")" << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
    cout << endl;
}

void ubahTengah(string nama, string nim, int posisi) {
    if (!isEmpty()) {
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi di luar jangkauan" << endl;
        } else if (posisi == 1) {
            cout << "Posisi bukan posisi tengah" << endl;
        } else {
            Node *bantu = head;
            int nomor = 1;

```

```

        while (nomor < posisi) {
            bantu = bantu->next;
            nomor++;
        }
        string namaLama = bantu->nama;
        bantu->nama = nama;
        bantu->nim = nim;
        cout << "\nData (" << namaLama << ") telah diganti dengan data (" <<
bantu->nama << ")" << endl;
    }
} else {
    cout << "List masih kosong!" << endl;
}
cout << endl;
}

```

```

void ubahBelakang(string nama, string nim) {
    if (!isEmpty()) {
        string namaLama = tail->nama;
        tail->nama = nama;
        tail->nim = nim;
        cout << "\nData (" << namaLama << ") telah diganti dengan data (" << tail-
>nama << ")" << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
    cout << endl;
}

```

```

void clearList() {
    Node *bantu = head;
    Node *hapus;
    while (bantu != NULL) {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

```

```

void tampil() {
    Node *bantu = head;
    cout << "_____ \n";
    cout << "| " << setw(10) << "Nama " << " | " << setw(10) << "NIM " << " |\n";
    cout << "_____ \n";
}

```



```

        if (!isEmpty()) {
            while (bantu != NULL) {
                cout << "|" << setw(10) << bantu->nama << "|" << setw(10) << bantu-
>nim << "|" << endl;
                bantu = bantu->next;
                cout << "_____ \n";
            }
            cout << endl;
        } else {
            cout << "List masih kosong!" << endl;
        }
    }
}

```

```

int main() {
    init();
    string nama, nim;
    int posisi;
    insertDepan("Jawad", "23300001");
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan NIM : ";
    cin >> nim;
    insertBelakang(nama, nim);
    insertBelakang("Farrel", "23300003");
    insertBelakang("Denis", "23300005");
    insertBelakang("Anis", "23300008");
    insertBelakang("Gahar", "23300040");
    insertBelakang("Udin", "23300048");
    insertBelakang("Ucok", "23300050");
    insertBelakang("Budi", "23300099");
    insertTengah("Bowo", "23300015", 6);
    tampil();
    int pilih;
    cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << endl;
    cout << "\n1. Tambah Depan";
    cout << "\n2. Tambah Belakang";
    cout << "\n3. Tambah Tengah";
    cout << "\n4. Ubah Depan";
    cout << "\n5. Ubah Belakang";
    cout << "\n6. Ubah Tengah";
    cout << "\n7. Hapus Depan";
    cout << "\n8. Hapus Belakang";
    cout << "\n9. Hapus Tengah";
    cout << "\n10. Hapus List";
    cout << "\n11. TAMPILKAN";
    cout << "\n0. KELUAR" << endl;
    cout << "\nPilih Operasi : " << endl;
}

```

```

cin >> pilih;
switch (pilih)
{
    case 1:
        cout << "==Tambah Depan==" << endl;
        cout << "\nMasukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        insertDepan(nama, nim);
        cout << "\nData telah ditambahkan\n" << endl;
        break;
    case 2:
        cout << "==Tambah Belakang==" << endl;
        cout << "\nMasukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        insertBelakang(nama, nim);
        cout << "\nData telah ditambahkan\n" << endl;
        break;
    case 3:
        cout << "==Tambah Tengah==" << endl;
        cout << "\nMasukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        insertTengah(nama, nim, posisi);
        cout << "\nData telah ditambahkan\n" << endl;
        break;
    case 4:
        cout << "==Ubah Depan==" << endl;
        cout << "\nMasukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        ubahDepan(nama, nim);
        break;
    case 5:
        cout << "==Ubah Belakang==" << endl;
        cout << "\nMasukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        ubahBelakang(nama, nim);

```

