

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

MODUL 4



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

Disusun oleh:

ABDA FIRAS RAHMAN

2311102049

IF-11-B

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

BAB I

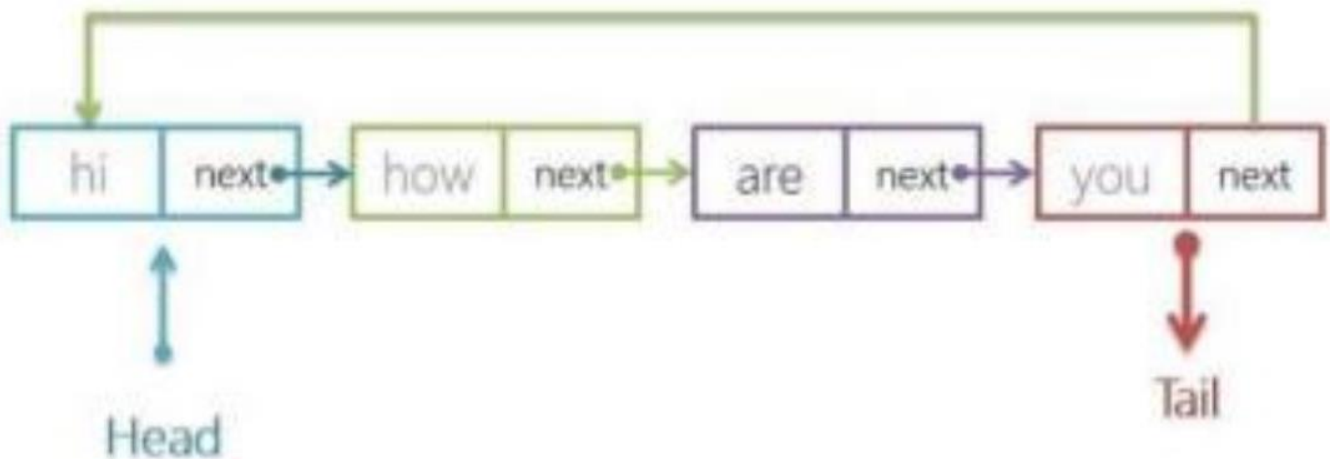
DASAR TEORI

LINKED LIST CIRCULAR DAN NON CIRCULAR

Linked List adalah salah satu struktur data penting dalam pemrograman yang digunakan untuk menyimpan dan mengelola data secara dinamis. Struktur data ini memungkinkan kita untuk dengan mudah membuat tempat baru untuk menyimpan data kapan saja dibutuhkan.

1. Linked List Non Circular

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



Gambar 2 *Single Linked List Circular*

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, *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  
    {
```

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) {  
            void tampil()  
            cout << helper->data << ends; helper = helper->next;  
        }  
    }  
}
```

```
}
```

2. 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). Circular linked list adalah linked list unidirectional. Kita hanya dapat melintasinya dalam satu arah. Tetapi jenis linked list ini memiliki simpul terakhir yang menunjuk ke simpul kepala. Jadi saat melintas, kita harus berhati-hati dan berhenti saat mengunjungi kembali simpul kepala.



Sumber: simplilearn.com

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;
}
}

```

C. 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;
}

```

OUTPUT :

```

2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11

```

DESKRIPSI PROGRAM:

Pada program C++ ini mengimplementasikan sebuah linked list dengan dua akhir, yang dimana dapat menyimpan data berupa bilangan bulat. Program ini memiliki banyak fungsi yang dapat digunakan untuk menambahkan, menghapus, dan mengubah data dalam linked list. Program ini secara berurutan melakukan operasi seperti: menyisipkan beberapa node di awal dan akhir linked list, menghapus node pertama dan terakhir, menyisipkan node di tengah, menghapus node di tengah, dan mengubah nilai node pertama, di tengah, dan terakhir. Kemudian, program menampilkan isi linked list setiap kali operasi dilakukan untuk memverifikasi hasilnya.

