

LAPORAN PRAKTIKUM
MODUL 3
SINGLE AND DOUBLE LINKED LIST



Disusun oleh :

Diva Octaviani

NIM : 2311102006

Dosen Pengampu:

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

PROGRAM STUDI TEKNIK INFORMATIKAFAKULTAS
INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024

BAB I

TUJUAN PRAKTIKUM

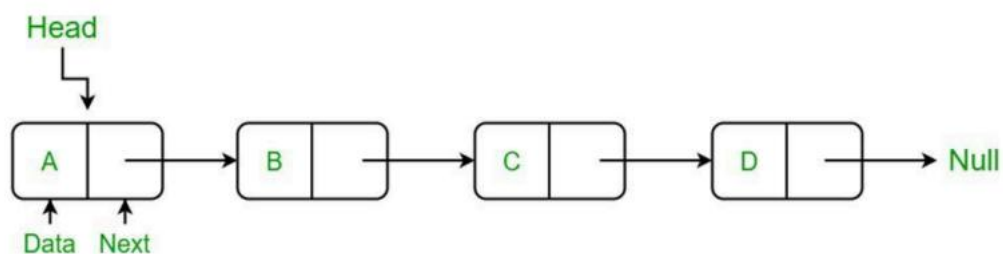
1. Mahasiswa memahami perbedaan konsep Single dan Double Linked List
2. Mahasiswa mampu menerapkan Single dan Double Linked List ke dalam pemrograman

BAB II

DASAR TEORI

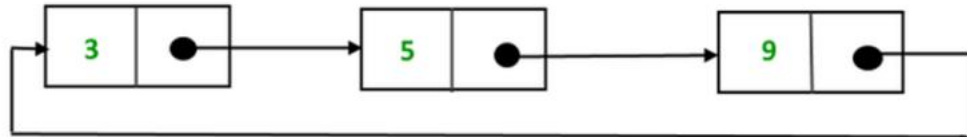
a) Single Linked List

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung-menysambung, 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. Elemen pada awal suatu list disebut head dan elemen terakhir dari suatu list disebut tail.



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.

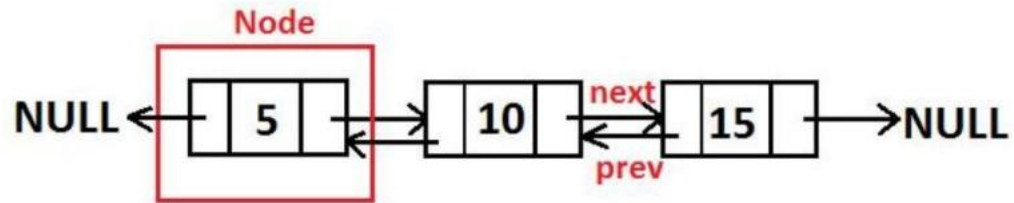


b) Double Linked List

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. 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.

Keuntungan dari Double Linked List adalah memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul dimana saja dengan efisien, sehingga sangat berguna dalam implementasi beberapa algoritma yang membutuhkan operasi tersebut. Selain itu, Double Linked List juga memungkinkan kita untuk melakukan traversal pada list baik dari depan (head) maupun dari belakang (tail) dengan mudah. Namun, kekurangan dari Double Linked List adalah penggunaan memori yang lebih besar dibandingkan dengan Single Linked List, karena setiap simpul membutuhkan satu pointer tambahan. Selain itu, Double Linked List juga membutuhkan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan jika dibandingkan dengan Single Linked List.

Representasi sebuah double linked list dapat dilihat pada gambar berikut ini :



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 III

GUIDED

1. Guided 1

Source Code

```
#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 = 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;
            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;  
  
}
```

Screenshoot Program

```
3      satu
3      satu5  dua
2      tiga3  satu5  dua
1      empat2 tiga3  satu5  dua
2      tiga3  satu5  dua
2      tiga3  satu
2      tiga7  lima3  satu
2      tiga3  satu
1      enam3  satu
1      enam8  tujuh
1      enam11 delapan
```

Deskripsi Program

Program ini adalah implementasi struktur data Single Linked List Non-Circular dalam bahasa C++. Program menyediakan berbagai fungsi untuk operasi pada linked list seperti inisialisasi, pengecekan kosong, penambahan node di depan, belakang, dan tengah, penghapusan node di depan, belakang, dan tengah, pengubahan data node di depan, tengah, dan belakang, penghitungan jumlah node, dan penampilan seluruh data dalam linked list. Fungsi-fungsi tersebut memungkinkan pengelolaan data dalam bentuk linked list yang fleksibel dan dinamis. Dalam akhir program, terdapat contoh penggunaan fungsi-fungsi tersebut untuk mendemonstrasikan cara kerja program.

