

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL 3  
SINGLE AND DOUBLE LINKED LIST**



**Disusun Oleh:**

Muhamad ihsan

2311102077

**Dosen**

Wahyu Andi Saputra, S.pd., M,Eng

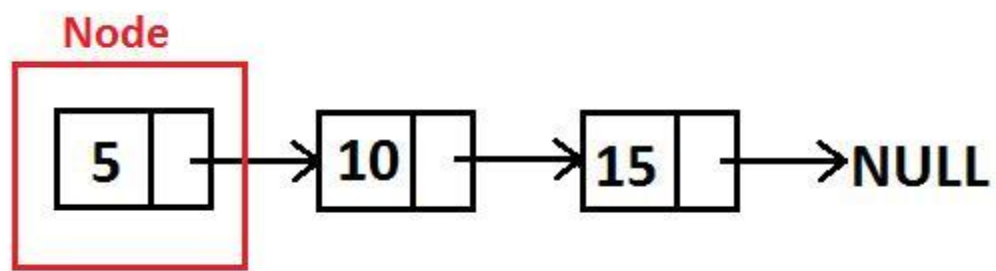
**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2024**

## A. Dasar Teori

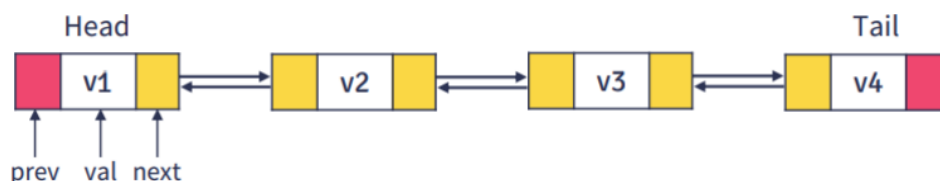
1. Single Linked List merupakan koleksi linear dari data, yang disebut sebagai nodes, dimana setiap node akan menunjuk pada node lain melalui sebuah pointer. Linked List dapat didefinisikan pula sebagai kumpulan nodes yang merepresentasikan sebuah sequence.

Representasi sebuah linked list dapat digambarkan melalui gambar di bawah ini:



Sebuah linked list yang hanya memiliki 1 penghubung ke node lain disebut sebagai single linked list. Di dalam sebuah linked list, ada 1 pointer yang menjadi gambaran besar, yakni pointer HEAD yang menunjuk pada node pertama di dalam linked list itu sendiri. Sebuah linked list dikatakan kosong apabila isi pointer head adalah NULL.

2. Double linked list - Double Linked List merupakan suatu linked list yang memiliki dua variabel pointer. Dimana pointer tersebut menunjuk ke node sebelum dan selanjutnya (prev & next). Double Linked List terdiri dari sejumlah elemen (node) dimana setiap node memiliki penunjuk prev (menunjuk node sebelumnya) dan next (menunjuk node selanjutnya). Penunjuk prev pada node head menunjuk ke NULL, menandakan bahwa node head (node awal). Penunjuk next pada node tail menunjuk ke NULL, menandakan bahwa node tail (node akhir).



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.

## B. GUIDED

### Guided 1 : Latihan Single Linked List

```
#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;
    }
}
}

```

Output:

```
edipertemuan3.cpp -o guidedipertemuan3 } ; if ($?) { .\guidedipertemuan3
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS C:\Users\ASUS\OneDrive\Desktop\praktikum struktur data\vscode>
```

Deskripsi:

Program di atas terdapat fungsi-fungsi dasar seperti inialisasi linked list, penambahan node di depan, di belakang, di tengah, penghapusan node di depan, di belakang, di tengah, serta pengubahan data pada node-node tertentu. Program dilengkapi dengan fungsi untuk menampilkan isi dari linked list serta membersihkan seluruh isi dari linked list.

## Guided 2 : Program Mencari Nilai Maksimal pada Array

```
#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;
}

```

## Output:

```

Enter your choice: 1
Enter data to add: 077
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 2
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 077
Enter new data: 001
Data not found
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 4
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5

```

### Deskripsi:

Program c++ juga terdapat fungsi-fungsi dasar untuk manipulasi data dalam Doubly Linked List, termasuk penambahan data, penghapusan data, pembaruan data, penghapusan seluruh data, serta menampilkan seluruh data yang tersimpan dalam Doubly Linked List. Selain itu, program ini juga menyediakan antar muka sederhana berbasis konsol untuk berinteraksi dengan Doubly Linked List melalui menu pilihan.

