

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

MODUL 3



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

SINGLE AND DOUBLE LINKED LIST

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.

Dalam Linked List, data disimpan dalam bentuk simpul atau node yang saling terhubung satu sama lain dengan menggunakan referensi atau alamat dari simpul selanjutnya dalam urutan. Setiap simpul berisi dua hal penting: data yang ingin kita simpan dan alamat referensi ke simpul berikutnya dalam urutan. Dengan begitu, setiap simpul dapat menyimpan data dan mengetahui alamat simpul selanjutnya.

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.

a) Single Linked List

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Setiap elemen dalam linked list dihubungkan ke elemen lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau node atau verteks. Pointer adalah alamat elemen.

Setiap simpul pada dasarnya dibagi atas dua bagian pertama disebut bagian isi atau informasi atau data yang berisi nilai yang disimpan oleh simpul. Bagian kedua disebut bagian pointer yang berisi alamat dari node berikutnya atau sebelumnya.

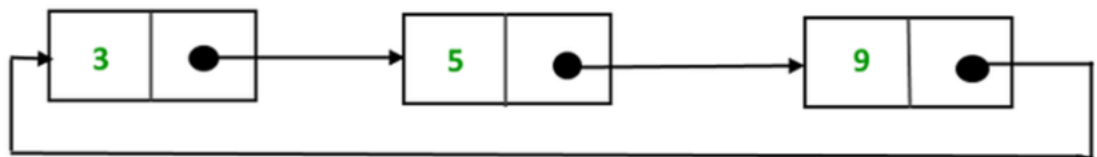
Dengan menggunakan struktur seperti ini, linked list dibentuk dengan cara menunjuk pointer next suatu elemen ke elemen yang mengikutinya. Pointer next pada elemen terakhir merupakan NULL, yang menunjukkan akhir dari suatu list.



Sumber: simplilearn.com

Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List umumnya lebih efisien dalam penggunaan memori dibandingkan dengan jenis Linked List lainnya, seperti Double Linked List dan Circular Linked List.

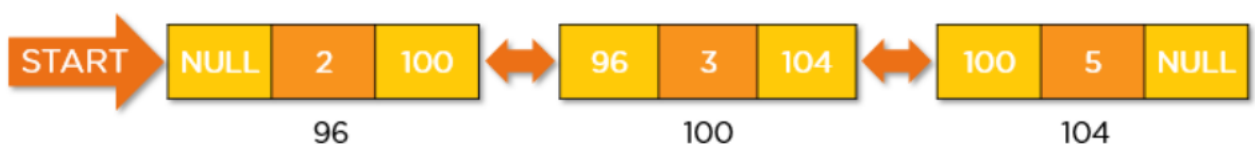
Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.



b) Double linked list

Double Linked List adalah struktur data Linked List yang mirip dengan Single Linked List, namun dengan tambahan satu pointer tambahan pada setiap simpul yaitu pointer prev yang menunjuk ke simpul sebelumnya. Dengan adanya pointer prev, Double Linked List memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul mana saja secara efisien. Jadi, kita bisa melintasinya secara dua arah. Tidak seperti singly linked list, simpul doubly linked list berisi satu pointer tambahan yang disebut previous pointer. Pointer ini menunjuk ke simpul sebelumnya.

Setiap simpul pada Double Linked List memiliki tiga elemen penting, yaitu elemen data (biasanya berupa nilai), pointer next yang menunjuk ke simpul berikutnya, dan pointer prev yang menunjuk ke simpul sebelumnya.



Di dalam sebuah linked list, ada 2 pointer yang menjadi penunjuk utama, yakni pointer HEAD yang menunjuk pada node pertama di dalam linked list itu sendiri dan pointer TAIL yang menunjuk pada node paling akhir di dalam linked list. Sebuah linked list dikatakan kosong apabila isi pointer head adalah NULL. Selain itu, nilai pointer prev dari HEAD selalu NULL, karena merupakan data pertama. Begitu pula dengan pointer next dari TAIL yang selalu bernilai NULL sebagai penanda data terakhir.

