LAPORAN PRAKTIKUM

MODUL III

Disusun oleh:
Brian Farrel Evandhika
NIM: 2311102037

Dosen Pengampu:
Muhammad Afrizal Amrustian, S. Kom., M. Kom.

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023

BAB I

TUJUAN PRAKTIKUM

A. Tujuan

1. Mahasiswa memahami perbedaan konsep Single dan Double Linked List
2. Mahasiswa mampu menerapkan Single dan Double Linked List ke dalam pemrograman

BAB II
DASAR TEORI

.

Linked List adalah struktur data yang terdiri dari sejumlah simpul atau node yang saling terhubung secara sekuensial melalui pointer. Setiap simpul memiliki dua bagian utama: bagian data yang menyimpan nilai dan bagian pointer yang menunjukkan ke simpul berikutnya dalam urutan. Dalam Single Linked List, setiap simpul memiliki satu pointer yang menunjuk ke simpul berikutnya, dengan simpul pertama disebut head dan simpul terakhir disebut tail. Circular Linked List adalah varian dari Single Linked List di mana pointer pada simpul terakhir merujuk kembali ke simpul pertama.

Double Linked List adalah varian dari Linked List yang memiliki dua pointer di setiap simpul: satu menunjuk ke simpul berikutnya dan satu lagi menunjuk ke simpul sebelumnya. Ini memungkinkan operasi penambahan dan penghapusan yang lebih fleksibel, serta traversal dari depan (head) dan belakang (tail) dengan mudah. Namun, penggunaan memori lebih besar dan operasi penambahan/penghapusan mungkin membutuhkan waktu lebih lama dibandingkan dengan Single Linked List.

BAB III

GUIDED

Guided 1

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    // komponen/member
    int data;
    string kata;
    Node *next;
};
Node *head;
Node *tail;
// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}
// Pengecekan
bool isEmpty()
{
    if (head == NULL)
        return true;
    else
        return false;
}
// Tambah Depan
void insertDepan(int nilai, string kata)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->kata = kata;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
```

```
        baru->next = head;
        head = baru;
    }
}
// Tambah Belakang
void insertBelakang(int nilai, string kata)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->kata = kata;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}
// Hitung Jumlah List
int hitungList()
{
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}
// Tambah Tengah
void insertTengah(int data, string kata, 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, *bantu;
        baru = new Node();
        baru->data = data;
        baru->kata = kata;
        // tranversing
        bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
// Hapus Depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}
// Hapus Belakang
void hapusBelakang()
{
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false)
    {
        if (head != tail)
        {
```

```
            hapus = tail;
            bantu = head;
            while (bantu->next != tail)
            {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}
// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *hapus, *bantu, *bantu2;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        int nomor = 1;
        bantu = head;
        while (nomor <= posisi)
        {
            if (nomor == posisi - 1)
            {
                bantu2 = bantu;
            }
            if (nomor == posisi)
            {
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
```

```
        }
        bantu2->next = bantu;
        delete hapus;
    }
}
// Ubah Depan
void ubahDepan(int data, string kata)
{
    if (isEmpty() == false)
    {
        head->data = data;
        head->kata = kata;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Tengah
void ubahTengah(int data, string kata, int posisi)
{
    Node *bantu;
    if (isEmpty() == false)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {
            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
            bantu->kata;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
```

```
    }
}
// Ubah Belakang
void ubahBelakang(int data, string kata)
{
    if (isEmpty() == false)
    {
        tail->data = data;
        tail->kata = kata;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Hapus List
void clearList()
{
    Node *bantu, *hapus;
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}
// Tampilkan List
void tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << "\t";
            cout << bantu->kata<<"\t";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
```

