# LAPORAN PRAKTIKUM
# STRUKTUR DATA DAN ALGORITMA

## MODUL IV
## CIRCULAR DAN NON CIRCULAR


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

**Disusun oleh:**

**MUHAMMAD AULIA MUZZAKI NUGRAHA**

**(2311102051)**

**IF-11-B**



**PROGRAM STUDI S1 INFORMATIKA**

**FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

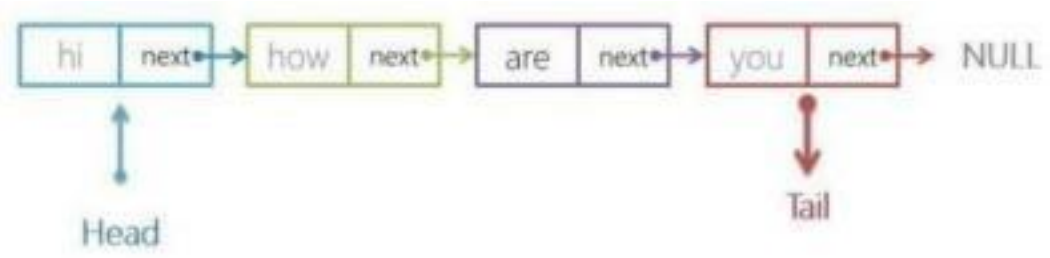
**2024**

# 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;
        }
        }
}
```

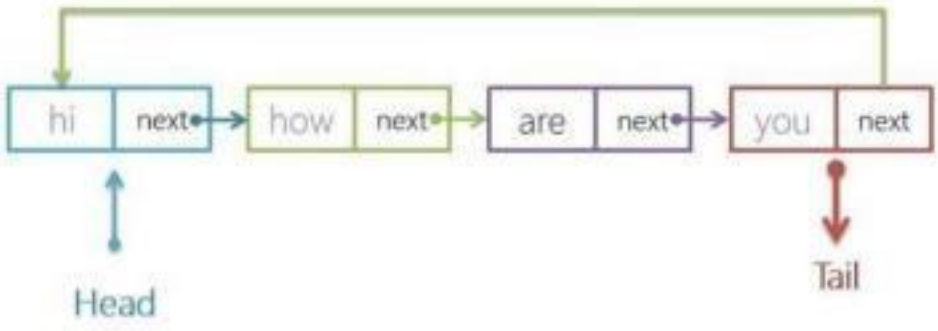
**6. Tampilkan Data Linked List**

```
        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) {
```

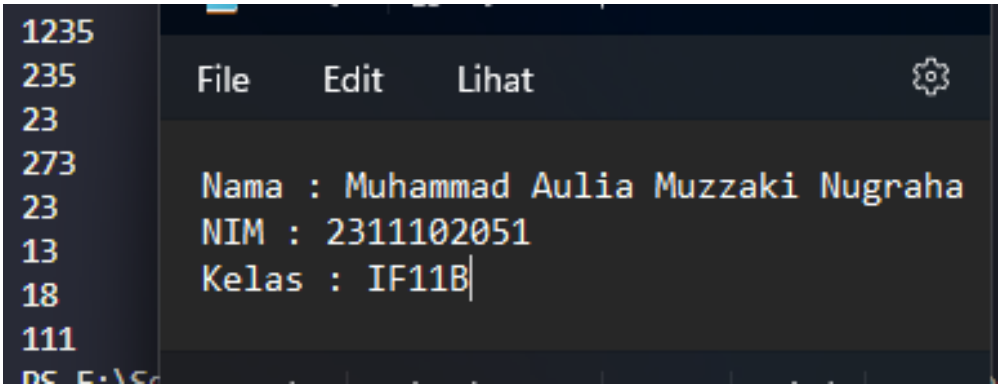
```
                sebelum = bantu;
            }
            if (nomor == posisi) {
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
        }
        sebelum->next = bantu;
        delete hapus;
    }
}