BAB II

GUIDED

LATIHAN – GUIDED

1. Guided 1

a) Latihan single linked list

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

// Tambah Node di posisi 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;
            delete hapus;
        } else {
            head = tail = NULL;
            delete hapus;
        }
    } else {
        cout << "List kosong!" << endl;
    }
}

```

```

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

// Hapus Node di posisi 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* hapus;
        Node* bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++)
        {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
    }
}

```



```

        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 posisi tengah
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;
            for (int nomor = 1; nomor < posisi; nomor++)
            {
                bantu = bantu->next;
            }
            bantu->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;
    }
}

// Hapus semua Node di 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 semua data Node di list
void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != NULL) {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

int main() {

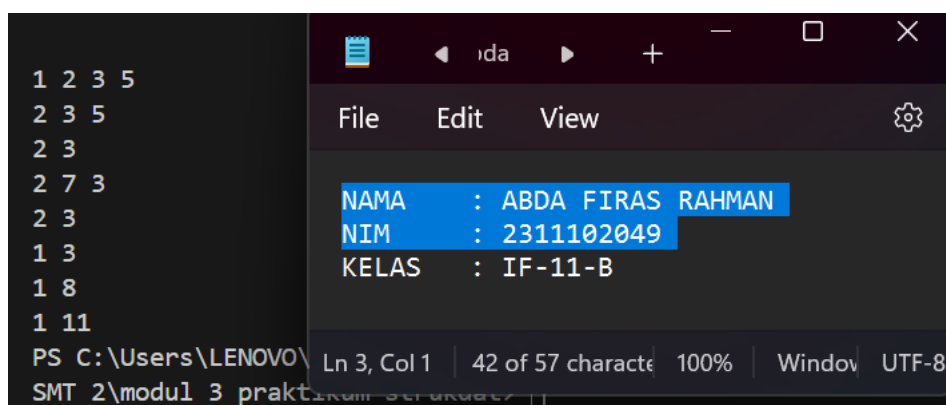
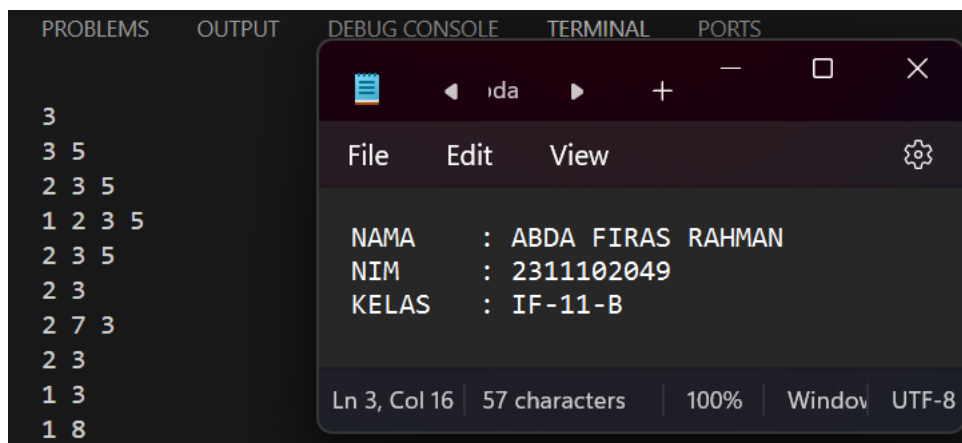
```

```

        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;
    }

```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Pada pemograman ini menggambarkan penggunaan dasar dari linked list dalam bahasa pemrograman C++. Setiap bagian dari program tersebut digunakan untuk melakukan operasi-operasi dasar terkait linked list dan pada bagian ini juga mendeklarasikan sebuah 'struct node' yang memiliki dua anggota yaitu pada 'int data' untuk menyimpan nilai data dan pada 'node* next' yang merupakan bagian pointer menunjuk pada node berikutnya. Serta bagian fungsi 'main()' bagian utama dari program yang melakukan fungsi fungsi yang telah diimplementasikan yang mencakup 'insertDepan()', 'insertBelakang()' dan lain lain.

.

2. Guided 2.

b) Double linked list

Source code

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

class Node {
public:
    int data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void push(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }
    }
}
```

```

        head = newNode;
    }

    void pop() {
        if (head == nullptr) {
            return;
        }
        Node* temp = head;
        head = head->next;

        if (head != nullptr) {
            head->prev = nullptr;
        } else {
            tail = nullptr;
        }

        delete temp;
    }

    bool update(int oldData, int newData) {
        Node* current = head;

        while (current != nullptr) {
            if (current->data == oldData) {
                current->data = newData;
                return true;
            }
            current = current->next;
        }
        return false;
    }

    void deleteAll() {
        Node* current = head;
        while (current != nullptr) {