### C. UNGUIDED

1. 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

### SOURCE CODE:

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

struct Node
{
    string nama;
    int usia;
    Node *next;
};

Node *head;
Node *tail;

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

```
bool isEmpty()
{
    return head == NULL;
}

void insertDepan(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

void insertBelakang(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

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

```

        return jumlah;
    }

void insertTengah(string nama, int usia, 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->nama = nama;
        baru->usia = usia;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan()
{
    if (!isEmpty())
    {
        Node *hapus = head;
        if (head->next != NULL)
        {
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
            delete hapus;
        }
    }
    else
    {

```

```

        cout << "List kosong!" << endl;
    }
}

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;
    }
}

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;
    }
}

void ubahDepan(string nama, int usia)
{
    if (!isEmpty())
    {
        head->nama = nama;
        head->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

void ubahTengah(string nama, int usia, 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->nama = nama;
            bantu->usia = usia;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

```



```

void ubahBelakang(string nama, int usia)
{
    if (!isEmpty())
    {
        tail->nama = nama;
        tail->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

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

void tampil()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        while (bantu != NULL)
        {
            cout << bantu->nama << " ";
            cout << bantu->usia << " , ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{

```

```

init();
int menu, usia, posisi;
string nama;
cout << "\n# Menu Linked List Mahasiswa #" << endl;
do
{
    cout << "\n 1. Insert Depan"
        << "\n 2. Insert Belakang"
        << "\n 3. Insert Tengah"
        << "\n 4. Hapus Depan"
        << "\n 5. Hapus Belakang"
        << "\n 6. Hapus Tengah"
        << "\n 7. Ubah Depan"
        << "\n 8. Ubah Belakang"
        << "\n 9. Ubah Tengah"
        << "\n 10. Tampilkan"
        << "\n 0. Keluar Program"
        << "\n Pilihan : ";

    cin >> menu;
    switch (menu)
    {
        case 1:
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";
            cin >> usia;
            insertDepan(nama, usia);
            cout << endl;
            tampil();
            break;

        case 2:
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";
            cin >> usia;
            insertBelakang(nama, usia);
            cout << endl;
            tampil();
            break;

        case 3:
            cout << "Masukkan Posisi : ";
            cin >> posisi;
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";
            cin >> usia;
            insertTengah(nama, usia, posisi);
            cout << endl;
    }
}

```

```
tampil();
break;
case 4:
    hapusDepan();
    cout << endl;
    tampil();
    break;
case 5:
    hapusBelakang();
    cout << endl;
    tampil();
    break;
case 6:
    cout << "Masukkan Posisi : ";
    cin >> posisi;
    hapusTengah(posisi);
    cout << endl;
    tampil();
    break;
case 7:
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan Usia : ";
    cin >> usia;
    ubahDepan(nama, usia);
    cout << endl;
    tampil();
    break;
case 8:
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan Usia : ";
    cin >> usia;
    ubahBelakang(nama, usia);
    cout << endl;
    tampil();
    break;
case 9:
    cout << "Masukkan Posisi : ";
    cin >> posisi;
    cout << "Masukkan Nama : ";
    cin >> nama;
    cout << "Masukkan Usia : ";
    cin >> usia;
    ubahTengah(nama, usia, posisi);
    cout << endl;
    tampil();
    break;
```

```

        case 10:
            tampil();
            break;

        default:
            cout << "Pilihan Salah" << endl;
            break;
    }
} while (menu != 0);
return 0;
}

```

## OUTPUT:

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

```

# Menu Linked List Mahasiswa #
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 1
Masukkan Nama : IHSAN
Masukkan Usia : 18

IHSAN 18 ,

```

2. Hapus data akechi

```

karin 18 , hoshino 18 , akechi 20 , yusuke 19 , michael 18 , jane 20 , john 19 , IHSAN 18 ,
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 6
Masukkan Posisi : 3

karin 18 , hoshino 18 , yusuke 19 , michael 18 , jane 20 , john 19 , IHSAN 18 ,

```

3. Tambahkan data berikut diantara John dan Jane : Futaba 18

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 3
Masukkan Posisi : 6
Masukkan Nama : futuba
Masukkan Usia : 18

NAMA      : Muhamad ihsan
NIM       : 2311102077
KELAS     : IF-11-B

Ln 3, Col 16 | 53 karakter | 100% | Windows | UTF-8

karin 18 , hoshino 18 , yusuke 19 , michael 18 , jane 20 , futuba 18 , john 19 , IHSAN 18 ,
```

4. Tambahkan data igor 20 di awal

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 2
Masukkan Nama : igor
Masukkan Usia : 20