```
}
int main()
{
    init();
    insertDepan(3, "satu");
    tampil();
    insertBelakang(5, "dua");
    tampil();
    insertDepan(2, "tiga");
    tampil();
    insertDepan(1, "empat");
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7,"lima", 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1,"enam") ;
    tampil();
    ubahBelakang(8,"tujuh");
    tampil();
    ubahTengah(11,"delapan", 2);
    tampil();
    return 0;
}
```

Deskripsi Program

Program ini merupakan contoh implementasi single linked list non-circular dalam C++. Program ini mendemonstrasikan berbagai operasi seperti menambahkan, menghapus, mengubah, dan menampilkan data dalam list. Contohnya, program ini menambahkan 4 Node di depan list, menghapus Node pertama dan terakhir, menambahkan Node di tengah list, dan menampilkan data dan kata dari semua Node. Program ini dapat dimodifikasi untuk kebutuhan Anda, seperti menambahkan fungsionalitas mencari Node, mengurutkan Node, dan membalikkan list. Pastikan untuk memeriksa kondisi error dan menangani pointer dengan hati-hati untuk menghindari crash program.

Screenshot Program

```
3       satu
3       satu    5       dua
2       tiga    3       satu    5       dua
1       empat   2       tiga    3       satu    5       dua
2       tiga    3       satu    5       dua
2       tiga    3       satu
2       tiga    7       lima    3       satu
2       tiga    3       satu
1       enam    3       satu
1       enam    8       tujuh
1       enam    11      tujuh
PS C:\Users\MSI GAMING\Documents\Kuliah\SEMESTER 2\PRAKTIKUM STRUKTUR
DATA DAN ALGORITME\Praktikum Modul 3\2311102037_Brian-Farrel-Evandhika
_IF-11-A_Praktikum-Struktur-Data>
```

Guided 2

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

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

class DoublyLinkedList
{
public:
    Node *head;
    Node *tail;
    DoublyLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }

    void push(int data, string kata)
    {
        Node *newNode = new Node;
        newNode->data = data;
        newNode->kata = kata;
        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, string oldKata, string newKata)
    {
        Node *current = head;
        while (current != nullptr)
        {
            if (current->data == oldData && current->kata == oldKata)
            {
                current->data = newData;
                current->kata = newKata;
                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 << " ";
            cout << current->kata << endl;
            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;
            string kata;
            cout << "Enter data to add: ";
            cin >> data;
            cout << "Enter kata to add: ";
            cin >> kata;
            list.push(data, kata);
```

```
            break;
        }
        case 2:
        {
            list.pop();
            break;
        }
        case 3:
        {
            int oldData, newData;
            string oldKata, newKata;
            cout << "Enter old data: ";
            cin >> oldData;
            cout << "Enter new data: ";
            cin >> newData;
            cout << "Enter old kata: ";
            cin >> oldKata;
            cout << "Enter new kata: ";
            cin >> newKata;
            bool updated = list.update(oldData, newData, oldKata, newKata);
            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;
}
```

Deskripsi Program

Program ini merupakan implementasi doubly linked list dengan menu berinteraksi dalam bahasa C++. Program ini berfungsi untuk manajemen data mahasiswa yang terdiri dari nilai (data) berupa angka dan nama (kata) berupa string. Program ini memungkinkan Anda untuk:Program ini menggunakan class DoublyLinkedList yang menangani operasi linked list. Class ini menyediakan method untuk menambah data di depan (push), menghapus data pertama (pop), mengubah data yang sudah ada (update), menghapus semua data (deleteAll), dan menampilkan isi list (display).Program utama berjalan dalam sebuah loop, menampilkan menu dengan pilihan yang tersedia. Berdasarkan pilihan pengguna, program akan melakukan operasi linked list yang sesuai menggunakan method-method dari class DoublyLinkedList.