// 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-tama, di dalam fungsi main(), program diinisialisasi dengan memanggil fungsi init(), yang menginisialisasi pointer kepala dan ekor listmenjadi NULL. Selanjutnya, dilakukan beberapa operasi pada linked listseperti memasukkan node di depan, di belakang, dan di tengah menggunakan fungsi insertDepan(), insertBelakang(), dan insertTengah(), menghapus node di depan, di belakang, dan di tengah menggunakan fungsi hapusDepan(), hapusBelakang(), dan hapusTengah(), serta mengubah nilai node di depan, di belakang, dan di tengah menggunakan fungsi ubahDepan(), ubahBelakang(), dan ubahTengah(). Setiap operasi pada linked list diikuti dengan pemanggilan fungsi tampil() untuk menampilkan isi dari linked list tersebut. Akhirnya, program selesai dengan mengembalikan nilai 0. Struktur linked list yang diimplementasikan adalah singly linked list, di mana 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;
```

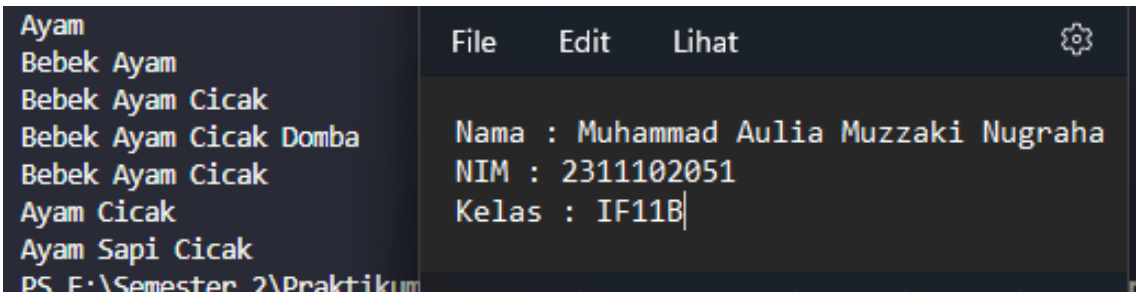
```
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

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 terdiri dari beberapa fungsi dasar, termasuk fungsi untuk menambahkan node di depan, di belakang, dan di tengah, menghapus node di depan, di belakang, dan di tengah, membersihkan seluruh isi list, serta menampilkan isi list. Pertama-tama, di dalam fungsi main(), program dimulai dengan pemanggilan fungsi init() untuk menginisialisasi pointer kepala dan ekor list menjadi NULL. Selanjutnya, dilakukan beberapa operasi pada circular linked list seperti memasukkan node di depan dan di belakang menggunakan fungsi insertDepan() dan insertBelakang(), menghapus node di depan dan di 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 pemanggilan 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 <iomanip>
#include <string>
using namespace std;

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

class LinkedList {
private:
    Mahasiswa* head;

public:
    LinkedList() {
        head = nullptr;
    }

    // Menambahkan data mahasiswa di depan
    void tambahDepan(string nama, string nim) {
        Mahasiswa* newNode = new Mahasiswa();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = head;
        head = newNode;
```

```
        cout << "Data telah ditambahkan\n";
    }

    // Menambahkan data mahasiswa di belakang
    void tambahBelakang(string nama, string nim) {
        Mahasiswa* newNode = new Mahasiswa();
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;
        if (head == nullptr) {
            head = newNode;
        } else {
            Mahasiswa* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
        cout << "Data telah ditambahkan\n";
    }

    // Menambahkan data mahasiswa di tengah berdasarkan posisi
    void tambahTengah(string nama, string nim, int posisi) {
        if (posisi < 1) {
            cout << "Posisi tidak valid\n";
            return;
        }
        Mahasiswa* newNode = new Mahasiswa();
        newNode->nama = nama;
        newNode->nim = nim;
        if (posisi == 1) {
            newNode->next = head;
            head = newNode;
        } else {
```

```
            Mahasiswa* temp = head;
            for (int i = 1; i < posisi - 1; i++) {
                if (temp == nullptr) {
                    cout << "Posisi tidak valid\n";
                    return;
                }
                temp = temp->next;
            }
            newNode->next = temp->next;
            temp->next = newNode;
        }
        cout << "Data telah ditambahkan\n";
    }

    // Mengubah data mahasiswa di depan
    void ubahDepan(string nama, string nim) {
        if (head == nullptr) {
            cout << "Linked List kosong\n";
            return;
        }
        head->nama = nama;
        head->nim = nim;
        cout << "Data di depan telah diubah\n";
    }

    // Mengubah data mahasiswa di belakang
    void ubahBelakang(string nama, string nim) {
        if (head == nullptr) {
            cout << "Linked List kosong\n";
            return;
        }
        Mahasiswa* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
```

```
        }
        temp->nama = nama;
        temp->nim = nim;
        cout << "Data di belakang telah diubah\n";
    }