NAMA      : Muhamad ihsan
NIM       : 2311102077
KELAS     : IF-11-B

Ln 3, Col 16 | 53 karakter | 100% | Windows | UTF-8

karin 18 , hoshino 18 , yusuke 19 , michael 18 , jane 20 , futuba 18 , john 19 , IHSAN 18 , igor 20 ,
```

5. Ubah data Michael menjadi reyn : 18

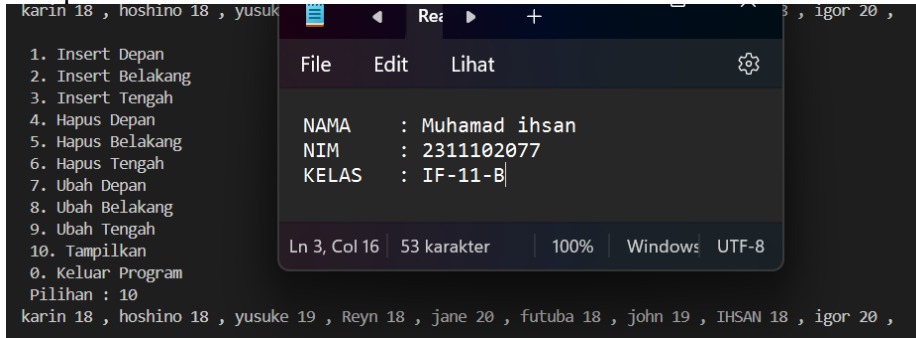
```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 9
Masukkan Posisi : 4
Masukkan Nama : Reyn
Masukkan Usia : 18

NAMA      : Muhamad ihsan
NIM       : 2311102077
KELAS     : IF-11-B

Ln 3, Col 16 | 53 karakter | 100% | Windows | UTF-8

karin 18 , hoshino 18 , yusuke 19 , Reyn 18 , jane 20 , futuba 18 , john 19 , IHSAN 18 , igor 20 ,
```

## 6. Tampilkan semua data



The screenshot shows a terminal window with a menu of operations for a linked list. The menu includes options for inserting, deleting, and updating data at different positions, as well as displaying all data and exiting the program. A modal window is open, displaying the following data:

NAMA	: Muhamad ihsan
NIM	: 23111102077
KELAS	: IF-11-B

The terminal also shows a list of names and ages: karin 18, hoshino 18, yusuke 19, Reyn 18, jane 20, futuba 18, john 19, IHSAN 18, igor 20.

### DESKRIPSI:

Program di atas Mengimplementasikan linked list dengan operasi dasar seperti penambahan elemen di depan dan belakang, penambahan elemen di tengah, serta penghapusan elemen di depan dan belakang. Setiap elemen dalam linked list memiliki dua bagian: nama dan usia. Fungsi-fungsi ini memungkinkan manipulasi data dengan efisien dan fleksibel

## 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

## SOURCE CODE:

```
#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 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;
}

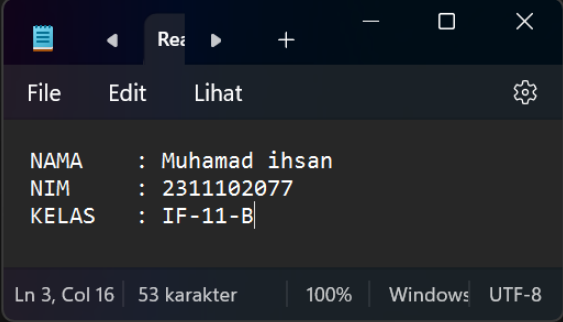
```

## OUTPUT:

1. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific

```
Masukkan posisi data produk: 2
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 : 7
```

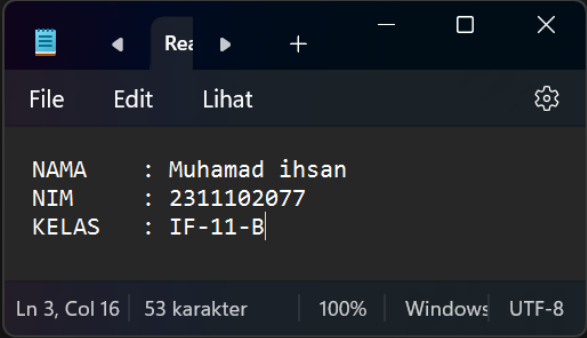
Nama Produk	Harga
originote	60000
somethinc	150000
azarine	65000
skintific	100000
wardah	50000
hansui	30000



2. Hapus produk wardah

