LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA

MODUL IV
CIRCULAR DAN NON CIRCULAR

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

Disusun oleh:

HAIKAL SATRIATAMA

2311102066

IF-11-B

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

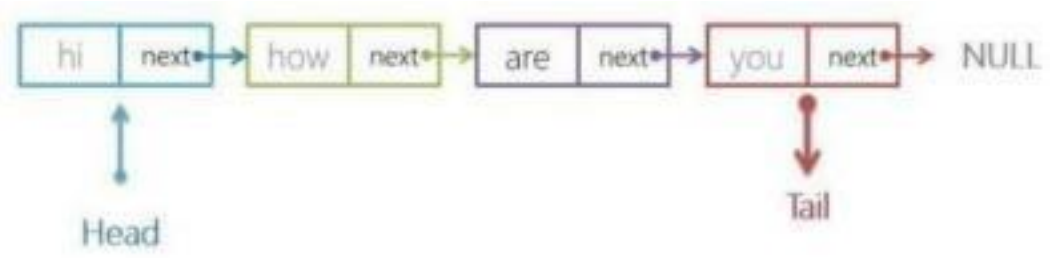
2024

BAB I

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. *Linkedlist non circular* dapat digambarkan sebagai berikut.



Gambar 1 *Single Linked List Non 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. **Pengecek 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;
        bar-> 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;
        }
        }
}
```

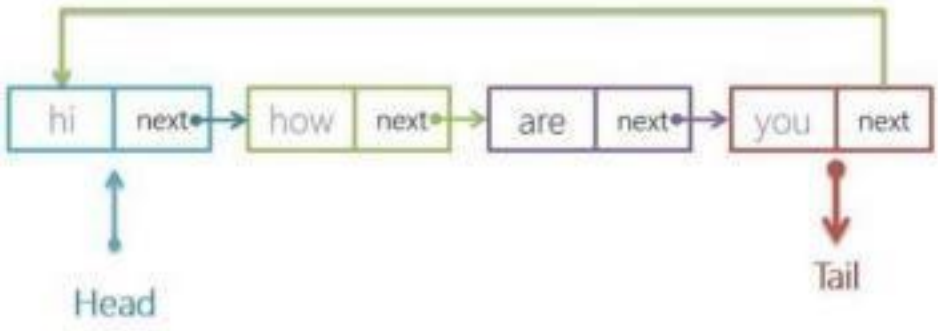
6. Tampilkan Data Linked List

```
        void tampil()
        {
        if (isEmpty() == true)
        {
        cout << "List kosong!" << endl;
        }

        else
        {
        node *helper;
        helper = head;
        while (helper != NULL)
        {
        cout << helper->data << endl;
        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. *Linked list circular* dapat digambarkan sebagai berikut.



Gambar 2 *Single 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 << endl;
        tail = tail->next;
        } while (tail != head);
        cout << endl;
        }
}
```

BAB II

GUIDED

LATIHAN – GUIDED

1. Guided 1
   Latihan linked list non circular
   Source Code

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

// Deklarasi Struct Node
struct Node {
    int data;
    Node* next;
};

Node* head;
Node* tail;

// Inisialisasi Node
void init() {
    head = NULL;
    tail = NULL;
}

// Pengecekan apakah list kosong
bool isEmpty() {
    return head == NULL;
}

// Tambah Node di depan
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;
    }
}

// Tambah Node di belakang
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;
    }
}

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

```
// Tambah Node di tengah
void insertTengah(int data, int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {
        Node* baru = 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;
    }
}

// Hapus Node di depan
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;
    }
}
```

```
// Hapus Node di belakang
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;
    }
}

// Hapus Node di tengah
void hapusTengah(int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {
        Node* bantu = head;
        Node* hapus;
        Node* sebelum;
        int nomor = 1;
        while (nomor <= posisi) {
            if (nomor == posisi - 1) {
```

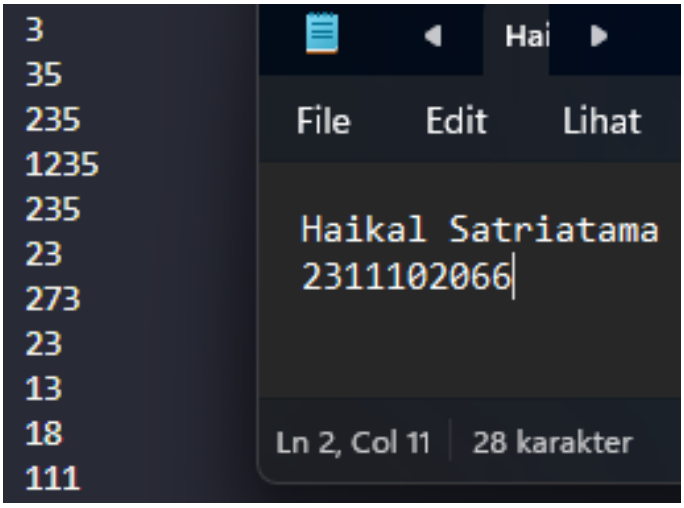
```
                sebelum = bantu;
            }
            if (nomor == posisi) {
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
        }
        sebelum->next = bantu;
        delete hapus;
    }
}