    // Mengubah data mahasiswa di tengah berdasarkan posisi
    void ubahTengah(string nama, string nim, int posisi) {
        if (posisi < 1) {
            cout << "Posisi tidak valid\n";
            return;
        }
        if (head == nullptr) {
            cout << "Linked List kosong\n";
            return;
        }
        Mahasiswa* temp = head;
        for (int i = 1; i < posisi; i++) {
            if (temp == nullptr) {
                cout << "Posisi tidak valid\n";
                return;
            }
            temp = temp->next;
        }
        temp->nama = nama;
        temp->nim = nim;
        cout << "Data di posisi " << posisi << " telah diubah\n";
    }

    // Menghapus data mahasiswa di depan
    void hapusDepan() {
        if (head == nullptr) {
            cout << "Linked List kosong\n";
            return;
```

```
        }
        Mahasiswa* temp = head;
        head = head->next;
        delete temp;
        cout << "Data di depan telah dihapus\n";
    }

    // Menghapus data mahasiswa di belakang
    void hapusBelakang() {
        if (head == nullptr) {
            cout << "Linked List kosong\n";
            return;
        }
        if (head->next == nullptr) {
            delete head;
            head = nullptr;
            cout << "Data di belakang telah dihapus\n";
            return;
        }
        Mahasiswa* temp = head;
        while (temp->next->next != nullptr) {
            temp = temp->next;
        }
        delete temp->next;
        temp->next = nullptr;
        cout << "Data di belakang telah dihapus\n";
    }

    // Menghapus data mahasiswa di tengah berdasarkan posisi
    void hapusTengah(int posisi) {
        if (posisi < 1) {
            cout << "Posisi tidak valid\n";
            return;
        }
```

```
        if (head == nullptr) {
            cout << "Linked List kosong\n";
            return;
        }
        Mahasiswa* temp = head;
        if (posisi == 1) {
            head = head->next;
            delete temp;
            cout << "Data di posisi " << posisi << " telah
dihapus\n";
            return;
        }
        for (int i = 1; i < posisi - 1; i++) {
            if (temp == nullptr || temp->next == nullptr) {
                cout << "Posisi tidak valid\n";
                return;
            }
            temp = temp->next;
        }
        Mahasiswa* nextNode = temp->next->next;
        delete temp->next;
        temp->next = nextNode;
        cout << "Data di posisi " << posisi << " telah dihapus\n";
    }

    void tampilkanData() {
        if (head == nullptr) {
            cout << "Linked List kosong\n";
            return;
        }
        cout << "DATA MAHASISWA\n";
        cout << setw(20) << left << "NAMA" << setw(20) << left <<
"NIM" << endl;
        Mahasiswa* temp = head;
        while (temp != nullptr) {
```

```
            cout << setw(20) << left << temp->nama << setw(20) <<
left << temp->nim << endl;
            temp = temp->next;
        }
    }

    // Menghapus semua data mahasiswa
    void hapusList() {
        while (head != nullptr) {
            Mahasiswa* temp = head;
            head = head->next;
            delete temp;
        }
        cout << "Semua data mahasiswa telah dihapus\n";
    }
};

int main() {
    LinkedList linkedList;
    int pilihan;
    string nama, nim;
    int posisi;
    while (true) {
        cout << "\n  PROGRAM DATA MAHASISWA\n";
        cout << "--------------------------" << endl;
        cout << endl ;
        cout << "1. Tambah Depan\n";
        cout << "2. Tambah Belakang\n";
        cout << "3. Tambah Tengah\n";
        cout << "4. Ubah Depan\n";
        cout << "5. Ubah Belakang\n";
        cout << "6. Ubah Tengah\n";
        cout << "7. Hapus Depan\n";
        cout << "8. Hapus Belakang\n";
```

```
        cout << "9. Hapus Tengah\n";
        cout << "10. Hapus List\n";
        cout << "11. TAMPILKAN\n";
        cout << "0. KELUAR\n";

        cout << endl;

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

        cout << endl;

        switch (pilihan) {
            case 1:
                cout << "-Tambah Depan\n";
                cout << endl;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                linkedList.tambahDepan(nama, nim);
                break;
            case 2:
                cout << "-Tambah Belakang\n";
                cout << endl;
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                linkedList.tambahBelakang(nama, nim);
                break;
            case 3:
                cout << "-Tambah Tengah\n";
                cout << endl;
```