Screenshoot Program

```
PS C:\Users\MSI GAMING\Documents\Kuliah\SEMESTER 2\PRAKTIKUM STRUKTUR
DATA DAN ALGORITME\Praktikum Modul 3\2311102037_Brian-Farrel-Evandhika
_IF-11-A_Praktikum-Struktur-Data>  & 'c:\Users\MSI GAMING\.vscode\exte
nsions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDe
bugLauncher.exe' '--stdin=Microsoft-MIEngine-In-1bibtxue.acg' '--stdou
t=Microsoft-MIEngine-Out-r5l2eqwo.ay5' '--stderr=Microsoft-MIEngine-Er
ror-wr1ie2kq.gin' '--pid=Microsoft-MIEngine-Pid-wbeyywbm.pzv' '--dbgEx
e=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 1
Enter kata to add: Brian
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 2
Enter kata to add: Farrel
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
2 Farrel
1 Brian

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice:
```

Unguided 2

```cpp
#include <iostream>
#include <iomanip>

using namespace std;

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

Node* head;
Node* tail;

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

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

int hitungList() {
    Node* hitung = head;
    int jumlah = 0;

    while (hitung != NULL) {
        jumlah++;
        hitung = hitung->next;
    }

    return jumlah;
}

void insertDepan(string nama, int umur) {
    Node* baru = new Node;
    baru->nama = nama;
    baru->umur = umur;
    baru->next = NULL;

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

```
}

void insertBelakang(string nama, int umur) {
    Node* baru = new Node;
    baru->nama = nama;
    baru->umur = umur;
    baru->next = NULL;

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

void insertTengah(string nama, int umur, int posisi) {
    if (posisi < 2 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else {
        Node* baru = new Node();
        baru->nama = nama;
        baru->umur = umur;

        Node* bantu = head;
        int nomor = 1;

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

        baru->next = bantu->next;
        bantu->next = baru;
    }
}

void hapusDepan() {
    if (!isEmpty()) {
        Node* hapus = head;
        if (head->next != NULL) {
            head = head->next;
        } else {
            head = tail = NULL;
        }
        delete hapus;
    } else {
        cout << "List kosong!" << endl;
```

```
    }
}

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

void hapusTengah(int posisi) {
    if (!isEmpty()) {
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi di luar jangkauan" << endl;
        } else if (posisi == 1) {
            hapusDepan();
        } else {
            Node* hapus;
            Node* bantu = head;
            int nomor = 1;

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

            hapus = bantu->next;
            bantu->next = hapus->next;
            delete hapus;
        }
    } else {
        cout << "List kosong!" << endl;
    }
}

void ubahTengah(string nama, int umur, int posisi) {
```

```
    if (!isEmpty()) {
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi di luar jangkauan" << endl;
        } else {
            Node* bantu = head;
            int nomor = 1;

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

            bantu->nama = nama;
            bantu->umur = umur;
        }
    } else {
        cout << "List kosong!" << endl;
    }
}

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

void tampil() {
    Node* bantu = head;
    cout << left << setw(20) << "Nama" << setw(20) << "Umur" << endl;
    if (!isEmpty()) {
        while (bantu != NULL){
            cout << left << setw(20) << bantu->nama << setw(20) << bantu->umur
<< endl;
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

int main() {
    init();
```

```
// Tampilan awal
insertDepan("Karin", 18);
insertDepan("Hoshino", 18);
insertDepan("Akechi", 20);
insertDepan("Yusuke", 19);
insertDepan("Michael", 18);
insertDepan("Jane", 20);
insertDepan("John", 19);
tampil();

while (true) {
    cout << "1. Tambah Depan" << endl;
    cout << "2. Tambah Belakang" << endl;
    cout << "3. Tambah Tengah" << endl;
    cout << "4. Hapus Depan" << endl;
    cout << "5. Hapus Belakang" << endl;
    cout << "6. Hapus Tengah" << endl;
    cout << "7. Ubah Tengah" << endl;
    cout << "8. Hapus Semua" << endl;
    cout << "9. Tampilkan Data" << endl;
    cout << "10. Exit" << endl;
    int choice;
    cout << "Masukkan pilihan: ";
    cin >> choice;
    switch (choice) {
        case 1: {
            string nama;
            int umur;
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan umur: ";
            cin >> umur;
            insertDepan(nama, umur);
            cout << "Data berhasil ditambahkan" << endl;
            break;
        }
        case 2: {
            string nama;
            int umur;
            cout << "Masukkan nama: ";
            cin >> nama;
            cout << "Masukkan umur: ";
            cin >> umur;
            insertBelakang(nama, umur);
            cout << "Data berhasil ditambahkan" << endl;
            break;
        }
```