```
Pilihan : 5
Masukkan posisi data produk: 4
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
```

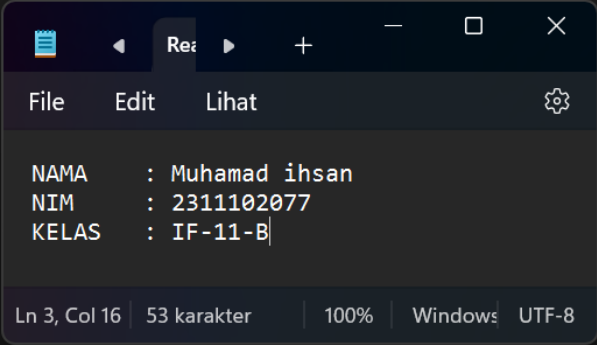
Nama Produk	Harga
originote	60000
somethinc	150000
azarine	65000
skintific	100000
hansui	30000



3. Update produk Hanasui menjadi Cleora dengan harga 55.000

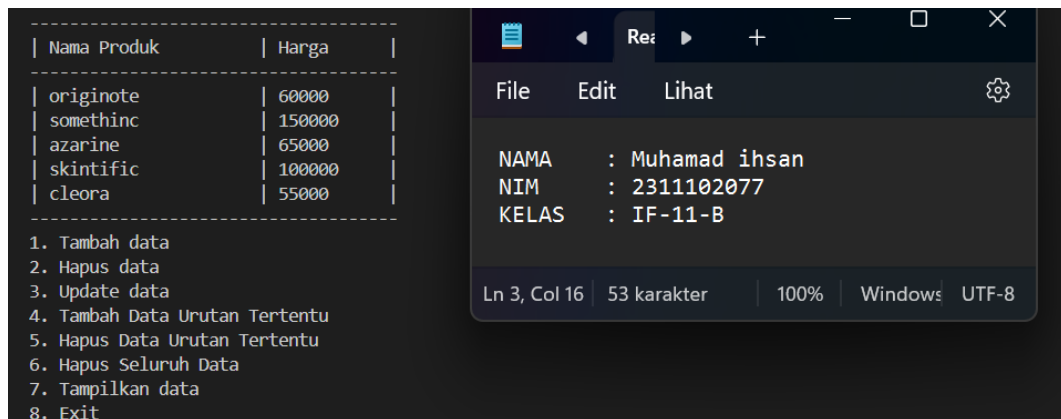
```
Masukkan posisi produk: 4
Masukkan nama baru produk: cleora
Masukkan harga baru produk: 55000
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 7
```

Nama Produk	Harga
originote	60000
somethinc	150000
azarine	65000
skintific	100000
cleora	55000





#### 4. Tampilkan menu



#### DESKRIPSI:

Program ini mengimplementasi dari Doubly Linked List untuk menyimpan data produk skincare. Pada program ini memiliki beberapa fungsi dasar seperti menambah data, menghapus data, mengupdate data, menambah data pada posisi tertentu, menghapus data pada posisi tertentu, menghapus seluruh data, dan menampilkan data. Program ini menggunakan Doubly Linked List untuk mengelola data produk skincare. Setiap node dalam Doubly Linked List memiliki dua pointer, yaitu pointer ke node sebelumnya (prev) dan pointer ke node berikutnya (next).

#### D. KESIMPULAN

Pada Singly-Linked List : Setiap simpul hanya memiliki satu pointer yang menunjuk ke simpul berikutnya. Analoginya mirip dengan langkah-langkah dalam perburuan. Setiap langkah berisi pesan dan pointer ke langkah berikutnya. Operasi dasar pada singly-linked list meliputi penambahan simpul, pencarian, dan penghapusan. Pada Doubly-Linked List: Setiap simpul memiliki dua pointer: satu menunjuk ke simpul sebelumnya dan satu menunjuk ke simpul berikutnya. Analoginya adalah seperti memiliki dua arah dalam perburuan. Anda dapat bergerak maju dan mundur. Operasi dasar pada doubly-linked list melibatkan penambahan, pencarian, dan penghapusan simpul.

#### E. REFERENSI

Reema Thareja. (2014). *Data structures using C. 02. OXFORD. New Delhi. ISBN: 9780198099307*  
<https://socs.binus.ac.id/2017/03/15/single-linked-list/>  
Sundell, Håkan, and Philippas Tsigas. "Lock-free dequeues and doubly linked lists." *Journal of Parallel and Distributed Computing* 68.7 (2008): 1008-1020.