```

```

        Node* temp = current;
        current = current->next;
        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}

void display() {
    Node* current = head;
    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}

};

int main() {
    DoublyLinkedList list;
    while (true) {
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Exit" << endl;

        int choice;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                int data;

```

```

        cout << "Enter data to add: ";
        cin >> data;
        list.push(data);
        break;
    }
    case 2: {
        list.pop();
        break;
    }
    case 3: {
        int oldData, newData;
        cout << "Enter old data: ";
        cin >> oldData;
        cout << "Enter new data: ";
        cin >> newData;
        bool updated = list.update(oldData,
newData);

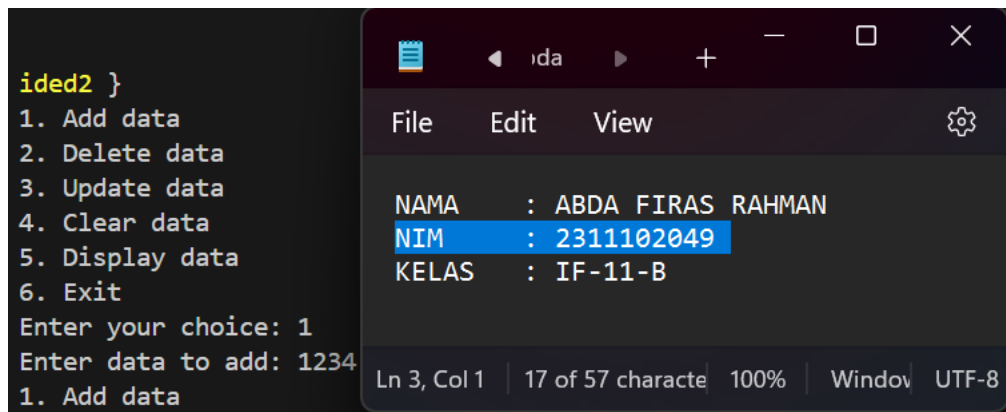
        if (!updated) {
            cout << "Data not found" << endl;
        }
        break;
    }
    case 4: {
        list.deleteAll();
        break;
    }
    case 5: {
        list.display();
        break;
    }
    case 6: {
        return 0;
    }
    default: {
        cout << "Invalid choice" << endl;