```
            case 3: {
                string nama;
                int umur, posisi;
                cout << "Masukkan nama: ";
                cin >> nama;
                cout << "Masukkan umur: ";
                cin >> umur;
                cout << "Masukkan posisi: ";
                cin >> posisi;
                insertTengah(nama, umur, posisi);
                cout << "Data berhasil ditambahkan" << endl;
                break;
            }
            case 4: {
                hapusDepan();
                cout << "Data berhasil dihapus" << endl;
                break;
            }
            case 5: {
                hapusBelakang();
                cout << "Data berhasil dihapus" << endl;
                break;
            }
            case 6: {
                int posisi;
                cout << "Masukkan posisi: ";
                cin >> posisi;
                hapusTengah(posisi);
                cout << "Data berhasil dihapus" << endl;
                break;
            }
            case 7: {
                string nama;
                int umur, posisi;
                cout << "Masukkan posisi: ";
                cin >> posisi;
                cout << "Masukkan nama: ";
                cin >> nama;
                cout << "Masukkan umur: ";
                cin >> umur;
                ubahTengah(nama, umur, posisi);
                cout << "Data berhasil diubah" << endl;
                break;
            }
            case 8: {
                clearList();
                break;
            }
```

```
            case 9: {
                tampil();
                break;
            }
            case 10: {
                return 0;
            }
            default: {
                cout << "Pilihan tidak sesuai!" << endl;
                break;
            }
        }
    }
    return 0;
}
```

Deskripsi Program

Program ini adalah sebuah aplikasi sederhana untuk mengelola data berupa nama dan umur dalam bentuk linked list di C++. Program ini menyediakan fungsi untuk menambahkan data di depan, di belakang, atau di tengah linked list, serta untuk menghapus data dari depan, belakang, atau tengahnya. Selain itu, program juga memungkinkan pengguna untuk mengubah data di posisi tengah, menghapus semua data, atau menampilkan seluruh data yang tersimpan. Pengguna dapat memilih opsi yang diinginkan dari menu yang disajikan dalam sebuah loop utama. Dengan melakukan pemilihan opsi yang sesuai, pengguna dapat melakukan manipulasi data dengan nyaman dan memanfaatkan fitur-fitur dasar dari struktur data linked list.

Screenshot Program

```
PS C:\Users\MSI GAMING\Documents\Kuliah\SEMESTER 2\PR
AKTIKUM STRUKTUR DATA DAN ALGORITME\Praktikum Modul 3
\2311102037_Brian-Farrel-Evandhika_IF-11-A_Praktikum-
Struktur-Data>  & 'c:\Users\MSI GAMING\.vscode\extens
ions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapter
s\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MI
Engine-In-gaavwr33.3lm' '--stdout=Microsoft-MIEngine-
Out-h5y4qs0t.omg' '--stderr=Microsoft-MIEngine-Error-
0hy22iz3.4ti' '--pid=Microsoft-MIEngine-Pid-coetiupd.
c2z' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--inter
preter=mi'
Nama                Umur
John                19
Jane                20
Michael             18
Yusuke              19
Akechi              20
Hoshino             18
Karin               18

