

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL 4
“LINKED LIST CIRCULAR DAN NON CIRCULAR”**



**DISUSUN OLEH :
RIZKY PERLINTA SEMBIRNG
2311102061**

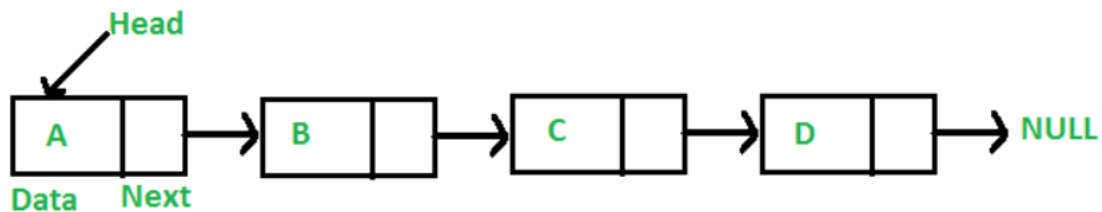
**DOSEN
WAHYU ANDI SAPUTRA, S.PD., M.PD.**

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

A. DASAR TEORI

1. Pengertian Linked List

Linked list adalah struktur data linier berbentuk rantai simpul di mana setiap simpul menyimpan 2 item, yaitu nilai data dan pointer ke simpul elemen berikutnya. Berbeda dengan array, elemen linked list tidak ditempatkan dalam alamat memori yang berdekatan melainkan elemen ditautkan menggunakan pointer.



Simpul pertama dari linked list disebut sebagai head atau simpul kepala. Apabila linked list berisi elemen kosong, maka nilai pointer dari head menunjuk ke NULL. Begitu juga untuk pointer berikutnya dari simpul terakhir atau simpul ekor akan menunjuk ke NULL.

Ukuran elemen dari linked list dapat bertambah secara dinamis dan mudah untuk menyisipkan dan menghapus elemen karena tidak seperti array, kita hanya perlu mengubah pointer elemen sebelumnya dan elemen berikutnya untuk menyisipkan atau menghapus elemen.

Jenis Jenis Linked List

Secara umum, linked list dapat dibagi ke dalam 4 jenis, yakni: Singly linked list, Doubly linked list, Circular linked list, dan Circular doubly linked list. Tetapi di dalam dasar teori ini kita hanya membahas Single dan Double list.

- **Singly linked list**

Single Linked List adalah salah satu struktur data yang paling mendasar dan penting dalam ilmu komputer dan pemrograman. Ini adalah kumpulan simpul yang terhubung, di mana setiap simpul memiliki dua bagian: data dan pointer ke simpul berikutnya dalam urutan. Ini adalah struktur data linier yang secara dinamis dapat tumbuh dan menyusut saat elemen-elemennya ditambahkan atau dihapus. Jadi, kita hanya dapat melintasinya dalam satu arah, yaitu dari simpul kepala ke simpul ekor.



- **Double List**

Double Linked List adalah struktur data yang mirip dengan Single Linked List, namun dengan tambahan pointer tambahan yang menunjuk ke simpul sebelumnya, sehingga memungkinkan navigasi maju dan mundur. Dalam Double Linked List, setiap simpul memiliki tiga bagian: data, pointer ke simpul sebelumnya, dan pointer ke simpul berikutnya.



2. Pengertian Linked List Non Circular

Single Linked List Non Circular adalah Node terakhir akan menunjuk ke NULL yang akan digunakan sebagai kondisi berhenti pada saat pembacaan isi linked list.

BENTUK NODE SINGLE LINKLED LIST NON CIRCULAR

Single : field pointer-nya hanya satu arah, pada akhir not pointer-nya menunjuk NULL

Linked List : Node - node tersebut saling terhubung satu sama lain

setiap node pada linked list mempunyai field yang berisi pointer ke node berikutnya, dan juga memiliki field yang berisi data. Mode terakhir akan menunjuk NULL yang akandigunakan sebagai kondisi berhenti pada saat pembacaan isi linked list.

SINGLE LINKED LIST NON CIRCULAR MENGGUNAKAN HEAD

Dibutuhkan satu buah variabel pointer : head yang akan selaku menunjuk pada nodepertama

Deklarasi Pointer Penunjuk Head Single Linked List sebagai berikut :

TNode*head

- Menambah Node Di Depan

Penambahan node baru akan dikaitkan di node paling depan, namun pada saat pertama kali (data masih kosong), maka penambahan data dilakukan dengan cara : node head ditunjukkan ke node baru tersebut. Prinsipnya adalah mengkaitkan node baru dengan head, kemudian head akan menunjuk pada data baru tersebut sehingga head akan tetap selalu menjadi data terdepan.