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 = 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;
}

```

OUTPUT:

```

pp -o guided2 } ;
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak
Bebek Ayam Cicak
Ayam Cicak
Ayam Sapi Cicak
PS C:\Users\LENOV M STRUKTUR DATA SMT 2\modul 4 praktikum strukdat>

```

Menu details:

| File | Edit | View |
|-------|------|-------------------|
| NAMA | : | ABDA FIRAS RAHMAN |
| KELAS | : | IF-11-B |
| NIM | : | 2311102049 |

Status bar: Ln 3, Col 17 | 57 characters | 100% | Window | UTF-8

DESKRIPSI PROGRAM:

Pada bagian program ini fungsi ini int() digunakan untuk memulihkan linked list. Ini dilakukan dengan mengatur nilai head dan tail menjadi NULL. Fungsi isEmpty() digunakan untuk mengecek apakah linked list kosong atau tidak. Ini dilakukan dengan mengembalikan nilai TRUE jika head adalah NULL, dan FALSE jika head tidak NULL. Fungsi buatNode(string data) digunakan untuk membuat Node baru dengan data yang diberikan dan next yang sama dengan NULL. Fungsi hitungList() digunakan untuk menghitung jumlah data dalam linked list. Ini dilakukan dengan menggunakan pointer bantu yang sama dengan head, dan menghitung jumlah Node yang ditemukan sampai nilai nextnya adalah NULL. Fungsi insertDepan(string data) digunakan untuk menambahkan data pada awal linked list. Ini dilakukan dengan membuat Node baru dengan data yang diberikan dan next yang sama dengan head. Kemudian, head dikembalikan ke Node baru.

Fungsi insertBelakang(string data) digunakan untuk menambahkan data pada akhir linked list.

Ini dilakukan dengan membuat Node baru dengan data yang diberikan dan next yang sama dengan NULL. Kemudian, Node baru dikembalikan ke tail. Fungsi insertTengah(string data, int posisi) digunakan untuk menambahkan data pada posisi tengah linked list. Ini dilakukan dengan menggunakan pointer bantu yang sama dengan head, dan melakukan looping sampai posisi yang diinginkan ditemukan. Kemudian, Node baru dibuat dengan data yang diberikan dan next yang sama dengan Node berikutnya. Kemudian, Node baru dikembalikan ke Node sebelumnya.

Fungsi hapusDepan() digunakan untuk menghapus data pada awal linked list. Ini dilakukan dengan mengambil nilai head, mengatur nilai head menjadi Node berikutnya, dan menghapus Node yang sebelumnya. Fungsi hapusBelakang() digunakan untuk menghapus data pada akhir linked list. Ini dilakukan dengan mengambil nilai head, mengatur nilai head menjadi Node sebelumnya, dan menghapus Node yang sebelumnya. Fungsi hapusTengah(int posisi) digunakan untuk menghapus data pada posisi tengah linked list. Ini dilakukan dengan menggunakan pointer bantu yang sama dengan head, dan melakukan looping sampai posisi yang diinginkan ditemukan. Kemudian, Node yang akan dihapus dikembalikan ke Node sebelumnya, dan Node tersebut dihapus. Fungsi clearList() digunakan untuk menghapus semua data dalam linked list.

UNGUIDED

UNGUIDED1

```
#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 diganθ 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;
}

```

OUTPUT:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus depan
8. Hapus belakang

Pilih Operasi :> 1
tambah depan

Masukkan Nama : firas
Masukkan NIM : 2311102049
Data firas berhasil diinput!

NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B

```

TAMPILAN OPERASI TAMBAH:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
10. Hapus list
11. Tampilkan
0. Exit

Pilih Operasi :> 1
tambah depan

Masukkan Nama : firas
Masukkan NIM : 2311102049
Data firas berhasil diinput!

NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B

```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
11. Tampilkan
0. Exit

Pilih Operasi :> 2
tambah belakang

Masukkan Nama : raka
Masukkan NIM : 2311102054

NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B

```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
11. Tampilkan
0. Exit

Pilih Operasi :> 3
tambah tengah
nama : zaki
NIM : 2311102051
Posisi: 2

NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B

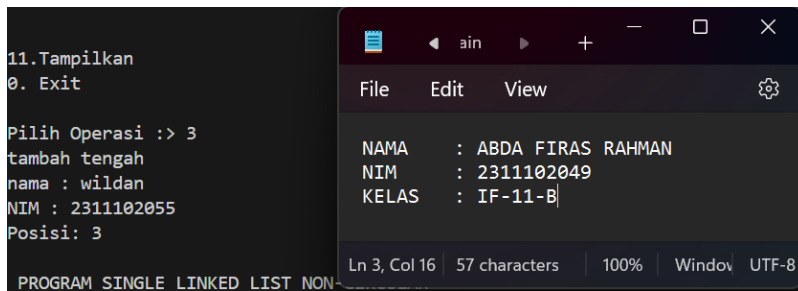
```



```
11.Tampilkan
0. Exit

Pilih Operasi :> 3
tambah tengah
nama : wildan
NIM : 2311102055
Posisi: 3

PROGRAM SINGLE LINKED LIST NON-
```

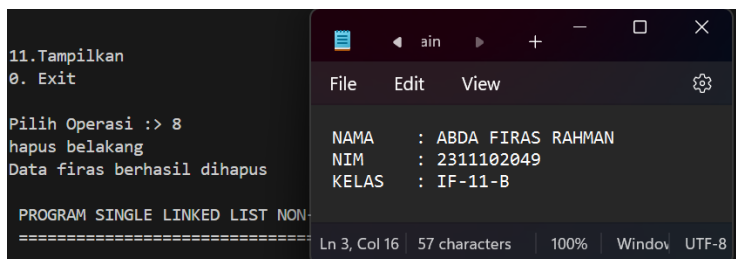


TAMPILAN OPERASI HAPUS:

```
11.Tampilkan
0. Exit

Pilih Operasi :> 8
hapus belakang
Data firas berhasil dihapus

PROGRAM SINGLE LINKED LIST NON-
```

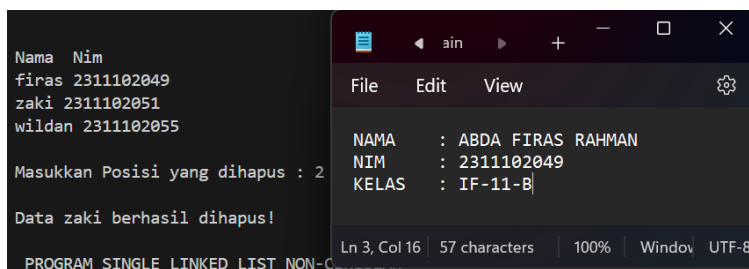


```
Nama Nim
firas 2311102049
zaki 2311102051
wildan 2311102055

Masukkan Posisi yang dihapus : 2

Data zaki berhasil dihapus!

PROGRAM SINGLE LINKED LIST NON-C
```



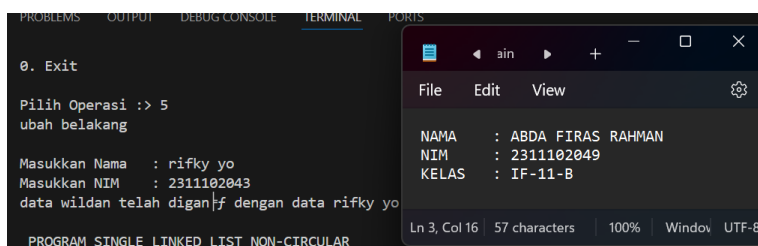
UBAH DATA DIBELAKANG:

```
0. Exit

Pilih Operasi :> 5
ubah belakang

Masukkan Nama : rifky yo
Masukkan NIM : 2311102043
data wildan telah diganlf dengan data rifky yo