```



```
        break;
    }
}
}
return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Pada program c++ ini mengimplementasikan double linked list dimana memiliki beberapa metode, termasuk push, pop, update, deleteAll, dan display. Metode push digunakan untuk menambahkan data ke dalam list, pop digunakan untuk menghapus data terakhir dari list, update digunakan untuk mengubah data yang sudah ada, deleteAll digunakan untuk menghapus semua data dari list, dan display digunakan untuk menampilkan data yang ada di list. Program ini juga digunakan untuk memanipulasi data dalam list.

A. UNGUIDED

1. *Soal mengenai Single Linked List*

Buatlah program menu Single Linked List Non-Circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user. Lakukan operasi berikut:

- a. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah). Data pertama yang dimasukkan adalah nama dan usia anda.

[Nama_anda] [Usia_anda]

John 19

Jane 20

Michael 18

Yusuke 19

Akechi 20

Hoshino 18

Karin 18

- b. Hapus data Akechi
- c. Tambahkan data berikut diantara John dan Jane : Futaba 18
- d. Tambahkan data berikut diawal : Igor 20
- e. Ubah data Michael menjadi : Reyn 18
- f. Tampilkan seluruh data

2. *Soal mengenai Double Linked List*

Modifikasi Guided Double Linked List dilakukan dengan penambahan operasi untuk menambah data, menghapus, dan update di tengah / di urutan tertentu yang diminta. Selain itu, buatlah agar tampilannya

menampilkan Nama produk dan harga.

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Skintific	100.000

Wardah	50.000
Hanasui	30.000

Case:

1. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific
2. Hapus produk wardah
3. Update produk Hanasui menjadi Cleora dengan harga 55.000
4. Tampilkan menu seperti dibawah ini

Toko Skincare Purwokerto

1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit

Pada menu 7, tampilan akhirnya akan menjadi seperti dibawah ini :

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Azarine	65.000
Skintific	100.000
Cleora	55.000

JAWABAN UNGUIDED 1

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

// Struktur node untuk menyimpan data mahasiswa
struct Mahasiswa {
    string nama;
    int usia;
    Mahasiswa* next;
};

// Kelas untuk linked list
class LinkedList {
private:
    Mahasiswa* head;

public:
    // Konstruktor
    LinkedList() {
        head = NULL;
    }

    // Fungsi untuk memasukkan data mahasiswa di depan
    linked list
    void insertDepan(string nama, int usia) {
        Mahasiswa* newNode = new Mahasiswa();
```

```

        newNode->nama = nama;
        newNode->usia = usia;
        newNode->next = head;
        head = newNode;
    }

    // Fungsi untuk memasukkan data mahasiswa di belakang
    linked list
    void insertBelakang(string nama, int usia) {
        Mahasiswa* newNode = new Mahasiswa();
        newNode->nama = nama;
        newNode->usia = usia;
        newNode->next = NULL;

        if (head == NULL) {
            head = newNode;
        } else {
            Mahasiswa* temp = head;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }

    // Fungsi untuk memasukkan data mahasiswa di tengah
    linked list setelah node tertentu
    void insertTengah(string nama, int usia, string
nama_setelah) {
        Mahasiswa* newNode = new Mahasiswa();
        newNode->nama = nama;
        newNode->usia = usia;

        Mahasiswa* temp = head;
        while (temp != NULL && temp->nama != nama_setelah)
        {
            temp = temp->next;
        }
    }

```

```

        if (temp != NULL) {
            newNode->next = temp->next;
            temp->next = newNode;
        } else {
            cout << "Data " << nama_setelah << " tidak
ditemukan" << endl;
        }
    }

// Fungsi untuk menghapus data mahasiswa
void hapus(string nama) {
    Mahasiswa* temp = head;
    Mahasiswa* prev = NULL;

    if (temp != NULL && temp->nama == nama) {
        head = temp->next;
        delete temp;
        return;
    }

    while (temp != NULL && temp->nama != nama) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) {
        cout << "Data " << nama << " tidak ditemukan"
<< endl;
        return;
    }

    prev->next = temp->next;
    delete temp;
}

// Fungsi untuk menampilkan seluruh data mahasiswa
void tampilkanData() {

```

```

        Mahasiswa* temp = head;
        while (temp != NULL) {
            cout << temp->nama << " " << temp->usia <<
endl;
            temp = temp->next;
        }
    }
};

int main() {
    LinkedList linkedList;

    // Memasukkan data sesuai dengan urutan yang diminta
    linkedList.insertDepan("Abda", 19);
    linkedList.insertBelakang("John", 19);
    linkedList.insertBelakang("Jane", 20);
    linkedList.insertBelakang("Michael", 18);
    linkedList.insertBelakang("Yusuke", 19);
    linkedList.insertBelakang("Akechi", 20);
    linkedList.insertBelakang("Hoshino", 18);
    linkedList.insertBelakang("Karin", 18);

    // Menghapus data Akechi
    linkedList.hapus("Akechi");

    // Menambahkan data Futaba di antara John dan Jane
    linkedList.insertTengah("Futaba", 18, "John");

    // Menambahkan data Igor di awal
    linkedList.insertDepan("Igor", 20);

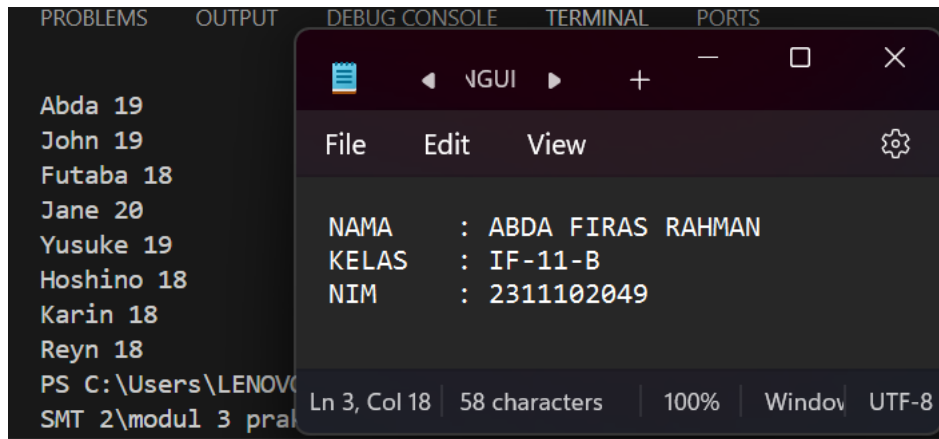
    // Mengubah data Michael menjadi Reyn
    linkedList.hapus("Michael");
    linkedList.insertBelakang("Reyn", 18);

    // Menampilkan seluruh data
    linkedList.tampilkanData();
}