1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Tengah
8. Hapus Semua
9. Tampilkan Data
10. Exit
Masukkan pilihan: 1
Masukkan nama: Brian
Masukkan umur: 18
Data berhasil ditambahkan
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Tengah
8. Hapus Semua
9. Tampilkan Data
10. Exit
Masukkan pilihan: 9
Nama                Umur
Brian               18
John                19
Jane                20
Michael             18
Yusuke              19
Akechi              20
Hoshino             18
Karin               18
```

Unguided 2

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

class Node {
public:
    string produk;
    int harga;
    Node *prev;
    Node *next;
};

class DoublyLinkedList {
public:
    Node *head;
    Node *tail;
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void push(const string& produk, int harga) {
        Node *newNode = new Node;
        newNode->produk = produk;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = head;
        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }
        head = newNode;
    }

    void insertAt(const string& produk, int harga, int posisi) {
        if (posisi <= 0) {
            cout << "Posisi harus lebih dari 0" << endl;
            return;
        }
        if (posisi == 1 || head == nullptr) {
            push(produk, harga);
            return;
        }

        Node *current = head;
        int count = 1;
        while (count < posisi - 1 && current->next != nullptr) {
```

```
            current = current->next;
            count++;
        }

        Node *newNode = new Node;
        newNode->produk = produk;
        newNode->harga = harga;
        newNode->prev = current;
        newNode->next = current->next;
        if (current->next != nullptr) {
            current->next->prev = newNode;
        }
        current->next = newNode;

        if (newNode->next == nullptr) {
            tail = 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(const string& oldProduk, const string& newProduk, int
newHarga) {
        Node *current = head;
        while (current != nullptr) {
            if (current->produk == oldProduk) {
                current->produk = newProduk;
                current->harga = newHarga;
                return true;
            }
            current = current->next;
        }
        return false;
    }

    void deleteAt(int posisi) {
```

```
        if (posisi <= 0 || head == nullptr) {
            cout << "Posisi tidak valid atau list kosong" << endl;
            return;
        }

        Node *current = head;
        int count = 1;
        while (count < posisi && current != nullptr) {
            current = current->next;
            count++;
        }

        if (current == nullptr) {
            cout << "Posisi melebihi ukuran list" << endl;
            return;
        }

        if (current == head) {
            pop();
            return;
        }

        if (current == tail) {
            tail = tail->prev;
            tail->next = nullptr;
            delete current;
            return;
        }

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

    void deleteAll() {
        Node *current = head;
        while (current != nullptr) {
            Node *temp = current;
            current = current->next;
            delete temp;
        }
        head = nullptr;
        tail = nullptr;
    }

    void display() {
        cout << "Nama Produk\tHarga" << endl;
        Node *current = head;
```

```
        while (current != nullptr) {
            cout << current->produk << "\t\t" << current->harga << endl;
            current = current->next;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList list;
    while (true) {
        cout << "Toko Skincare Purwokerto" << endl;
        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;

        int choice;
        cout << "Masukkan pilihan: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                string produk;
                int harga;
                cout << "Masukkan nama produk: ";
                cin >> produk;
                cout << "Masukkan harga: ";
                cin >> harga;
                list.push(produk, harga);
                cout << "Data berhasil ditambahkan" << endl;
                break;
            }
            case 2: {
                list.pop();
                cout << "Data terakhir berhasil dihapus" << endl;
                break;
            }
            case 3: {
                string oldProduk, newProduk;
                int newHarga;
                cout << "Masukkan nama produk yang akan diupdate: ";
                cin >> oldProduk;
                cout << "Masukkan nama produk baru: ";
```

```
                cin >> newProduk;
                cout << "Masukkan harga baru: ";
                cin >> newHarga;
                bool updated = list.update(oldProduk, newProduk, newHarga);
                if (updated) {
                    cout << "Data berhasil diupdate" << endl;
                } else {
                    cout << "Data tidak ditemukan" << endl;
                }
                break;
            }
            case 4: {
                string produk;
                int harga, posisi;
                cout << "Masukkan nama produk: ";
                cin >> produk;
                cout << "Masukkan harga: ";
                cin >> harga;
                cout << "Masukkan posisi: ";
                cin >> posisi;
                list.insertAt(produk, harga, posisi);
                cout << "Data berhasil ditambahkan di urutan " << posisi <<
endl;
                break;
            }
            case 5: {
                int posisi;
                cout << "Masukkan posisi data yang ingin dihapus: ";
                cin >> posisi;
                list.deleteAt(posisi);
                cout << "Data pada posisi " << posisi << " berhasil dihapus"
<< endl;
                break;
            }
            case 6: {
                list.deleteAll();
                cout << "Semua data berhasil dihapus" << endl;
                break;
            }
            case 7: {
                list.display();
                break;
            }
            case 8: {
                return 0;
            }
            default: {
                cout << "Pilihan tidak valid" << endl;
```