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

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

// Ubah data Node di tengah
void ubahTengah(int data, int posisi) {
    if (!isEmpty()) {
```

```
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi diluar 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;
    }
}

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

// Tampilkan isi list
void tampil() {
    Node* bantu = head;
    if (!isEmpty()) {
```

```
        while (bantu != NULL) {
            cout << bantu->data << ends;
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

int main() {
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();
```

```
    return 0;
}
```

Screenshoot program



Deskripsi program

Pertama, fungsi init digunakan dalam fungsi main() untuk menginisialisasi program. Ini menginisialisasi pointer kepala dan ekor list menjadi NULL. Selain itu, berbagai operasi dapat dilakukan pada daftar yang terhubung, seperti memasukkan node di depan, di belakang, dan di tengah dengan menggunakan fungsi insertDepan(), insertBelakang(), dan insertTengah; menghapus node di depan, di belakang, dan di tengah dengan menggunakan fungsi hapusDepan(), hapusBelakang(), dan hapusTengah; dan mengubah nilai node di depan, di belakang, dan di tengah dengan menggunakan fungsi ubahDepan(), ubahBelakang(), dan ubahTengah Pemanggilan fungsi tampil() untuk menampilkan isi linked list dilakukan setelah setiap operasi pada daftar terkait. Terakhir, program selesai dengan mengembalikan nilai 0. Struktur daftar tunggal terhubung digunakan, dengan setiap node terdiri dari sebuah integer (int data) dan pointer next yang menunjuk ke node berikutnya.

2. Guided 2
   Latihan 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 {
```

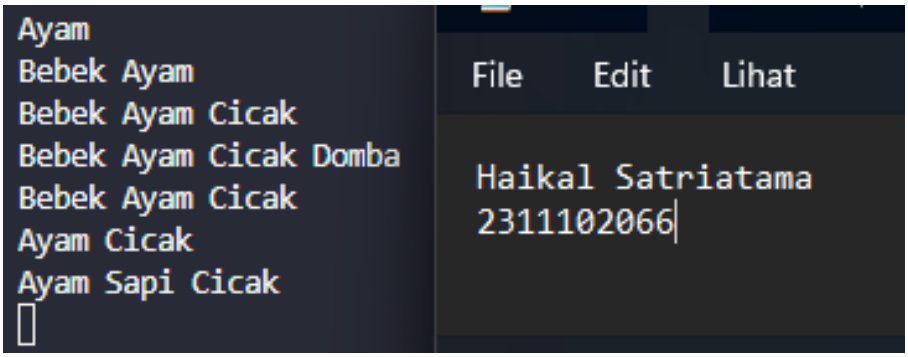
```
        baru->data = data;
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

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;
}
```

Screenshoot program



Deskripsi program

Circular linked list ini menawarkan sejumlah operasi dasar, termasuk menambahkan node di bagian depan, belakang, dan tengah, menghapus node di bagian depan, belakang, dan tengah, membersihkan seluruh isi list, serta menampilkan isi list. Dalam fungsi main(), program dimulai dengan memanggil fungsi init() untuk menginisialisasi pointer kepala dan ekor list menjadi NULL. Selanjutnya, berbagai operasi dilakukan

pada circular linked list seperti memasukkan node di depan dan belakang menggunakan fungsi insertDepan() dan insertBelakang(), menghapus node di depan dan belakang menggunakan fungsi hapusDepan() dan hapusBelakang(), memasukkan node di tengah menggunakan fungsi insertTengah(), dan menghapus node di tengah menggunakan fungsi hapusTengah(). Setiap operasi pada linked list diikuti dengan memanggil fungsi tampil() untuk menampilkan isi dari linked list tersebut. Akhirnya, program selesai dengan mengembalikan nilai 0.