- Menambah Node Di Belakang

Penambahan dilakukan di belakang, namun pada saat pertama kali, node langsung ditunjuk oleh head, membutuhkan pointer bantu untuk mengetahui node terbelakang kemudian dikaitkan dengan node baru, perlu digunakan perulangan.

- Menghapus Node Di Depan

Penghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penggunaan suatu pointer lain (hapus) yang digunakan untuk menunjuk node yang akan dihapus, barulah kemudian menghapus pointer menggunakan perintah delete. Sebelum data terdepan terhapus, terlebih dahulu head harus menunjuk ke alamat berikutnya agar list tidak putus, jika head masih NULL berarti data masih kosong.

- Menghapus Node Di Belakang

Membutuhkan pointer bantu dan hapus. Pointer hapus digunakan untuk menunjuk node yang akan dihapus, Pointer bantu untuk menunjuk node sebelum node yang akan dihapus yang akan menjadi node yang terakhir. Pointer bantu digunakan untuk menunjuk ke nilai NULL selalu bergerak sampai sebelum node yang akan dihapus kemudian pointer hapus diletakkan setelah pointer bantu. Selanjutnya pointer hapus akan menunjuk ke NULL.

SINGLE LINKED LIST NON CIRCULAR MENGGUNAKAN HEAD DAN TAIL

Dibutuhkan dua variabel pointer head dan tail. Head selalu menunjuk ke node pertama, sedangkan tail selalu menunjuk ke node terakhir

Kelebihan dari single linked list dengan head dan tail adalah pada penambahan data di belakang, hanya dibutuhkan tail yang mengikat node baru saja tanpa harus menggunakan perulangan pointer bantu.

- Menghapus Node Di Depan (Dengan Head dan Tail)

Penghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penunjukan terlebih dahulu dengan pointer hapus pada head, kemudian dilakukan pergeseran head ke node berikutnya sehingga data setelah head menjadi head baru, kemudian menghapus pointer hapus dengan menggunakan perintah delete. Jika tail masih NULL maka berarti list masih kosong.

- Menghapus Node Di Belakang(Dengan Head Dan Tail)

Penghapusan tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penunjukan terlebih dahulu dengan menggunakan variabel hapus pada tail. Jika Tail masih NULL maka berarti list masih kosong. Dibutuhkan pointer bantu untuk membantu pergeseran dari head ke node berikutnya sampai sebelum tail, sehingga tail dapat ditunjukkan ke bantu, dan bantu tersebut akan menjadi tail yang baru. Setelah itu hapus pointer hapus dengan menggunakan menggunakan perintah delete.

B. Guided

Guided 1 : Linked List Non Circular

Source Code :

```
#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;
}

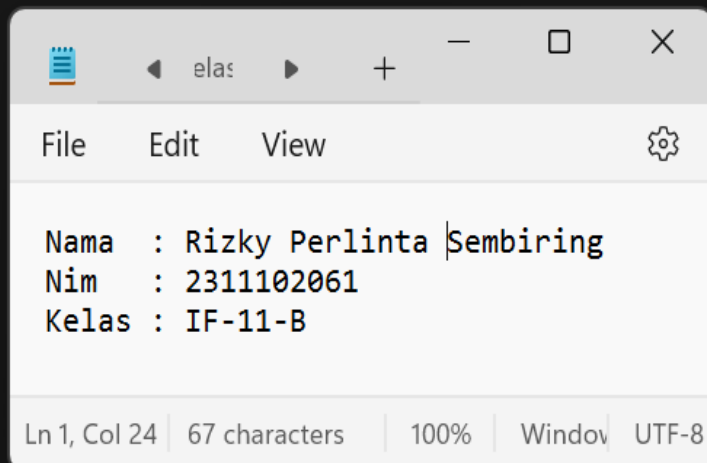
```