PROGRAM SINGLE LINKED LIST NON-CIRCULAR
```

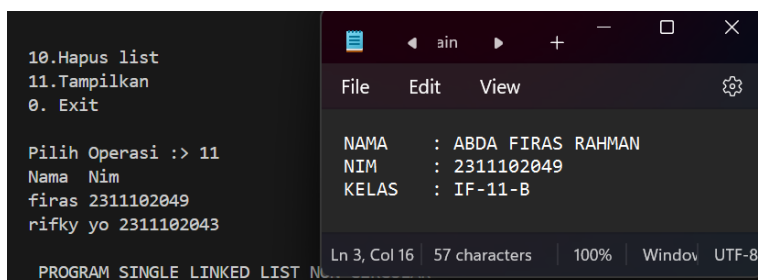


TAMPILKAN OPERASINYA:

```
10.Hapus list
11.Tampilkan
0. Exit

Pilih Operasi :> 11
Nama Nim
firas 2311102049
rifky yo 2311102043

PROGRAM SINGLE LINKED LIST N
```



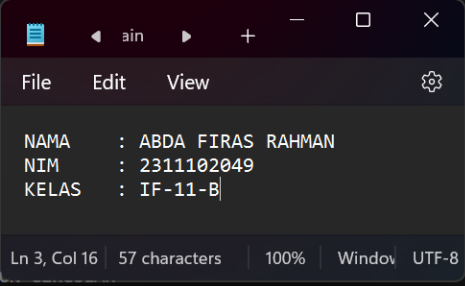
SETELAH MEMBUAT MENU TERSEBUT, MASUKKAN DATA SESUAI URUTAN BERIKUT, LALU TAMPILKAN DATA YANG TELAH DIMASUKKAN:

TAMPILAN PADA TAMBAH DEPAN:

```
0. Exit
Pilih Operasi :> 1
tambah depan

Masukkan Nama : fakih
Masukkan NIM : 2311102048
Data fakih berhasil diinput!

PROGRAM SINGLE LINKED LIST N...
```



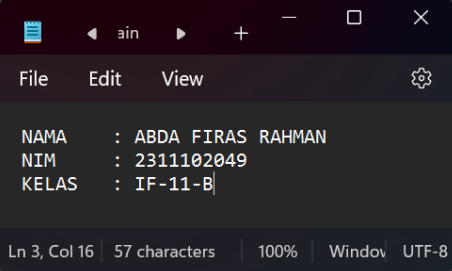
TAMPILAN PADA TAMBAH BELAKANG:

```
11.Tampilkan
0. Exit

Pilih Operasi :> 2
tambah belakang

Masukkan Nama : rizky
Masukkan NIM : 2311102061

PROGRAM SINGLE LINKED LIST N...
```

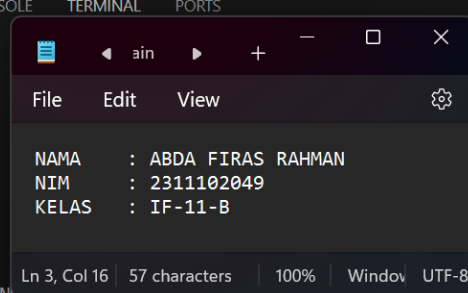


TAMPILAN PADA TAMBAH TENGAH:

```
11.Tampilkan
0. Exit

Pilih Operasi :> 3
tambah tengah
nama : firas
NIM : 2311102049
Posisi: 2

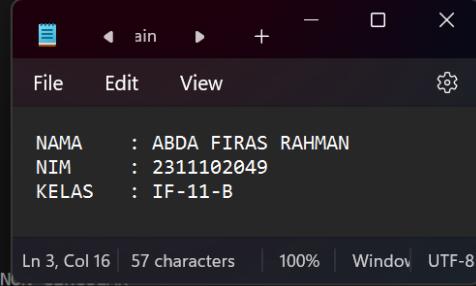
PROGRAM SINGLE LINKED LIST N...
```



```
11.Tampilkan
0. Exit

Pilih Operasi :> 3
tambah tengah
nama : galih
NIM : 2311102050
Posisi: 3

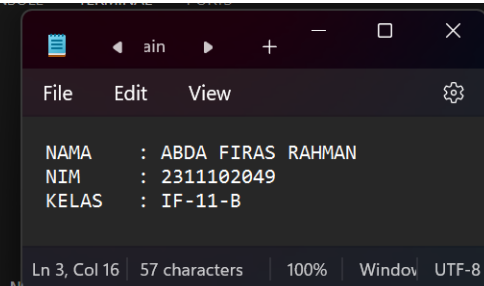
PROGRAM SINGLE LINKED LIST N...
```



```
11.Tampilkan
0. Exit

Pilih Operasi :> 3
tambah tengah
nama : denis
NIM : 23300005
Posisi: 4

PROGRAM SINGLE LINKED LIST N...
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
11.Tampilkan
0. Exit

Pilih Operasi :> 3
tambah tengah
nama : anis
NIM : 23300008
Posisi: 5

PROGRAM SINGLE LINKED LIST N...
```

```
ain + - □ ×
File Edit View
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
Ln 3, Col 16 | 57 characters | 100% | Window UTF-8
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
11.Tampilkan
0. Exit

Pilih Operasi :> 3
tambah tengah
nama : bowo
NIM : 23300015
Posisi: 6

PROGRAM SINGLE LINKED LIST N...
```

```
ain + - □ ×
File Edit View
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B|
Ln 3, Col 16 | 57 characters | 100% | Window UTF-8
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
11.Tampilkan
0. Exit

Pilih Operasi :> 3
tambah tengah
nama : gahar
NIM : 23300040
Posisi: 7

PROGRAM SINGLE LINKED LIST N...
```

```
ain + - □ ×
File Edit View
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
Ln 3, Col 16 | 57 characters | 100% | Window UTF-8
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
11.Tampilkan
0. Exit

Pilih Operasi :> 3
tambah tengah
nama : udin
NIM : 23300045
Posisi: 8

PROGRAM SINGLE LINKED LIST N...
```

```
ain + - □ ×
File Edit View
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B|
Ln 3, Col 16 | 57 characters | 100% | Window UTF-8
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
11.Tampilkan
0. Exit

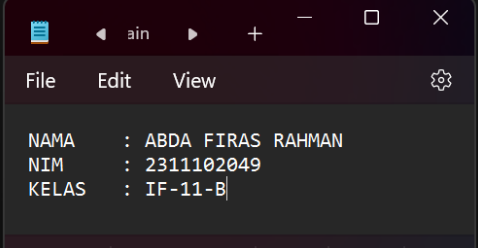
Pilih Operasi :> 3
tambah tengah
nama : ucok
NIM : 23300050
Posisi: 9

PROGRAM SINGLE LINKED LIST N...
```

```
ain + - □ ×
File Edit View
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
Ln 3, Col 16 | 57 characters | 100% | Window UTF-8
```

TAMPILAN NYA:

```
Pilih Operasi :> 11
Nama Nim
fakih 2311102048
firas 2311102049
galih 2311102050
denis 23300005
anis 23300008
bowo 23300015
gahar 23300040
udin 23300045
ucok 23300050
```



LAKUKAN PERINTAH BERIKUT:

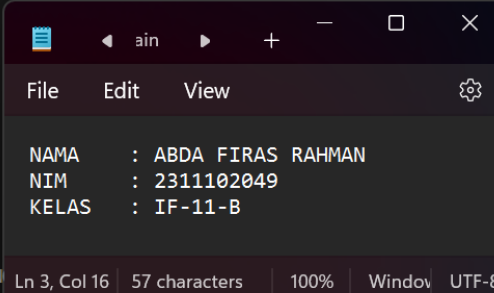
Tambahkan data berikut diantara Farrel dan Denis:

Wati 23300004

```
11.Tampilkan
0. Exit

Pilih Operasi :> 3
tambah tengah
nama : wati
NIM : 23300004
Posisi: 4

PROGRAM SINGLE LINKED LIST N
```

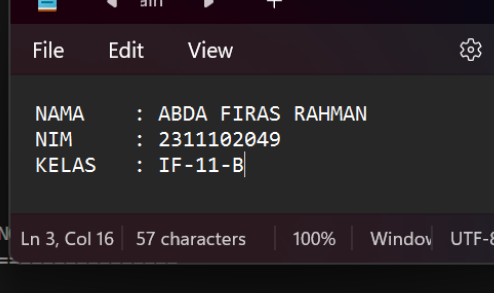


HAPUS DATA DENIS

```
ucok 23300050
firas 2311102049
rifky yo 2311102043
rizky 2311102061

Masukkan Posisi yang dihapus
Data denis berhasil dihapus!