BAB III

UNGUIDED

TUGAS – UNGUIDED

1. Unguided 1
   Buatlah program menu Linked List Non Circular untuk menyimpan Nama dan NIM mahasiswa, dengan menggunakan input dari user.

   Source Code

```
#include <iostream>

#include <string> // Include string header explicitly if you use
string directly

using namespace std;


struct mahasiswa

{

    string nama;

    string nim;

};


struct node

{

    mahasiswa ITTP;

    node *next;

};


node *head = nullptr; // Initialize pointers to nullptr

node *tail = nullptr;

node *bantu = nullptr;

node *hapus = nullptr;

node *before = nullptr;

node *baru = nullptr;


// Function prototypes

void init();

bool isEmpty();
```

```
mahasiswa Pendataan();
void insertDepan(mahasiswa ITTP);
void insertBelakang(mahasiswa ITTP);
int hitungList();
void insertTengah(mahasiswa identitas, int posisi);
void ubahDepan(mahasiswa data);
void ubahBelakang(mahasiswa data);
void ubahTengah(mahasiswa data);
void tampil();
void hapusDepan();
void hapusBelakang();
void hapusTengah();
void hapusList();

int main()
{
    init();
    mahasiswa ITTP;
back:
    int operasi, posisi;
    cout << " PROGRAM DATA MAHASISWA" << endl;
    cout << " ======================" << endl;
    cout << "1. Tambah Depan" << endl;
    cout << "2. Tambah Belakang" << endl;
    cout << "3. Tambah Tengah" << endl;
    cout << "4. Ubah Depan" << endl;
    cout << "5. Ubah Belakang" << endl;
    cout << "6. Ubah Tengah" << endl;
    cout << "7. Hapus depan" << endl;
    cout << "8. Hapus belakang" << endl;
    cout << "9. Hapus Tengah" << endl;
    cout << "10.Hapus list" << endl;
    cout << "11.Tampilkan" << endl;
    cout << "0. 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;
}

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

bool isEmpty()
{
    return head == nullptr;
```

```
}

mahasiswa Pendataan()
{
    mahasiswa ITTP;
    cout << "\nMasukkan Nama\t: ";
    cin.ignore();
    getline(cin, ITTP.nama);
    cout << "Masukkan NIM\t: ";
    cin >> ITTP.nim;
    return ITTP;
}

void insertDepan(mahasiswa ITTP)
{
    node *baru = new node;
    baru->ITTP = ITTP;
    baru->next = nullptr;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
    cout << "Data " << ITTP.nama << " berhasil diinput!\n";
}

void insertBelakang(mahasiswa ITTP)
{
    node *baru = new node;
    baru->ITTP = ITTP;
```

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

int hitungList()
{
    int penghitung = 0;
    node *bantu = head;
    while (bantu != nullptr)
    {
        penghitung++;
        bantu = bantu->next;
    }
    return penghitung;
}

void insertTengah(mahasiswa identitas, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "posisi diluar jangakauan";
    }
    else if (posisi == 1)
    {
        cout << "INi bukan posisi tengah\n";
    }
```

```
    else
    {
        node *baru = new node;
        baru->ITTP = identitas;
        node *bantu = head;
        int penghitung = 1;
        while (penghitung != posisi - 1)
        {
            penghitung++;
            bantu = bantu->next;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void ubahDepan(mahasiswa data)
{
    if (!isEmpty())
    {
        string namaBefore = head->ITTP.nama;
        head->ITTP = data;
        cout << "data " << namaBefore << " telah diganti dengan data
" << data.nama << endl;
    }
    else
    {
        cout << "List kosong" << endl;
    }
}

void ubahBelakang(mahasiswa data)
{
    if (!isEmpty())
```

```
    {
        string namaBefore = tail->ITTP.nama;
        tail->ITTP = data;
        cout << "data " << namaBefore << " telah diganti dengan data
" << data.nama << endl;
    }
    else
    {
        cout << "List kosong" << endl;
    }
}

void ubahTengah(mahasiswa data)
{
    if (!isEmpty())
    {
        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
        {
            node *bantu = head;
            int penghitung = 1;
            while (penghitung != posisi)
            {
```

```
                penghitung++;
                bantu = bantu->next;
            }
            bantu->ITTP = data;
        }
    }
    else
    {
        cout << "List kosong" << endl;
    }
}