Screenshot 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 E:\Semester 2\Pratikum struktur data\guided>

```

Deskripsi Program: Kode tersebut merupakan Linked List non-Circular. Struktur data ini terdiri dari simpul-simpul berurutan dengan setiap simpul(node) menyimpan nilai integer dan pointer ke simpul berikutnya. Fungsi-fungsi yang disediakan meliputi inisialisasi, pengecekan kekosongan, penambahan simpul di depan atau di belakang, penambahan simpul di tengah berdasarkan posisi, penghapusan simpul di depan, di belakang, atau di tengah, pengubahan nilai data simpul di depan, di belakang, atau di tengah, penghapusan seluruh isi Linked List, dan tampilan isi Linked List. Ini memungkinkan penyimpanan dan manipulasi data dengan fleksibilitas dalam program.

Guided 2: Linked List Circular

Source Code:

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

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

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

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

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

void insertBelakang(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;
    }
}

void insertTengah(string data, int posisi) {
    if (isEmpty()) {
        head = baru;
        tail = head;
        baru->next = head;
    } else {
        baru->data = 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()) {
        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;
    }
}

void hapusBelakang() {
    if (!isEmpty()) {
        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;
    }
}

```

```

}

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

void tampil() {
    if (!isEmpty()) {
        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;
}

```

Screenshot Output:

The screenshot shows the output of a C++ program in a terminal window. The output is as follows:

```

Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak
PS E:\Semester 2\Pratikum struktur data\guided>

```

Overlaid on the terminal is a Notepad++ window titled "elas". The window contains the following text:

```

Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11-B

```

The Notepad++ window also shows a menu bar with "File", "Edit", and "View", and a status bar at the bottom indicating "Ln 1, Col 24 | 67 characters | 100% | Window | UTF-8".

Deskripsi Program: Kode tersebut merupakan Linked List Circular, di mana setiap simpul (node) menyimpan data string dan memiliki pointer yang menunjuk ke simpul berikutnya dalam lingkaran. Fungsi-fungsi yang disediakan mencakup inisialisasi, pengecekan kekosongan, pembuatan simpul baru, penambahan di depan, di belakang, atau di tengah, penghapusan di depan, di belakang, atau di tengah, penghapusan seluruh isi, dan tampilan isi Linked List. Ini memungkinkan penggunaan struktur data fleksibel untuk penyimpanan dan manipulasi data dalam program.

C. UNGUIDED

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:

- Tampilkan 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 :
```

- Tampilkan Operasi Tambah:

```
-Tambah Depan

Masukkan Nama :
Masukkan NIM :

Data telah ditambahkan
```

```
-Tambah Tengah

Masukkan Nama :
Masukkan NIM :
Masukkan Posisi :

Data telah ditambahkan
```

- Tampilkan Operasi Hapus

-Hapus Belakang

Data (nama mahasiswa yang dihapus) berhasil dihapus

-Hapus Tengah

Masukkan posisi :

Data (nama mahasiswa yang dihapus) berhasil dihapus

- Tampilkan 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)

- Tampilkan Operasi Tampil Data:

DATA MAHASISWA

NAMA NIM

Nama1 NIM1

Nama2 NIM2

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

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

Nama	NIM
Jawad	23300001
[Nama Anda]	[NIM Anda]
Farrel	23300003
Denis	23300005
Anis	23300008
Bowo	23300015
Gahar	23300040
Udin	23300048
Ucok	23300050
Budi	23300099

3. Lakukan perintah berikut:

- a) Tambahkan data berikut diantara Farrel dan Denis:

Wati 23300004

- b) Hapus data Denis

- c) 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 terkahir menjadi berikut:

Lucy 23300101

- g) Hapus data awal

h) Ubah data awal menjadi berikut:

Bagas 2330002

i) Hapus data akhir

j) Tampilkan seluruh data

Source Code:

```
#include <iostream>
#include <iomanip>
using namespace std;
//Membuat struct dengan variabel mahasiswa terdapat 2 field
struct mahasiswa
{
string nama;//Field nama berØpe data string
string nim;//Field nim berØpe data integer
};
//Struct node terdapat 2 field
struct node
{
mahasiswa ITTP;// field ITTP berØpe inisialisasi struct mahasiswa
node *next;// Field next berØpe pointer node
};
//inisialisasi
node *head, *tail, *bantu, *hapus, *before, *baru;
//Menginisialisasikan nilai head & tail dengan nilai NULL
void init()
{
head = NULL;
tail = NULL;
}
//Pengecekan Nilai
bool isEmpty()
{
{
if (head == NULL)
{

return true;
}
else
{
return false;
}
}
}
mahasiswa Pendataan()
{
mahasiswa ITTP;
```

```

cout << "\nMasukkan Nama\t: ";
cin.ignore();
getline(cin, ITTP.nama);
cout << "Masukkan NIM\t: ";
cin >> ITTP.nim;
return ITTP;
}
// Fungsi Tambah depan
void insertDepan(mahasiswa ITTP)
{
node *baru = new node;
baru->ITTP.nama = ITTP.nama;
baru->ITTP.nim = ITTP.nim;
baru->next = NULL;
if (isEmpty() == true)
{
head = tail = baru;
tail->next = NULL;
}

else
{
baru->next = head;
head = baru;
}
cout << "Data " << ITTP.nama << " berhasil diinput!\n";