PROGRAM SINGLE LINKED LIST N
```



TAMBAHKAN DATA BERIKUT DI AWAL:

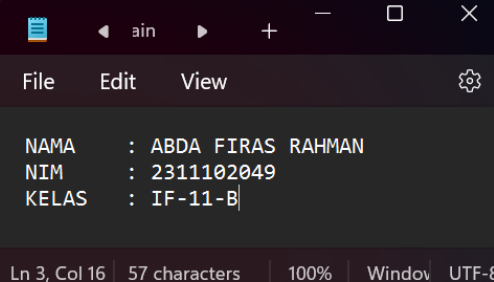
Owi 23300000

```
0. Exit

Pilih Operasi :> 1
tambah depan

Masukkan Nama : owi
Masukkan NIM : 23300000
Data owi berhasil diinput!

PROGRAM SINGLE LINKED LIST N
```



TAMBAHKAN DATA BERIKUT DI AKHIR:

David 23300100

```
11.Tampilkan
9. Exit

Pilih Operasi :> 2
tambah belakang

Masukkan Nama : david
Masukkan NIM : 23300100

PROGRAM SINGLE LINKED LIST N
=====
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
Ln 3, Col 16 | 57 characters | 100% | Window | UTF-8
```

UBAH DATA UDIN MENJADI DATA BERIKUT:
Idin 23300045

```
Pilih Operasi :> 6
ubah tengah

Masukkan Nama : idin
Masukkan NIM : 2330045

Masukkan posisi data yang akan
=====
PROGRAM SINGLE LINKED LIST N
=====
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
Ln 3, Col 16 | 57 characters | 100% | Window | UTF-8
```

UBAH DATA TERKAHIR MENJADI BERIKUT:
LUCY 23300101

```
Pilih Operasi :> 5
ubah belakang

Masukkan Nama : lucy
Masukkan NIM : 233000101

data david telah digan
=====
PROGRAM SINGLE LINKED LIST N
=====
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
```

HAPUS DATA AWAL

```
9. Exit

Pilih Operasi :> 7
hapus depan
Data owi berhasil dihapus

=====
PROGRAM SINGLE LINKED LIST N
=====
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
```

UBAH DATA AWAL MENJADI BERIKUT:
BAGAS 23300002

```
Masukkan Nama : bagas
Masukkan NIM : 23300002

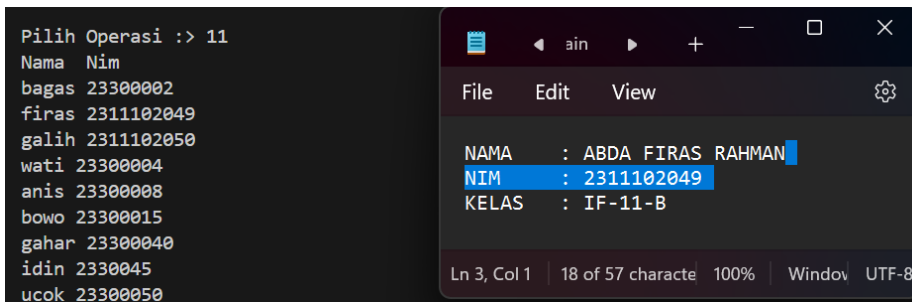
data fakhri telah digan
=====
PROGRAM SINGLE LINKED LIST N
=====
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
```

HAPUS DATA AKHIR:

```
Pilih Operasi :> 8
hapus belakang
Data bagas berhasil dihapus

=====
PROGRAM SINGLE LINKED LIST N
=====
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
```

TAMPILKAN SELURUH DATA