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

void hapusDepan()
{
    if (!isEmpty())
    {
        string dataBefore = head->ITTP.nama;
        node *hapus = head;
        if (head != tail)
        {
            head = head->next;
            delete hapus;
```

```
        }
        else
        {
            head = tail = nullptr;
        }
        cout << "Data " << dataBefore << " berhasil dihapus\n";
    }
    else
    {
        cout << "List kosong" << endl;
    }
}

void hapusBelakang()
{
    if (!isEmpty())
    {
        string dataBefore = tail->ITTP.nama;
        if (head != tail)
        {
            node *hapus = tail;
            node *bantu = head;
            while (bantu->next != tail)
            {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = nullptr;
            delete hapus;
        }
        else
        {
            head = tail = nullptr;
        }
```

```
        cout << "Data " << dataBefore << " berhasil dihapus\n";
    }
    else
    {
        cout << "List kosong" << endl;
    }
}

void hapusTengah()
{
    if (!isEmpty())
    {
        tampil();
        cout << endl;
        int posisi;
        cout << "Masukkan Posisi yang dihapus : ";
        cin >> posisi;
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "\nPosisi di luar jangkauan!\n";
        }
        else if (posisi == 1 || posisi == hitungList())
        {
            cout << "\nBukan Posisi tengah\n";
        }
        else
        {
            node *bantu = head;
            int penghitung = 1;
            while (penghitung <= posisi)
            {
                if (penghitung == posisi - 1)
                {
                    before = bantu;
```

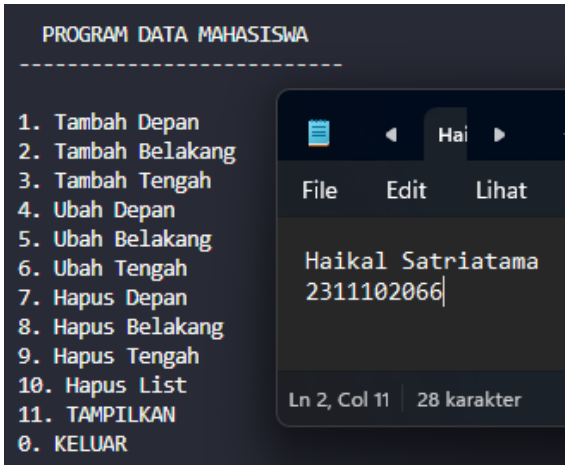
```
                }
                if (penghitung == posisi)
                {
                    hapus = bantu;
                }
                bantu = bantu->next;
                penghitung++;
            }
            string dataBefore = hapus->ITTP.nama;
            before->next = bantu;
            delete hapus;
            cout << "\nData " << dataBefore << " berhasil
dihapus!\n";
        }
    }
    else
    {
        cout << "\n!!! List Data Kosong !!!\n";
    }
}

void hapusList()
{
    node *bantu = head;
    while (bantu != nullptr)
    {
        hapus = bantu;
        delete hapus;
        bantu = bantu->next;
    }
    init();
    cout << "\nsemua data berhasil dihapus\n";
}
```

Deskripsi program & Screenshot Program

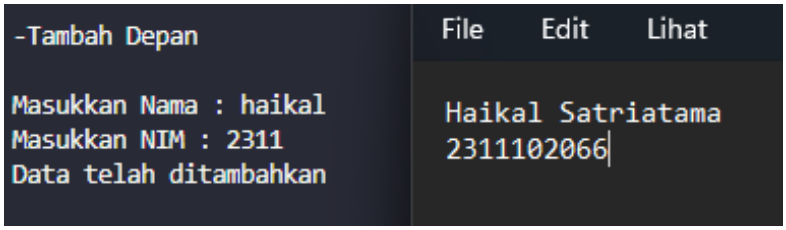
Di dalam kelas LinkedList, terdapat beragam metode yang berfungsi untuk mengelola informasi mengenai mahasiswa, termasuk nama dan NIM mereka. Metode-metode tersebut mencakup penambahan data di bagian depan (melalui tambahDepan()), bagian belakang (melalui tambahBelakang()), atau di tengah berdasarkan posisi (melalui tambahTengah()), serta pengubahan data di bagian depan (melalui ubahDepan()), bagian belakang (melalui ubahBelakang()), atau di tengah berdasarkan posisi (melalui ubahTengah()). Selain itu, terdapat juga metode untuk menghapus data di bagian depan (hapusDepan()), bagian belakang (hapusBelakang()), atau di tengah berdasarkan posisi (hapusTengah()), menampilkan seluruh data mahasiswa (tampilkanData()), dan menghapus keseluruhan data dalam linked list (hapusList()).

Dalam fungsi main(), terdapat loop tak terbatas yang memungkinkan pengguna untuk memilih operasi apa yang ingin dilakukan pada linked list melalui menu yang disediakan. Pengguna dapat memilih untuk menambahkan data, mengubah data, menghapus data, menampilkan seluruh data, atau keluar dari program. Setiap pilihan akan memanggil metode yang sesuai pada objek linkedList yang telah dibuat.

1. Buatlah menu untuk menambahkan, mengubah, menghapus, dan melihat Nama dan NIM mahasiswa
   - Tampilan menu



   - Operasi Tambah