}
//Fungsi Tambah Belakang
void insertBelakang(mahasiswa ITTP)
{
node *baru = new node;
baru->ITTP.nama = ITTP.nama;
baru->ITTP.nim = ITTP.nim;
baru->next = NULL;
if (isEmpty() == true)
{
head = tail = baru;
tail->next = NULL;
}
else
{
tail->next = baru;
tail = baru;
}
}
//Hitung List
int hitungList()
{

```

```

int penghitung = 0;

node *bantu;
bantu = head;
while (bantu != NULL)
{
    penghitung++;
    bantu = bantu->next;
}
return penghitung;
}

//Fungsi Tambah tengah
void insertTengah(mahasiswa iden0tas, int posisi)
{
    node *baru = new node;
    baru->ITTP.nama = iden0tas.nama;
    baru->ITTP.nim = iden0tas.nim;
    node *bantu;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "posisi diluar jangkauan";
    }
    else if (posisi == 1)
    {
        cout << "Ini bukan posisi tengah\n";
    }
    else
    {
        bantu = head;
        int penghitung = 1;
        while (penghitung != posisi - 1)
        {
            penghitung++;
            bantu = bantu->next;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

//Fungsi Ubah depan
void ubahDepan(mahasiswa data)
{
    string namaBefore = head->ITTP.nama;
    head->ITTP.nama = data.nama;
    head->ITTP.nim = data.nim;
    cout << "data " << namaBefore << " telah digan0 dengan data " <<
    data.nama << endl;
}

```

```

//Fungsi Ubah Belakang
void ubahBelakang(mahasiswa data)
{
    string namaBefore = tail->ITTP.nama;
    tail->ITTP.nama = data.nama;
    tail->ITTP.nim = data.nim;
    cout << "data " << namaBefore << " telah digan0 dengan data " <<
    data.nama << endl;
}

//Fungsi Ubah Tengah
void ubahTengah(mahasiswa data)
{
    int posisi;
    cout << "\nMasukkan posisi data yang akan diubah : ";
    cin >> posisi;

    if (posisi < 1 || posisi > hitungList())
    {
        cout << "\nPosisi diluar jangkauan\n";
    }
    else if (posisi == 1)
    {
        cout << "\nBukan posisi tengah\n";
    }
    else
    {
        bantu = head;
        int penghitung = 1;
        while (penghitung != posisi)
        {
            penghitung++;
            bantu = bantu->next;
        }
        bantu->ITTP.nama = data.nama;
        bantu->ITTP.nim = data.nim;
    }
}

//Fungsi Menampilkan List
void tampil()
{
    node *bantu = head;
    cout << "Nama "
    << " Nim\n";
    while (bantu != NULL)
    {
        cout << bantu->ITTP.nama << " " << bantu->ITTP.nim << endl;
        bantu = bantu->next;
    }
}

```

```

}
}
//Fungsi Hapus Depan
void hapusDepan()
{
string dataBefore = head->ITTP.nama;
hapus = head;
if (head != tail)
{
head = head->next;
delete hapus;
}
else
{
head = tail = NULL;
}
cout << "Data " << dataBefore << " berhasil dihapus\n";
}
//Fungsi Hapus Belakang
void hapusBelakang()
{
string dataBefore = head->ITTP.nama;
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;
}
cout << "Data " << dataBefore << " berhasil dihapus\n";
}
//Fungsi Hapus Tengah
void hapusTengah()
{
tampil();
cout << endl;
if (isEmpty() == false)
{
back:

```

```

int posisi;
cout << "Masukkan Posisi yang dihapus : ";
cin >> posisi;
if (posisi < 1 || posisi > hitungList())
{
cout << "\nPosisi di luar jangkauan!\n";

cout << "Masukkan posisi baru\n";
goto back;

}
else if (posisi == 1 || posisi == hitungList())
{
cout << "\nBukan Posisi tengah\n";

cout << "Masukkan posisi baru\n";
goto back;
}
else
{
bantu = head;
int penghitung = 1;
while (penghitung <= posisi)
{
if (penghitung == posisi - 1)
{
before = bantu;
}
if (penghitung == posisi)
{
hapus = bantu;
}
bantu = bantu->next;
penghitung++;
}
string dataBefore = hapus->ITTP.nama;
before->next = bantu;
delete hapus;
cout << "\nData " << dataBefore << " berhasil dihapus!\n";