```

        break;
    case 6:
        cout << "=="Ubah Tengah==" << endl;
        cout << "\nMasukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan posisi : ";
        cin >> posisi;
        ubahTengah(nama, nim, posisi);
        break;
    case 7:
        cout << "=="Hapus Depan==" << endl;
        hapusDepan();
        break;
    case 8:
        cout << "=="Hapus Belakang==" << endl;
        hapusBelakang();
        break;
    case 9:
        cout << "=="Hapus Tengah==" << endl;
        cout << "\nMasukkan posisi : ";
        cin >> posisi;
        hapusTengah(posisi);
        break;
    case 10:
        cout << "=="Hapus List==" << endl;
        clearList();
        cout << "Semua data berhasil dihapus" << endl;
        break;
    case 11:
        tampil();
        break;
    case 0:
        return 0;
        break;
    default:
        cout << "Pilihan Tidak Valid" << endl;
        break;
}

while (pilih != 0)
{
    cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << endl;
    cout << "\n1. Tambah Depan";
    cout << "\n2. Tambah Belakang";
    cout << "\n3. Tambah Tengah";

```

```

cout << "\n4. Ubah Depan";
cout << "\n5. Ubah Belakang";
cout << "\n6. Ubah Tengah";
cout << "\n7. Hapus Depan";
cout << "\n8. Hapus Belakang";
cout << "\n9. Hapus Tengah";
cout << "\n10. Hapus List";
cout << "\n11. TAMPILKAN";
cout << "\n0. KELUAR" << endl;
cout << "\nPilih Operasi : ";
cin >> pilih;
switch (pilih)
{
    case 1:
        cout << "==Tambah Depan==" << endl;
        cout << "\nMasukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        insertDepan(nama, nim);
        cout << "\nData telah ditambahkan" << endl;
        break;
    case 2:
        cout << "==Tambah Belakang==" << endl;
        cout << "\nMasukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        insertBelakang(nama, nim);
        cout << "\nData telah ditambahkan" << endl;
        break;
    case 3:
        cout << "==Tambah Tengah==" << endl;
        cout << "\nMasukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";
        cin >> nim;
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        insertTengah(nama, nim, posisi);
        cout << "\nData telah ditambahkan" << endl;
        break;
    case 4:
        cout << "==Ubah Depan==" << endl;
        cout << "\nMasukkan Nama : ";
        cin >> nama;
        cout << "Masukkan NIM : ";

```

```

        cin >> nim;
        ubahDepan(nama, nim);
        break;
case 5:
    cout << "==Ubah Belakang==" << endl;
    cout << "\nMasukkan Nama : ";
    cin >> nama;
    cout << "Masukkan NIM : ";
    cin >> nim;
    ubahBelakang(nama, nim);
    break;
case 6:
    cout << "==Ubah Tengah==" << endl;
    cout << "\nMasukkan Nama : ";
    cin >> nama;
    cout << "Masukkan NIM : ";
    cin >> nim;
    cout << "Masukkan posisi : ";
    cin >> posisi;
    ubahTengah(nama, nim, posisi);
    break;
case 7:
    cout << "==Hapus Depan==" << endl;
    hapusDepan();
    break;
case 8:
    cout << "==Hapus Belakang==" << endl;
    hapusBelakang();
    break;
case 9:
    cout << "==Hapus Tengah==" << endl;
    cout << "\nMasukkan posisi : ";
    cin >> posisi;
    hapusTengah(posisi);
    break;
case 10:
    cout << "==Hapus List==" << endl;
    clearList();
    cout << "Semua data berhasil dihapus" << endl;
    break;
case 11:
    tampil();
    break;
case 0:
    return 0;
    break;
default:

```