```

```
    return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Pada program ini menggunakan ListMahasisw yang memiliki metode untuk menambahkan data, menghapus data, mengubah data, dan menampilkan seluruh data menggunakan metode insertDepan, insertBelakang, dan insertTengah. Untuk menghapus data menggunakan deleteNode, Untuk menampilkan seluruh data menggunakan metode display.

JAWABAN UNGUIDED 2

```
#include <iostream>

#include <iomanip>

using namespace std;

class Node

{

public:

    string namaProduk;

    int harga;

    Node *prev;
```



```

        Node *next;

};

class DoublyLinkedList
{
public:
    Node *head;
    Node *tail;

    DoublyLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }

    void push(string namaProduk, int harga)
    {
        Node *newNode = new Node;

        newNode->namaProduk = namaProduk;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr)
        {
            head->prev = newNode;

```

```

        }

        else

        {

            tail = newNode;

        }

        head = newNode;

    }

    void pushCenter(string namaProduk, int harga, int
posisi)

    {

        if (posisi < 0)

        {

            cout << "Posisi harus bernilai non-
negatif." << endl;

            return;

        }

        Node *newNode = new Node;

        newNode->namaProduk = namaProduk;

        newNode->harga = harga;

        if (posisi == 0 || head == nullptr)

        {

            newNode->prev = nullptr;

```

```
        newNode->next = head;

        if (head != nullptr)
        {
            head->prev = newNode;
        }
        else
        {
            tail = newNode;
        }

        head = newNode;
    }
    else
    {
        Node *temp = head;

        int count = 0;

        while (temp != nullptr && count < posisi)
        {
            temp = temp->next;

            count++;
        }

        if (temp == nullptr)
        {
            newNode->prev = tail;

            newNode->next = nullptr;
        }
    }
}
```

```

        tail->next = newNode;

        tail = newNode;

    }

    else

    {

        newNode->prev = temp->prev;

        newNode->next = temp;

        temp->prev->next = newNode;

        temp->prev = newNode;

    }

}

}

void pop()

{

    if (head == nullptr)

    {

        return;

    }

    Node *temp = head;

    head = head->next;

    if (head != nullptr)

    {

        head->prev = nullptr;

    }

}

```

```
        else

        {

            tail = nullptr;

        }

        delete temp;

    }

    void popCenter(int posisi)

    {

        if (head == nullptr)

        {

            cout << "List kosong. Tidak ada yang bisa
dihapus." << endl;

            return;

        }

        if (posisi < 0)

        {

            cout << "Posisi harus bernilai non-
negatif." << endl;

            return;

        }

        if (posisi == 0)

        {
```

```
        Node *temp = head;

        head = head->next;

        if (head != nullptr)
        {
            head->prev = nullptr;
        }
        else
        {
            tail = nullptr;
        }

        delete temp;
    }
    else
    {
        Node *temp = head;

        int count = 0;

        while (temp != nullptr && count < posisi)
        {
            temp = temp->next;

            count++;
        }

        if (temp == nullptr)
        {
```

```

        cout << "Posisi melebihi ukuran list.
Tidak ada yang dihapus." << endl;

        return;