}
}
else
{
cout << "\n!!! List Data Kosong !!!\n";
}
}

```



```

}
//Fungsi Hapus List
void hapusList()
{
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        delete hapus;
        bantu = bantu->next;
    }
    init();
    cout << "\nsemua data berhasil dihapus\n";
}
//Program Utama
int main()
{
    init();
    mahasiswa ITTP;
back:
    int operasi, posisi;
    cout << " PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << endl;
    cout << " =====\n\n" << 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 Teangah" << endl;
    cout << "10.Hapus list" << endl;
    cout << "11.Tampilkan" << endl;
    cout << "0. Exit" << endl;

    cout << "\nPilih Operasi :> ";
    cin >> operasi;

    switch (operasi)
    {
    case 1:
        cout << "tambah depan\n";
        insertDepan(Pendataan());
        cout << endl;
        goto back;
        break;

```

```
case 2:
cout << "tambah belakang\n";
insertBelakang(Pendataan());
cout << endl;
goto back;

break;
case 3:
cout << "tambah tengah\n";
cout << "nama : ";
cin >> ITTP.nama;
cout << "NIM : ";
cin >> ITTP.nim;
cout << "Posisi: ";
cin >> posisi;
insertTengah(ITTP, posisi);
cout << endl;
goto back;
break;
case 4:
cout << "ubah depan\n";
ubahDepan(Pendataan());
cout << endl;
goto back;
break;
case 5:
cout << "ubah belakang\n";
ubahBelakang(Pendataan());
cout << endl;
goto back;
break;
case 6:
cout << "ubah tengah\n";
ubahTengah(Pendataan());
cout << endl;
goto back;

break;
case 7:
cout << "hapus depan\n";
hapusDepan();
cout << endl;
goto back;
break;
case 8:
cout << "hapus belakang\n";
hapusBelakang();
cout << endl;
goto back;
```

```

break;
case 9:
cout << "hapus tengah\n";
hapusTengah();
cout << endl;
goto back;
break;
case 10:
cout << "hapus list\n";
hapusList();
cout << endl;
goto back;
break;
case 11:
tampil();
cout << endl;
goto back;
break;

case 0:
cout << "\nEXIT PROGRAM\n";
break;

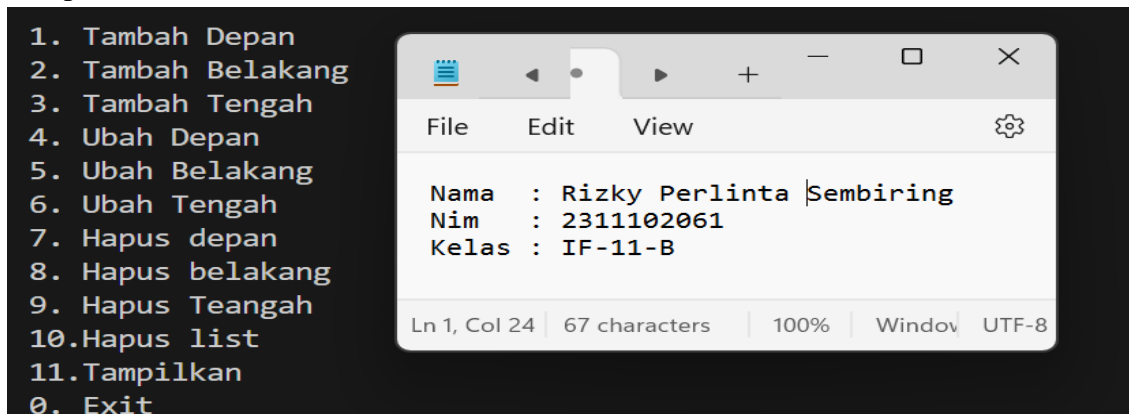
default:
cout << "\nSalah input operasi\n";
cout << endl;
goto back;
break;
}