```

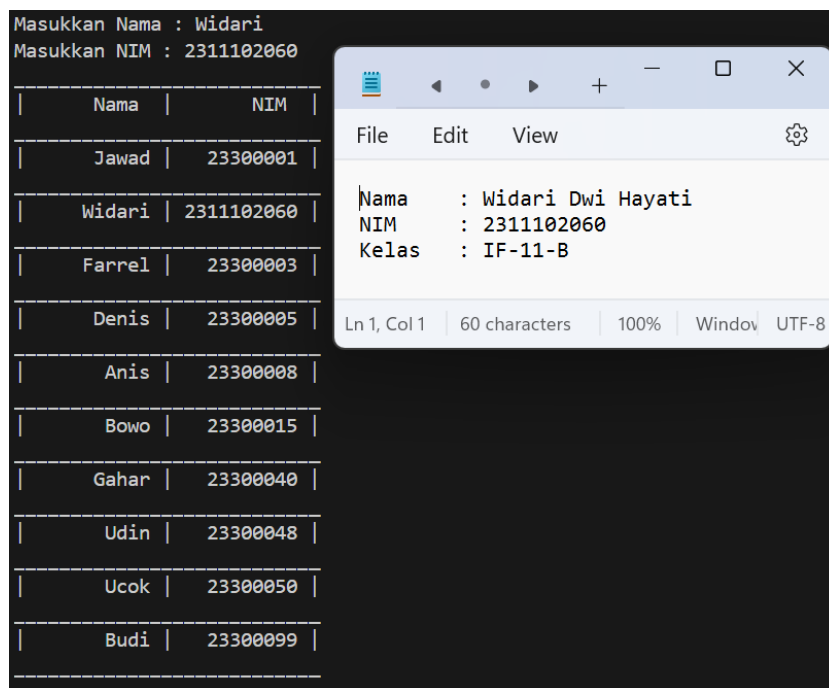
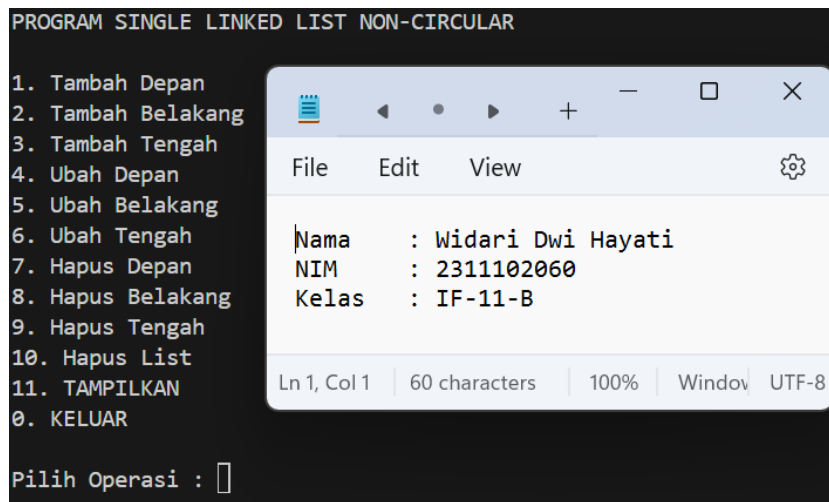
        cout << "Pilihan Tidak Valid" << endl;
        break;
    }

}

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 : 3

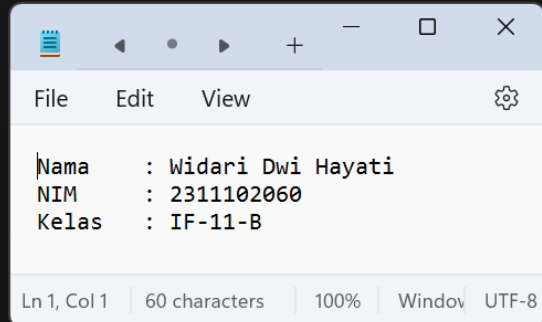
==Tambah Tengah==

Masukkan Nama : Wati

Masukkan NIM : 2330004

Masukkan Posisi : 4

Data telah ditambahkan



PROGRAM SINGLE LINKED LIST NON-CIRCULAR

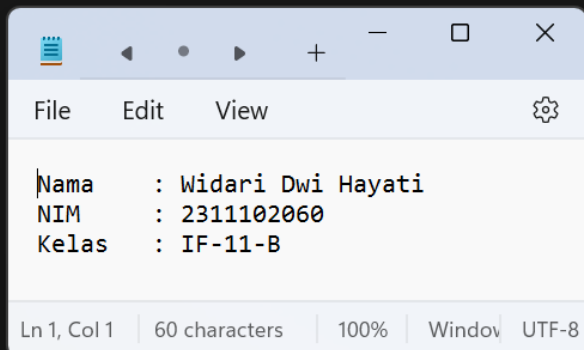
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. TAMPILKAN
0. KELUAR