2. Guided 2

Source Code

```
#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 newKata)
{
    Node *current = head;
    while (current != nullptr)
    {
        if (current->data == oldData)
        {
            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 newKata;
    cout << "Enter old data: ";
    cin >> oldData;
    cout << "Enter new data: ";
```

```

        cin >> newData;

        cout << "Enter new kata: ";

        cin >> newKata;

        bool updated = list.update(oldData,
                                    newData, 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;

}
```

Screenshoot Program



```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
```

Deskripsi Program

Program ini mengimplementasikan struktur data Double Linked List dalam bahasa C++. Kelas Node digunakan untuk mewakili setiap node dalam double linked list yang menyimpan data integer dan string, serta pointer ke node sebelum dan sesudahnya. Kelas DoublyLinkedList memiliki pointer head dan tail menunjuk ke node pertama dan terakhir. Kelas ini menyediakan operasi seperti push untuk menambah node baru di depan, pop untuk menghapus node pertama, update untuk mengubah data dan kata pada node, deleteAll untuk menghapus semua node, dan display untuk menampilkan data. Pada fungsi main, program menampilkan menu pilihan kepada pengguna untuk melakukan operasi pada double linked list, seperti menambah data, menghapus data, mengupdate data, menghapus semua data, dan menampilkan data, serta opsi untuk keluar dari program.

LATIHAN KELAS - UNGUIDED

1. 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

- 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

Source Code

```
#include <iostream>
#include <string>
using namespace std;
// Deklarasi Struct Node
struct Node
{
    string nama;
    int usia;
    Node *next;
};
Node *head = nullptr;
// Fungsi untuk menambahkan data di depan linked list
void insertDepan(string nama, int usia)
{
    Node *newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    newNode->next = head;
    head = newNode;
}
// Fungsi untuk menambahkan data di belakang linked list
void insertBelakang(string nama, int usia)
{
    Node *newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    newNode->next = nullptr;
    if (head == nullptr)
    {
        head = newNode;
        return;
    }
    Node *temp = head;
```

```

        while (temp->next != nullptr)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    // Fungsi untuk menambahkan data di tengah linked list
    void insertTengah(string nama, int usia, string nama_sebelumnya)
    {
        Node *newNode = new Node;
        newNode->nama = nama;
        newNode->usia = usia;
        Node *temp = head;
        while (temp != nullptr)
        {
            if (temp->nama == nama_sebelumnya)
            {
                newNode->next = temp->next;
                temp->next = newNode;
                return;
            }
            temp = temp->next;
        }
    }
    // Fungsi untuk menampilkan seluruh data
    void tampilkanData()
    {
        Node *temp = head;
        while (temp != nullptr)
        {
            cout << temp->nama << " " << temp->usia << endl;
            temp = temp->next;
        }
    }
}

```

```

int main()
{
    string namaAnda;
    int usiaAnda;
    cout << "Masukkan nama Anda : ";
    getline(cin, namaAnda);
    cout << "Masukkan usia Anda : ";
    cin >> usiaAnda;
    cin.ignore();
    insertDepan(namaAnda, usiaAnda);
    insertBelakang("John", 19);
    insertBelakang("Jane", 20);
    insertBelakang("Michael", 18);
    insertBelakang("Yusuke", 19);
    insertBelakang("Akechi", 20);
    insertBelakang("Hoshino", 18);
    insertBelakang("Karin", 18);
    cout << "Data Mahasiswa :" << endl;
    tampilkanData();
    return 0;
}

```

Screenshoot program

Tampilan program setelah menginputkan nama dan usia :

```

Masukkan nama Anda : Diva Octaviani
Masukkan usia Anda : 18
Data Mahasiswa :
Diva Octaviani 18
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18

```

Menghapus data “Akechi” :

```
Diva Octaviani 18
John 19
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
```

Menambahkan data “Futaba 18” diantara John dan Jane :

```
Diva Octaviani 18
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
```

Menambahkan data “Igor 20” di awal :

```
Igor 20
Diva Octaviani 18
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
```

Mengubah data “Michael” menjadi “Reyn 18” :

```
Igor 20
Diva Octaviani 18
John 19
Futaba 18
Jane 20
Reyn 18
Yusuke 19
Hoshino 18
Karin 18
```

Tampilan seluruh data :

```
Data Mahasiswa :  
Igor 20  
Diva Octaviani 18  
John 19  
Futaba 18  
Jane 20  
Reyn 18  
Yusuke 19  
Hoshino 18  
Karin 18
```