return 0;
}

```

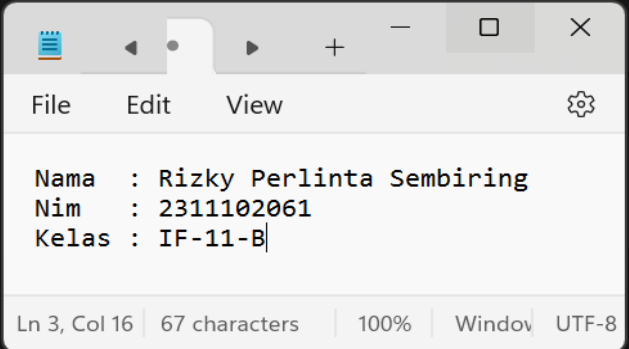
Screenshot Output:

1. Tampilan Menu



2. Tampilan berisi data

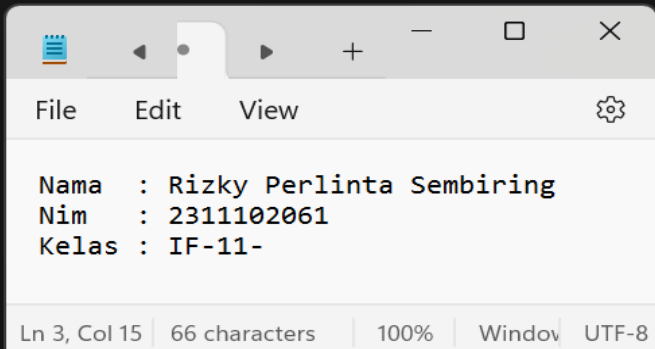
```
Nama Nim
Jawad 23300001
Rizky 2311102061
Farrel 23300003
Denis 23300005
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099
```



```
Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11-B|
```

3. Tambahkan data Wati 23300004 diantara Farrel dan Denis:

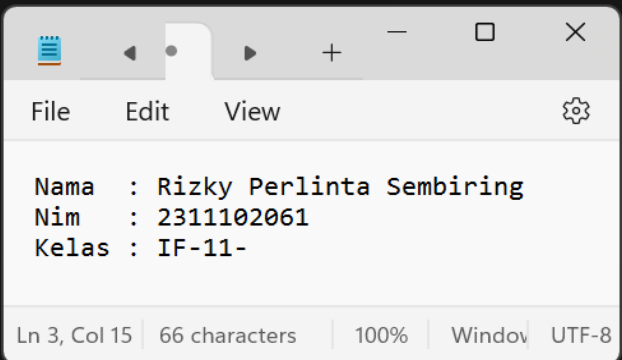
```
Nama Nim
Jawad 23300001
Rizky 2311102061
Farrel 23300003
Wati 23300004
Denis 23300005
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099
```



```
Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11-
```

4. Hapus data Denis

```
Nama Nim
Jawad 23300001
Rizky 2311102061
Farrel 23300003
Wati 23300004
Denis 23300005
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099
```



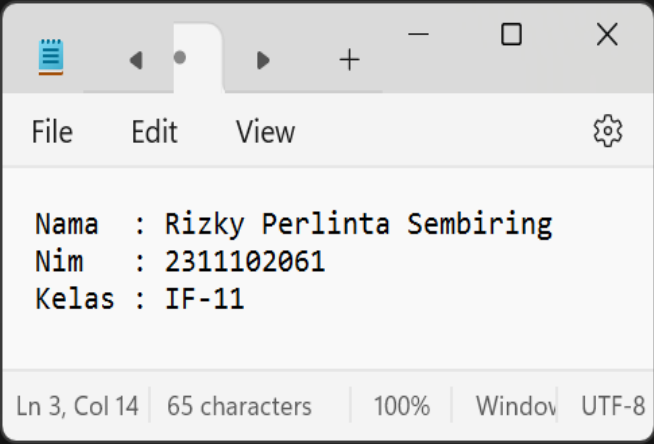
```
Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11-
```

Masukkan Posisi yang dihapus : 5

Data Denis berhasil dihapus!

5. Tambahkan data Owi 2330000 di awal

```
Owi 2330000
Jawad 23300001
Rizky 2311102061
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099
```



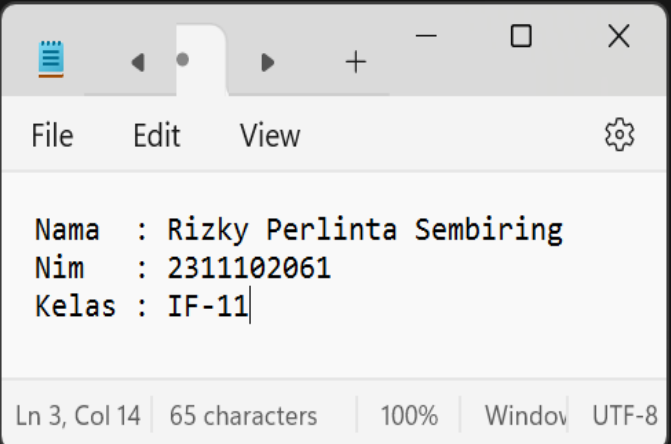
File Edit View

Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11

Ln 3, Col 14 | 65 characters | 100% | Window | UTF-8

6. Tambahkan data David 23300100 di akhir