```
Pilih Operasi :> 11
Nama Nim
bagas 23300002
firas 2311102049
galih 2311102050
wati 23300004
anis 23300008
bowo 23300015
gahar 23300040
idin 2330045
ucok 23300050
```

```
File Edit View
NAMA : ABDA FIRAS RAHMAN
NIM : 2311102049
KELAS : IF-11-B
Ln 3, Col 1 | 18 of 57 character | 100% | Window UTF-8
```

DESKRIPSI PROGRAM:

Program ini menggunakan struktur data mahasiswa yang memiliki dua field, yaitu nama dan NIM. Dan struct node yang memiliki field ITTP yang berisi data mahasiswa dan next yang berisi pointer ke Node berikutnya. Program ini juga menggunakan beberapa variabel global, seperti head, tail, bantu, hapus, before, dan baru, yang digunakan untuk menyimpan referensi ke Node. Fungsi init() digunakan untuk memulihkan linked list dengan mengatur nilai head dan tail menjadi NULL. Fungsi isEmpty() digunakan untuk mengecek apakah linked list kosong atau tidak. Ini dilakukan dengan mengembalikan nilai TRUE jika head adalah NULL, dan FALSE jika head tidak NULL. Fungsi Pendataan() digunakan untuk meminta input data dari pengguna, seperti nama dan NIM. Fungsi insertDepan(mahasiswa ITTP) digunakan untuk menambahkan data pada awal linked list. Ini dilakukan dengan membuat Node baru dengan data yang diberikan dan next yang sama dengan head. Kemudian, head dikembalikan ke Node baru.

Fungsi insertBelakang(mahasiswa ITTP) digunakan untuk menambahkan data pada akhir linked list. Ini dilakukan dengan membuat Node baru dengan data yang diberikan dan next yang sama dengan NULL. Kemudian, Node baru dikembalikan ke tail.

Fungsi insertTengah(mahasiswa idenOtas, int posisi) digunakan untuk menambahkan data pada posisi tengah linked list. Ini dilakukan dengan menggunakan pointer bantu yang sama dengan head, dan melakukan looping sampai posisi yang diinginkan ditemukan. Kemudian, Node baru dibuat dengan data yang diberikan dan next yang sama dengan Node berikutnya. Kemudian, Node baru dikembalikan ke Node sebelumnya.

Fungsi ubahDepan(mahasiswa data) digunakan untuk mengubah data pada awal linked list. Ini dilakukan dengan mengubah data field nama dan NIM pada Node yang sama dengan head. Fungsi ubahBelakang(mahasiswa data) digunakan untuk mengubah data pada akhir linked list. Ini dilakukan dengan mengubah data field nama dan NIM pada Node yang sama dengan tail. Fungsi ubahTengah(mahasiswa data) digunakan untuk mengubah data pada posisi tengah linked list. Ini dilakukan dengan menggunakan pointer bantu yang sama dengan head, dan melakukan looping sampai posisi yang diinginkan ditemukan. Kemudian, data field nama dan NIM pada Node tersebut diubah.

BAB IV

KESIMPULAN

Linked List adalah salah satu struktur data penting dalam pemrograman yang digunakan untuk menyimpan dan mengelola data secara dinamis. Struktur data ini memungkinkan kita untuk dengan mudah membuat tempat baru untuk menyimpan data kapan saja dibutuhkan.

Setiap Linked List memiliki dua elemen khusus, yaitu “head” dan “tail”:

- Head: Merupakan simpul pertama dalam Linked List dan berfungsi sebagai titik awal akses ke seluruh data dalam Linked List.
- Tail: Merupakan simpul terakhir dalam Linked List dan menjadi penanda akhir dari urutan simpul.

✓ 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 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). Circular linked list adalah linked list unidirectional. Kita hanya dapat melintasinya dalam satu arah. Tetapi jenis linked list ini memiliki simpul terakhir yang menunjuk ke simpul kepala.

DAFTAR PUSTAKA

Struktur Data Linked List: Pengertian, Karakteristik, dan Jenis-jenisnya :

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