Pilih Operasi : 9

==Hapus Tengah==

Masukkan posisi : 5

Data (Denis) berhasil dihapus



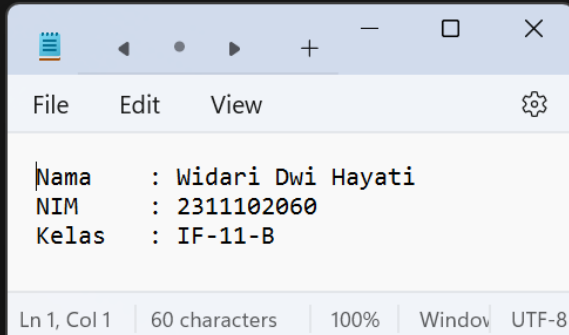
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

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 : Owi
Masukkan NIM : 2330000

Data telah ditambahkan



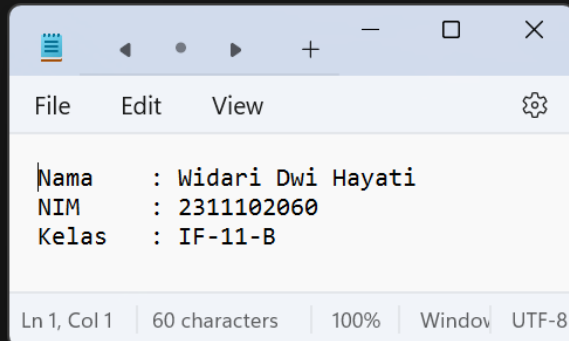
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

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 : 2
==Tambah Belakang==

Masukkan Nama : David
Masukkan NIM : 23300100

Data telah ditambahkan



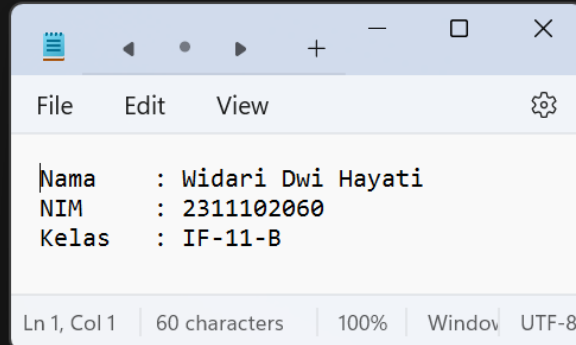
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

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 : 6
==Ubah Tengah==

Masukkan Nama : Idin
Masukkan NIM : 23300045
Masukkan posisi : 9

Data (Udin) telah diganti dengan data (Idin)



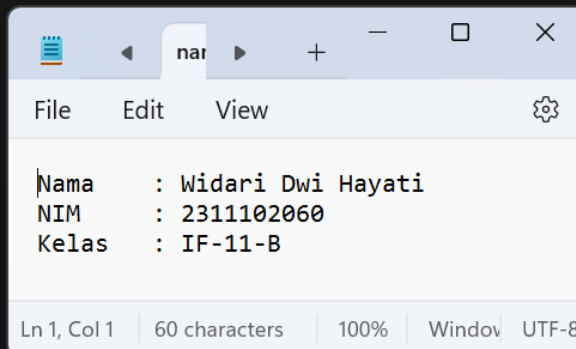
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

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 : 5
==Ubah Belakang==

Masukkan Nama : Lucy
Masukkan NIM : 23300101

Data (David) telah diganti dengan data (Lucy)



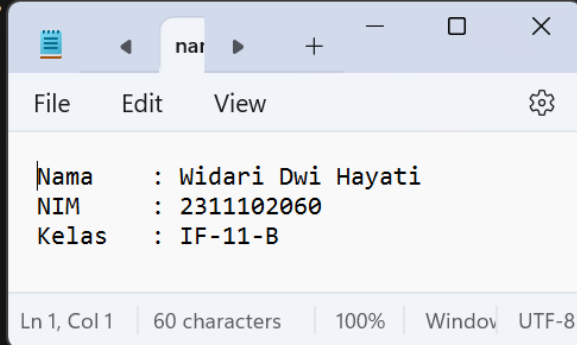
PROGRAM SINGLE LINKED LIST NON-CIRCULAR

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 (Owi) berhasil dihapus



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

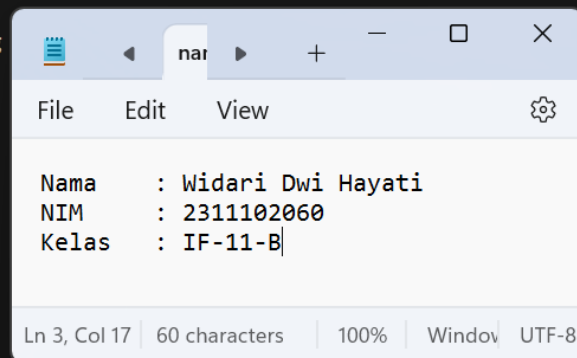
Pilih Operasi : 4

==Ubah Depan==

Masukkan Nama : Bagas

Masukkan NIM : 2330002

Data (Jawad) telah diganti dengan data (Bagas)