```
Owi 2330000
Jawad 23300001
Rizky 2311102061
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Udin 23300048
Ucok 23300050
Budi 23300099
David 23300100
```



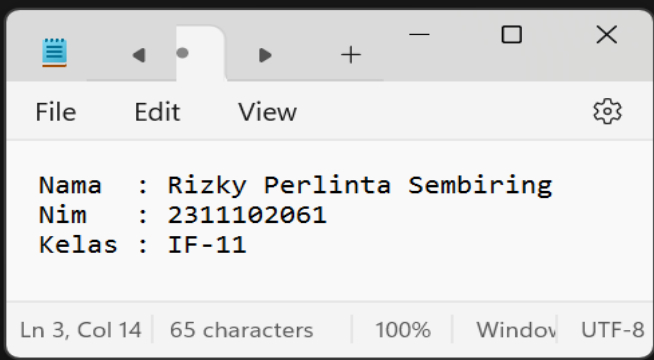
File Edit View

Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11

Ln 3, Col 14 | 65 characters | 100% | Window | UTF-8

7. Ubah data udin menjadi Idin 23300045

```
Owi 2330000
Jawad 23300001
Rizky 2311102061
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
David 23300100
```



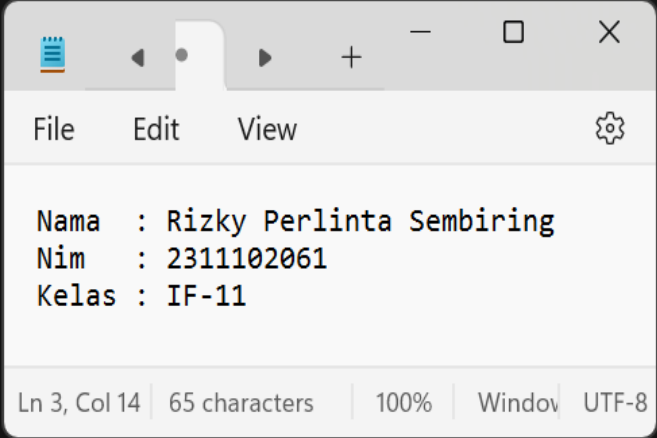
File Edit View

Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11

Ln 3, Col 14 | 65 characters | 100% | Window | UTF-8

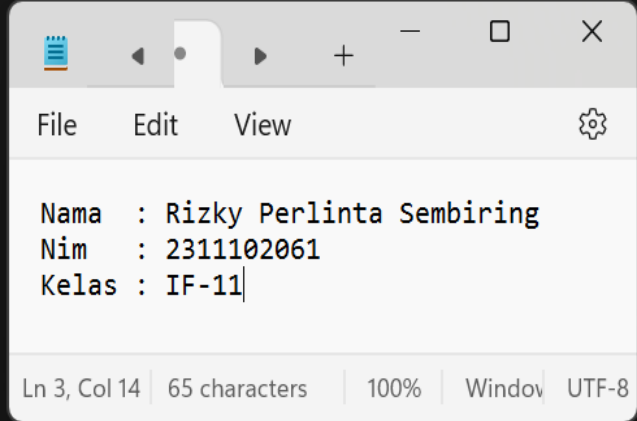
8. Ubah data terakhir menjadi Lucy 23300101

```
Owi 2330000
Jawad 23300001
Rizky 2311102061
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
Lucy 23300101
```



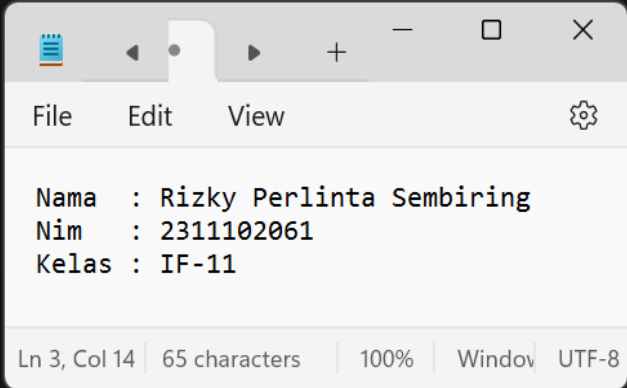
9. Hapus data awal