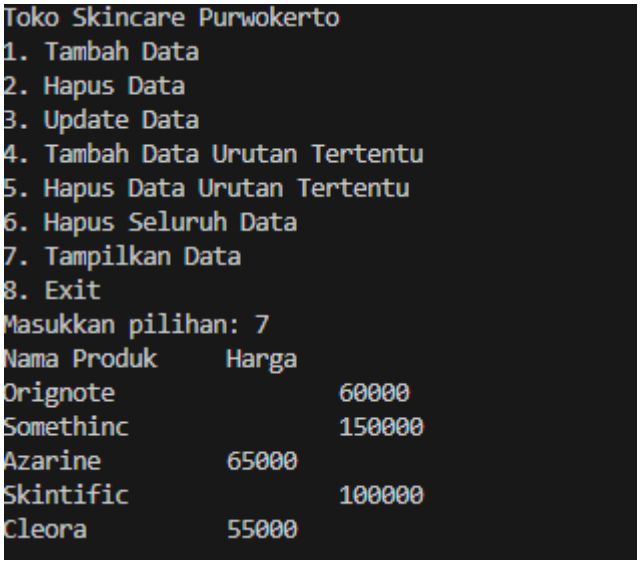
```
                break;
            }
        }
    }

    return 0;
}
```

Deskripsi Program

Program tersebut adalah implementasi dari Doubly Linked List dalam bahasa C++. Doubly Linked List adalah struktur data linear yang terdiri dari sejumlah node di mana setiap node memiliki dua pointer, yaitu pointer ke node sebelumnya (prev) dan pointer ke node berikutnya (next). Setiap node menyimpan data berupa nama produk dan harga. Program ini menyediakan beberapa operasi dasar untuk memanipulasi Doubly Linked List, seperti menambahkan data di depan (push), menghapus data dari depan (pop), mengupdate data, menambahkan data di tengah (insertAt), menghapus data di urutan tertentu (deleteAt), menghapus semua data, dan menampilkan semua data. Program memiliki loop utama yang memungkinkan pengguna untuk memilih operasi yang diinginkan dari menu yang disediakan. Setiap operasi diimplementasikan dalam metode yang sesuai dengan struktur Doubly Linked List dan ditampilkan dalam tampilan yang sesuai dengan format yang diminta.

Screenshot Program



BAB IV

KESIMPULAN

Linked List adalah struktur data yang terdiri dari simpul atau node yang terhubung secara sekuensial melalui pointer. Dalam Single Linked List, setiap simpul memiliki satu pointer yang menunjuk ke simpul berikutnya, dengan simpul pertama disebut head dan simpul terakhir disebut tail. Circular Linked List adalah varian dari Single Linked List di mana pointer pada simpul terakhir merujuk kembali ke simpul pertama, membentuk lingkaran. Double Linked List adalah varian dari Linked List di mana setiap simpul memiliki dua pointer: satu menunjuk ke simpul berikutnya dan satu lagi menunjuk ke simpul sebelumnya. Ini memungkinkan traversal dari depan (head) dan belakang (tail), serta operasi penambahan dan penghapusan yang lebih fleksibel. Namun, penggunaan memori lebih besar dan operasi tertentu mungkin membutuhkan waktu lebih lama dibandingkan dengan Single Linked List. Dengan berbagai variasi dan karakteristiknya, Linked List menjadi struktur data yang fleksibel dan dapat disesuaikan untuk berbagai kebutuhan pemrograman.