Deskripsi program

Dalam contoh kode di atas, kita telah melihat implementasi Linked List dalam bahasa pemrograman C++. Kita dapat memasukkan node baru di awal, akhir, atau setelah simpul tertentu dalam Linked List. Kita juga dapat menghapus node berdasarkan nama dan mengubah data (nama dan usia) dari node tertentu. Selain itu, kita dapat menampilkan semua data dalam Linked List. Linked List adalah struktur data yang fleksibel dan efisien untuk menyimpan dan mengelola kumpulan data yang dinamis

2. Unguided 2

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 <string>
using namespace std;
struct Node
{
    string namaProduk;
    int harga;
    Node *prev;
    Node *next;
};
class DoublyLinkedList
{
private:
    Node *head;
    Node *tail;

public:
    DoublyLinkedList() : head(nullptr), tail(nullptr) {}
    void insertBelakang(string namaProduk, int harga)
    {
        Node *newNode = new Node;
```

```

        newNode->namaProduk = namaProduk;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = nullptr;
        if (head == nullptr)
        {
            head = tail = newNode;
        }
        else
        {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
    }
    void display()
    {
        Node *current = head;
        while (current != nullptr)
        {
            cout << current->namaProduk << "\t" << current->
            >harga << endl;
            current = current->next;
        }
    }
};

int main()
{
    DoublyLinkedList list;
    // Menambahkan data
    list.insertBelakang("Originote", 60000);
    list.insertBelakang("Somethinc", 150000);

```

```

    list.insertBelakang("Skintific", 100000);
    list.insertBelakang("Wardah", 50000);
    list.insertBelakang("Hanasui", 30000);
    cout << "Nama Produk\tHarga" << endl;
    list.display();
    return 0;
}

```

Screenshoot program

Nama Produk	Harga
Originote	60000
Somethinc	150000
Skintific	100000
Wardah	50000
Hanasui	30000

Menambahkan produk Azarine dengan harga 65.000 diantara Somethinc dan Skintific :

Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Wardah	50000
Hanasui	30000

Menghapus produk Wardah :

Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Hanasui	30000

Meng-update produk Hanasui menjadi Cleora dengan harga 55.000 :

Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Cleora	55000

Menampilkan menu :

```
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
Pilih menu:
```

```
Menu 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
Masukkan pilihan anda: 7
Nama Produk Harga
Originote 60000
somethinc 150000
azarine 65000
skintific 100000
cleora 55000
```

Deskripsi program

Program ini mengimplementasikan struktur data Double Linked List untuk mengelola data produk pada sebuah toko skincare. Program memiliki kelas DoubleLinkedList yang berisi fungsi-fungsi untuk menambah produk baru dengan nama dan harga secara terurut alfabetis, menghapus produk berdasarkan nama, memperbarui harga produk, menampilkan semua produk beserta nama dan harganya, serta menambahkan produk baru di antara dua produk tertentu. Di dalam fungsi main, program menampilkan menu pilihan kepada pengguna untuk melakukan berbagai operasi pada linked list seperti menambah data produk, menghapus data produk, mengupdate data produk, menambahkan data produk pada urutan tertentu, dan menampilkan semua data produk yang ada dalam linked list. Program akan terus berjalan hingga pengguna memilih untuk keluar.

BAB IV

KESIMPULAN

1. Single Linked List dan Double Linked List adalah dua struktur data yang populer untuk menyimpan data secara linear.
2. Single Linked List :
 - Memiliki pointer next untuk menunjuk ke node berikutnya.
 - Efisien dalam penggunaan memori.
 - Operasi penambahan dan penghapusan pada awal atau akhir list lebih cepat.
 - Operasi penambahan dan penghapusan pada tengah list lebih lambat.
3. Double Linked List :
 - Memiliki pointer next dan prev untuk menunjuk ke node berikutnya dan sebelumnya.
 - Membutuhkan lebih banyak memori dibandingkan Single Linked List.
 - Operasi penambahan dan penghapusan pada awal, akhir, dan tengah list lebih cepat.
4. Single Linked List lebih cocok untuk aplikasi yang membutuhkan memori kecil dan operasi penambahan dan penghapusan pada awal atau akhir list lebih sering dilakukan.
5. Double Linked List lebih cocok untuk aplikasi yang membutuhkan operasi penambahan dan penghapusan pada tengah list lebih sering dilakukan.

DAFTAR PUSTAKA

Asisten Praktikum, “*Modul 3 Single and Double Linked List*”, Learning Management System, 2024.