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 : 8

==Hapus Belakang==

Data (Lucy) berhasil dihapus

nar

File Edit View

Nama : Widari Dwi Hayati

NIM : 2311102060

Kelas : IF-11-B

Ln 3, Col 17 | 60 characters | 100% | Window UTF-8

10. Hapus List

11. TAMPILKAN

0. KELUAR

Pilih Operasi : 11

Nama	NIM
Bagas	2330002
Widari	2311102060
Farrel	23300003
Wati	2330004
Anis	23300008
Bowo	23300015
Gahar	23300040
Idin	23300045
Ucok	23300050
Budi	23300099

nar

File Edit View

Nama : Widari Dwi Hayati

NIM : 2311102060

Kelas : IF-11-B

Ln 3, Col 17 | 60 characters | 100% | Window UTF-8

Deskripsi:

Program ini menggunakan singly linked list dalam bahasa C++. Program ini memiliki struktur Node yang berisi dua string, yaitu nama dan nim, dan juga pointer ke node berikutnya, next. Linked list ini memiliki dua variabel global, yaitu head dan tail, yang awalnya diatur ke NULL oleh fungsi init().

Program ini menyediakan beberapa fungsi untuk menyisipkan node pada awal (insertDepan()), akhir (insertBelakang()), dan tengah (insertTengah()) dari list. Selain itu, program ini juga memiliki fungsi untuk menghapus node dari awal (hapusDepan()), akhir (hapusBelakang()), dan tengah (hapusTengah()) dari list. Program dapat juga menampilkan isi dari list (tampil()), menghitung jumlah node (hitungList()), dan membersihkan seluruh list (clearList()).

Fungsi main() akan mempersiapkan list, menambah beberapa node, dan kemudian menyediakan menu untuk pengguna berkomunikasi dengan list. Pengguna dapat menambah, menghapus, menampilkan isi dari list, dan membersihkan list. Program akan terus berjalan hingga pengguna memilih untuk keluar.

D. Kesimpulan

Ada dua jenis linked list, yaitu linked list non circular dan linked list circular. Linked list non circular memiliki node pertama (head) dan node terakhir (tail) yang tidak saling terhubung, serta pointer terakhir (tail) selalu bernilai 'NULL'. Linked list circular tidak memiliki akhir karena node terakhir (tail) tidak bernilai 'NULL', tetapi terhubung dengan node pertama (head). Program yang menggunakan single linked list memiliki struktur Node yang berisi dua string, yaitu nama dan nim, dan pointer ke node berikutnya, next.

Linked list ini memiliki dua variabel global, yaitu head dan tail, yang awalnya diatur ke NULL oleh fungsi init(). Program ini menyediakan beberapa fungsi untuk menambahkan node pada awal, akhir, dan tengah dari list, serta fungsi untuk menghapus node dari awal, akhir, dan tengah dari list. Selain itu, program ini juga memiliki fungsi untuk mengubah nilai data node di depan, tengah, dan belakang, serta fungsi untuk menghitung jumlah node dan menampilkan semua data pada linked list. Program utama akan mempersiapkan list, menambah beberapa node, dan kemudian menyediakan menu untuk pengguna berkomunikasi dengan list. Pengguna dapat menambah, menghapus, menampilkan isi dari list, dan membersihkan list. Program akan terus berjalan hingga pengguna memilih untuk keluar.

E. Referensi (APA)

Stroustrup, B. (2021). A Tour of C++ (2nd ed.). Addison-Wesley Professional.

Lippman, S. L., Lajoie, J., & Moo, B. (2020). C++ Primer (6th ed.). Addison-Wesley Professional.

Malik, D. S. (2014). C++ Programming: From Problem Analysis to Program Design (7th ed.). Cengage Learning.