```
                cout << "Masukkan Nama : ";
                cin >> nama;
                cout << "Masukkan NIM : ";
                cin >> nim;
                cout << "Masukkan Posisi : ";
                cin >> posisi;
                linkedList.tambahTengah(nama, nim, posisi);
                break;
            case 4:
                cout << "-Ubah Depan\n";
                cout << endl;
                cout << "Masukkan Nama Baru : ";
                cin >> nama;
                cout << "Masukkan NIM Baru : ";
                cin >> nim;
                linkedList.ubahDepan(nama, nim);
                break;
            case 5:
                cout << "-Ubah Belakang\n";
                cout << endl;
                cout << "Masukkan Nama Baru : ";
                cin >> nama;
                cout << "Masukkan NIM Baru : ";
                cin >> nim;
                linkedList.ubahBelakang(nama, nim);
                break;
            case 6:
                cout << "-Ubah Tengah\n";
                cout << endl;
                cout << "Masukkan Nama Baru : ";
                cin >> nama;
                cout << "Masukkan NIM Baru : ";
                cin >> nim;
                cout << "Masukkan Posisi : ";
```

```
                cin >> posisi;
                linkedList.ubahTengah(nama, nim, posisi);
                break;
            case 7:
                cout << "-Hapus Depan\n";
                cout << endl;
                linkedList.hapusDepan();
                break;
            case 8:
                cout << "-Hapus Belakang\n";
                cout << endl;
                linkedList.hapusBelakang();
                break;
            case 9:
                cout << "-Hapus Tengah\n";
                cout << endl;
                cout << "Masukkan Posisi : ";
                cin >> posisi;
                linkedList.hapusTengah(posisi);
                break;
            case 10:
                cout << "-Hapus List\n";
                cout << endl;
                linkedList.hapusList();
                break;
            case 11:
                cout << "-Tampilkan\n";
                cout << endl;
                linkedList.tampilkanData();
                break;
            case 0:
                exit(0);
            default:
                cout << "Pilihan tidak valid\n";
```

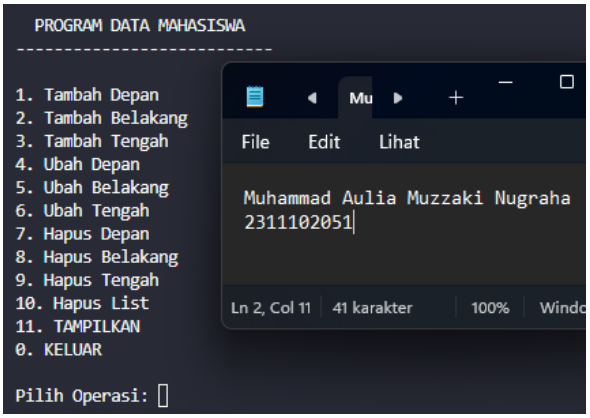
```
            }
        }


        return 0;
}
```

**Deskripsi program & Screenshot Program**

Linked list yang dibuat memiliki data mahasiswa yang terdiri dari nama dan NIM. Dalam kelas LinkedList, terdapat beberapa metode untuk melakukan operasi pada linked list seperti menambahkan data di depan (tambahDepan()), di belakang (tambahBelakang()), atau di tengah berdasarkan posisi (tambahTengah()), mengubah data di depan (ubahDepan()), di belakang (ubahBelakang()), atau di tengah berdasarkan posisi (ubahTengah()), menghapus data di depan (hapusDepan()), di belakang (hapusBelakang()), atau di tengah berdasarkan posisi (hapusTengah()), menampilkan seluruh data mahasiswa (tampilkanData()), dan menghapus seluruh 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

```
Pilih Operasi: 1

-Tambah Depan

Masukkan Nama : z
Masukkan NIM : 1111
Data telah ditambahkan
```

```
File    Edit    Lihat

Muhammad Aulia Muzzaki Nugraha
2311102051
```

```
Pilih Operasi: 2

-Tambah Belakang

Masukkan Nama : a
Masukkan NIM : 2222
Data telah ditambahkan
```

```
Pilih Operasi: 3

-Tambah Tengah

Masukkan Nama : k
Masukkan NIM : 3333
Masukkan Posisi : 2
Data telah ditambahkan
```

- Operasi Hapus