```
-Tambah Belakang

Masukkan Nama : Satria
Masukkan NIM : 2210
Data telah ditambahkan
```

```
-Tambah Tengah

Masukkan Nama : Tama
Masukkan NIM : 2100
Masukkan Posisi : 2
Data telah ditambahkan
```

- Operasi Hapus

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

```
Pilih Operasi :> 9
hapus tengah
Nama  Nim
Haikal 2311
Tama 2100

Masukkan Posisi yang dihapus : 1

Bukan Posisi tengah
```

- Operasi ubah

```
Pilih Operasi :> 4
ubah depan

Masukkan Nama   : h
Masukkan NIM    : 1212
data Haikal telah diganti dengan data h
```

```
Pilih Operasi :> 6
ubah tengah

Masukkan Nama   : J
Masukkan NIM    : 777

Masukkan posisi data yang akan diubah : 2
```

- Operasi tampil data

```
Pilih Operasi :> 11
Nama  Nim
h 1212
J 777
```

2. Masukkan data sesuai urutan

```
Pilih Operasi :> 11
Nama  Nim
Budi 23300099
Ucok 23300050
Udin 23300048
Gahar 23300040
Bowo 23300015
Anis 23300008
Denis 23300005
Farrel 23300003
Haikal Satriatama 2311102066
Jawadd 23300001
```

3. Lakukan Perintah

```
Pilih Operasi :> 3
tambah tengah
nama : Wati
NIM : 2330004
Posisi: 8
```

```
Masukkan Posisi yang dihapus : 7

Data Denis berhasil dihapus!
```

```
Pilih Operasi :> 1
tambah depan

Masukkan Nama    : Owi
Masukkan NIM     : 2330000
Data Owi berhasil diinput!
```

```
Pilih Operasi :> 2
tambah belakang

Masukkan Nama    : David
Masukkan NIM     : 23300100
```

```
Pilih Operasi :> 6
ubah tengah

Masukkan Nama    : Idin
Masukkan NIM     : 23300045

Masukkan posisi data yang akan diubah : 4
```

```
Pilih Operasi :> 5
ubah belakang

Masukkan Nama    : Lucy
Masukkan NIM     : 23300101
data David telah diganti dengan data Lucy
```

```
Pilih Operasi :> 7
hapus depan
Data Owi berhasil dihapus
```

```
Pilih Operasi :> 4
ubah depan

Masukkan Nama    : Bagas
Masukkan NIM     : 2330002
data Budi telah diganti dengan data Bagas
```

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

```
Pilih Operasi :> 11
Nama  Nim
Bagas 2330002
Ucok 23300050
Idin 23300045
Gahar 23300040
Bowo 23300015
Anis 23300008
Wati 2330004
Farrel 23300003
Haikal Satriatama 2311102066
Jawadd 23300001
```

BAB IV

KESIMPULAN

Linked list adalah struktur data yang terdiri dari serangkaian simpul yang saling terhubung, di mana setiap simpul memiliki dua komponen utama: data yang disimpan dan referensi ke simpul berikutnya dalam rangkaian. Ada dua jenis linked list yang umum digunakan: circular linked list dan non-circular linked list. Circular linked list memiliki sifat khusus di mana simpul terakhir dalam daftar memiliki referensi yang kembali ke simpul pertama, membentuk sebuah lingkaran atau cincin. Sementara itu, non-circular linked list, seperti namanya, tidak memiliki referensi kembali ke simpul pertama, sehingga simpul terakhir menunjuk ke nilai null.

DAFTAR PUSTAKA

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357.

Trivusi web, Struktur data Linked list : Pengertian, karakteristik, dan jenis-jenisnya (2022) https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html