```
Jawad 23300001
Rizky 2311102061
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
Lucy 23300101
```



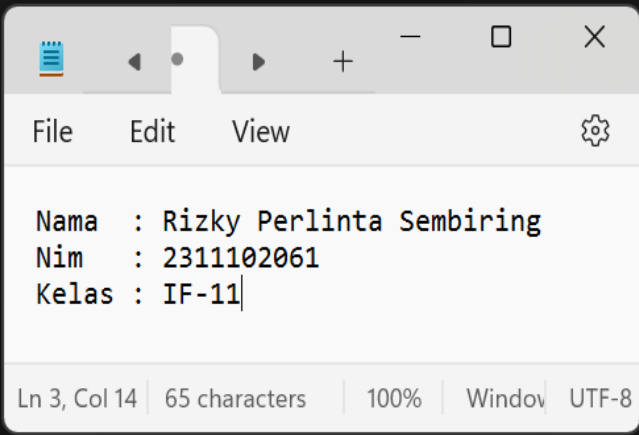
10. Ubah data awal menjadi Bagas 23300002

```
Bagas 2330002
Rizky 2311102061
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
Lucy 23300101
```



11. Hapus data akhir

```
Bagas 2330002
Rizky 2311102061
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
```



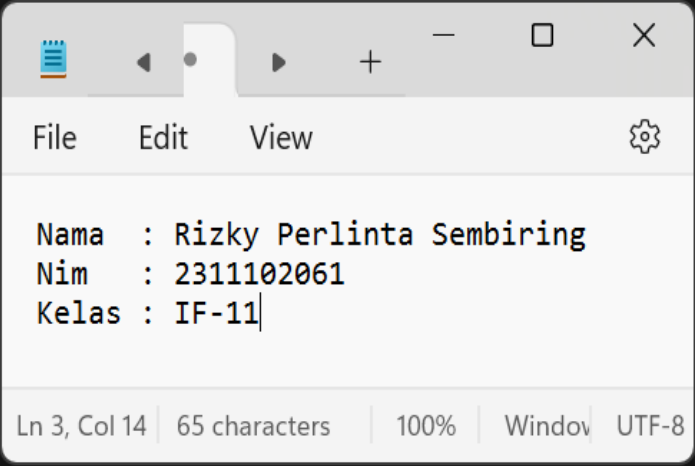
File Edit View

Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11

Ln 3, Col 14 | 65 characters | 100% | Window | UTF-8

12. Tampilkan seluruh data

```
Nama Nim
Bagas 2330002
Rizky 2311102061
Farrel 23300003
Wati 23300004
Anis 23300008
Bowo 23300015
Gahar 23300040
Idin 23300045
Ucok 23300050
Budi 23300099
```



File Edit View

Nama : Rizky Perlinta Sembiring
Nim : 2311102061
Kelas : IF-11

Ln 3, Col 14 | 65 characters | 100% | Window | UTF-8

Deskripsi Program : Program ini adalah program yang memanfaatkan struct mahasiswa dan struct node untuk menyimpan dan menampilkan data mahasiswa dalam bentuk linked list. Program ini memiliki beberapa fungsi yang dapat menambahkan, mengubah, dan menghapus data mahasiswa dalam linked list.

Pengguna juga dapat menjalankan program ini dan melakukan operasi-operasi yang ada pada menu, seperti tambah depan, tambah belakang, tambah tengah, ubah depan, ubah belakang, ubah tengah, hapus depan, hapus belakang, hapus tengah, dan hapus list. Program ini akan menampilkan data mahasiswa yang telah diinputkan melalui fungsi tampil, dan akan menghapus data mahasiswa melalui fungsi hapusDepan, hapusBelakang, atau hapusTengah.

D. Kesimpulan

Kesimpulannya, pemilihan antara linked list circular dan non-circular bergantung pada kebutuhan spesifik dari aplikasi atau algoritma yang sedang diimplementasikan. Jika akses linear dan tidak terbatas pada ujung linked list diperlukan, linked list circular mungkin merupakan pilihan yang lebih baik. Namun, jika urutan linear dari elemen dan manipulasi di ujung linked list lebih penting, linked list linear mungkin lebih sesuai.

E. Referensi

[1] Asisten Pratikum “Modul 3 SINGLE AND DOUBLE LIST”, Learning Management System, 2024.

[2] Taufikkipo (2012, juli). “Single Linked List Non Circular”.

Diakses pada 7 April 2024, dari

<https://taufikkipo.blogspot.com/2012/07/single-linked-list-non-circular.html>

[3] Trivusi (2020, 16 September).” Struktur Data Linked List: Pengertian, Karakteristik, dan Jenis-jenisnya”.

Diakses Pada 29 Maret 2024, dari

<https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html>