```
Pilih Operasi: 7

-Hapus Depan

Data di depan telah dihapus
```

```
Pilih Operasi: 8

-Hapus Belakang

Data di belakang telah dihapus
```

```
-Hapus Tengah

Masukkan Posisi : 2
Data di posisi 2 telah dihapus
```

- Operasi ubah

```
-Ubah Depan

Masukkan Nama Baru : H
Masukkan NIM Baru : 1212
Data di depan telah diubah
```

```
-Ubah Tengah

Masukkan Nama Baru : J
Masukkan NIM Baru : 777
Masukkan Posisi : 2
Data di posisi 2 telah diubah
```

- Operasi tampil data

```
-Tampilkan

DATA MAHASISWA
NAMA                    NIM
H                       1212
J                       777
```

2. Masukkan data sesuai urutan

```
Pilih Operasi: 11

-Tampilkan

DATA MAHASISWA
NAMA                    NIM
Jawad                   23300001
Zaki                    2311102051
Farrel                  23300003
Denis                   23300005
Anis                    23300008
Bowo                    23300015
Gahar                   23300040
Udin                    23300048
Ucok                    23300050
Budi                    23300099
```

3. Lakukan Perintah

```
-Tambah Tengah

Masukkan Nama : Wati
Masukkan NIM : 2330004
Masukkan Posisi : 3
Data telah ditambahkan
```

```
-Hapus Tengah

Masukkan Posisi : 5
Data di posisi 5 telah dihapus
```

```
-Tambah Depan

Masukkan Nama : Owi
Masukkan NIM : 2330000
Data telah ditambahkan
```

```
-Tambah Belakang

Masukkan Nama : David
Masukkan NIM : 23300100
Data telah ditambahkan
```

```
-Ubah Tengah

Masukkan Nama Baru : Idin
Masukkan NIM Baru : 23300045
Masukkan Posisi : 9
Data di posisi 9 telah diubah
```

```
-Ubah Belakang

Masukkan Nama Baru : Lucy
Masukkan NIM Baru : 23300101
Data di belakang telah diubah
```

```
-Hapus Depan

Data di depan telah dihapus
```

```
-Ubah Depan

Masukkan Nama Baru : Bagas
Masukkan NIM Baru : 2330002
Data di depan telah diubah
```

```
Pilih Operasi: 8

-Hapus Belakang

Data di belakang telah dihapus
```

```
DATA MAHASISWA
NAMA                    NIM
Bagas                   2330002
Zaki                    2311102051
Wati                    2330004
Farrel                  23300003
Anis                    23300008
Bowo                    23300015
Gahar                   23300040
Idin                    23300045
Ucok                    23300050
Budi                    23300099
```

# BAB IV
# KESIMPULAN

Linked list adalah struktur data yang terdiri dari sejumlah simpul yang terhubung, di mana setiap simpul memiliki dua bagian: data yang disimpan dan referensi ke simpul berikutnya. Dua jenis linked list yang umum digunakan adalah circular linked list dan non-circular linked list. Circular linked list memiliki sifat bahwa simpul terakhir dalam list memiliki referensi yang kembali ke simpul pertama, membentuk lingkaran atau cincin. Sementara non-circular linked list, seperti namanya, tidak memiliki referensi kembali ke simpul pertama, sehingga simpul terakhir menunjuk ke nullptr.

Kedua jenis linked list memiliki kegunaan dan kelebihan masing-masing:

a. Non-circular linked list lebih umum digunakan dan lebih mudah untuk diimplementasikan. Mereka cocok untuk banyak aplikasi dan memungkinkan operasi seperti penambahan dan penghapusan simpul dengan mudah.
b. Circular linked list sering digunakan ketika kita perlu mengakses elemen list dalam lingkaran terus menerus, atau ketika kita membutuhkan penyelesaian siklus dalam algoritma.

Dalam pengimplementasiannya, circular linked list memerlukan penanganan khusus saat menambahkan atau menghapus simpul, terutama untuk mengatur referensi yang menutupi lingkaran.

# DAFTAR PUSTAKA

Assiten Praktikum. Modul 4. Learning Management System. Diakses pada 10 April 2024.

Meidyan Permata Pitri, Guntoro Barovih, Rezania Agramaninstari Azdy, Yuniansyah, Andri Saputra, Yesi Sriyeni, Arsia Rini, Fadhila Tangguh Admojo. Algoritma dan Struktur Data. Widina Bhakti Persada, 2022. ISBN: 93-623-459-182-8