    }

    if (temp == tail)
    {
        tail = tail->prev;
        tail->next = nullptr;
        delete temp;
    }
    else
    {
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
        delete temp;
    }
}

bool update(string oldNamaProduk, string
newNamaProduk, int newHarga)
{
    Node *current = head;

    while (current != nullptr)

```

```

        {

            if (current->namaProduk == oldNamaProduk)

            {

                current->namaProduk = newNamaProduk;

                current->harga = newHarga;

                return true;

            }

            current = current->next;

        }

        return false;

    }

```

```

bool    updateCenter(string    newNamaProduk,    int
newHarga, int posisi)

{

    if (head == nullptr)

    {

        cout << "List kosong. Tidak ada yang dapat
diperbarui." << endl;

        return false;

    }

    if (posisi < 0)

    {

        cout    <<    "Posisi    harus    bernilai    non-
negatif." << endl;

        return false;

    }

```



```

    }

    Node *current = head;

    int count = 0;

    while (current != nullptr && count < posisi)
    {
        current = current->next;

        count++;
    }

    if (current == nullptr)
    {
        cout << "Posisi melebihi ukuran list. Tidak
ada yang diperbarui." << endl;

        return false;
    }

    current->namaProduk = newNamaProduk;

    current->harga = newHarga;

    return true;
}

void deleteAll()
{
    Node *current = head;

```

```

        while (current != nullptr)
        {
            Node *temp = current;

            current = current->next;

            delete temp;
        }

        head = nullptr;
        tail = nullptr;
    }

    void display()
    {
        if (head == nullptr)
        {
            cout << "List kosong." << endl;

            return;
        }

        Node *current = head;

        cout << setw(37) << setfill('-') << "-" <<
setfill(' ') << endl;

        cout << "| " << setw(20) << left << "Nama Produk"
            << " | " << setw(10) << "Harga"
            << " |" << endl;

        cout << setw(37) << setfill('-') << "-" <<

```

```

        setfill(' ') << endl;

        while (current != nullptr)
        {
            cout << "| " << setw(20) << left << current-
>namaProduk << " | " << setw(10) << current->harga << "
|" << endl;

            current = current->next;
        }

        cout << setw(37) << setfill('-') << "-" <<
setfill(' ') << endl;

    }

};

int main()
{
    DoublyLinkedList list;

    int choice;

    cout << endl

        << "Toko Skincare Mbak Najwa Purwokerto" <<
endl;

    do
    {
        cout << "1. Tambah data" << endl;

        cout << "2. Hapus data" << endl;

        cout << "3. Update data" << endl;

        cout << "4. Tambah Data Urutan Tertentu" <<

```

```
endl;

    cout << "5. Hapus Data Urutan Tertentu" << endl;

    cout << "6. Hapus Seluruh Data" << endl;

    cout << "7. Tampilkan data" << endl;

    cout << "8. Exit" << endl;


    cout << "Pilihan : ";

    cin >> choice;


    switch (choice)
    {
    case 1:
    {
        string namaProduk;

        int harga;

        cout << "Masukkan nama produk: ";

        cin.ignore();

        getline(cin, namaProduk);

        cout << "Masukkan harga produk: ";

        cin >> harga;

        list.push(namaProduk, harga);

        break;
    }

    case 2:
    {

        list.pop();
```

```

        break;

    }

    case 3:

    {

        string newNamaProduk;

        int newHarga, posisi;

        cout << "Masukkan posisi produk: ";

        cin >> posisi;

        cout << "Masukkan nama baru produk: ";

        cin >> newNamaProduk;

        cout << "Masukkan harga baru produk: ";

        cin >> newHarga;

        bool                updatedCenter                =
list.updateCenter(newNamaProduk, newHarga, posisi);

        if (!updatedCenter)

        {

            cout << "Data not found" << endl;

        }

        break;

    }

    case 4:

    {

        string namaProduk;

        int harga, posisi;

        cout << "Masukkan posisi data produk: ";

        cin >> posisi;

```

```
        cout << "Masukkan nama produk: ";

        cin.ignore();

        getline(cin, namaProduk);

        cout << "Masukkan harga produk: ";

        cin >> harga;

        list.pushCenter(namaProduk, harga, posisi);

        break;
    }

    case 5:

    {

        int posisi;

        cout << "Masukkan posisi data produk: ";

        cin >> posisi;

        list.popCenter(posisi);

        break;

    }

    case 6:

    {

        list.deleteAll();

        break;

    }

    case 7:

    {

        list.display();

        break;

    }
```

```
        case 8:

            {

                return 0;

            }

        default:

            {

                cout << "Invalid choice" << endl;

                break;

            }

        }

    } while (choice != 8);


    return 0;

}
```

SCREENSHOOT PROGRAM

```
"C:\Users\LENOVO\OneDrive\  ×  +  v

Toko Skincare Mbak Najwa Purwokerto
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 1
Masukkan nama produk: originate
Masukkan harga produk: 60000
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 1
Masukkan nama produk: somethinc
Masukkan harga produk: 150000
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 1
Masukkan nama produk: azarine
Masukkan harga produk: 65000
1. Tambah data
2. Hapus data
```

 iided + - □ ×

File Edit View ⚙

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

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

Toko Skincare Mbak Najwa Purwokerto

1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 1

Masukkan nama produk: originate

Masukkan harga produk: 60000

1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 1

Masukkan nama produk: somethinc

Masukkan harga produk: 150000


1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 1

Masukkan nama produk: azarine

Masukkan harga produk: 65000

1. Tambah data
2. Hapus data

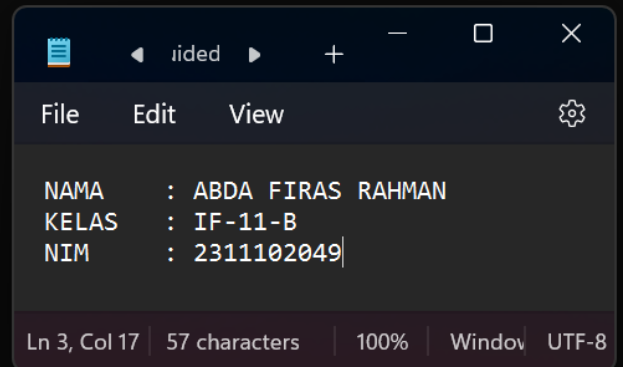
 iided + - □ ×

File Edit View ⚙

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

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

```
Masukkan nama produk: azarine
Masukkan harga produk: 65000
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 1
Masukkan nama produk: skintific
Masukkan harga produk: 100000
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 1
Masukkan nama produk: cleora
Masukkan harga produk: 50000
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 7
```



3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan : 1

Masukkan nama produk: cleora

Masukkan harga produk: 50000

1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

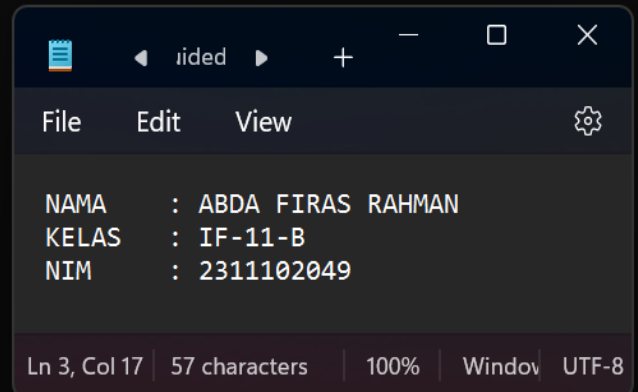
Pilihan : 7

Nama Produk	Harga

cleora	50000
skintific	100000
azarine	65000
somethinc	150000
originate	60000

1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit

Pilihan :



DESKRIPSI PROGRAM

Ini merupakan program dari double linked list untuk menyimpan list data skincare di toko mboten purwokerto. Sama seperti single linked list program ini memiliki beberapa menu diantaranya tambah data, hapus data, update data, tambah data urutan tertentu, hapus data urutan tertentu, hapus seluruh data dan tampilkan data.

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.

DAFTAR PUSTAKA

Pengertian Linked List: Struktur Data dalam Pemrograman :

<https://fikti.umsu.ac.id/pengertian-linked-list-struktur-data-dalam-pemrograman/>

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

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