LAPORAN PRAKTIKUM

MODUL III
SINGLE AND DOUBLE LINKED LIST

Disusun oleh:
Raihan Ramadhan
NIM: 2311102040

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

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2024

BAB I

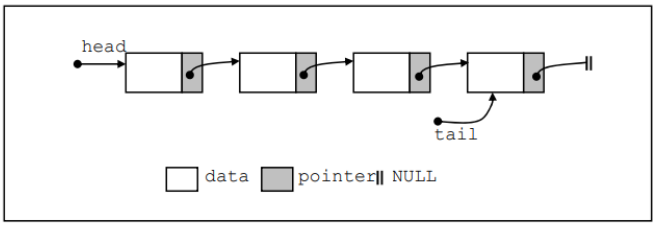
TUJUAN PRAKTIKUM

Pembelajaran ini bertujuan untuk memberikan pemahaman mendalam kepada mahasiswa mengenai perbedaan konsep antara Single dan Double Linked List dalam struktur data. Tujuan ini meliputi kemampuan mahasiswa untuk mengidentifikasi karakteristik khusus dari masing-masing jenis linked list dan memahami dampaknya dalam pengembangan program. Selain itu, mereka diharapkan dapat mengaplikasikan kedua jenis linked list tersebut dalam konteks pemrograman. Hal ini akan mempersiapkan mereka dengan keterampilan yang diperlukan untuk menangani manipulasi data yang kompleks.

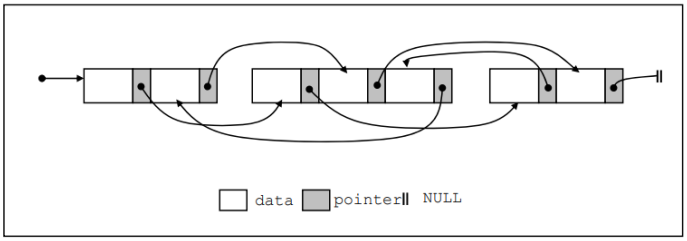
BAB II

DASAR TEORI

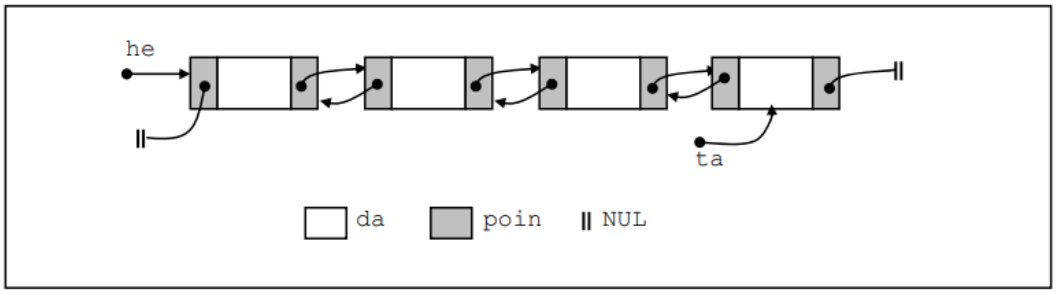
Single linked list atau biasa disebut linked list terdiri dari elemen-elemen individu, dimana masing-masing dihubungkan dengan pointer tunggal. Masing-masing elemen terdiri dari dua bagian, yaitu sebuah data dan sebuah pointer yang disebut dengan pointer next. Dengan menggunakan struktur two-member seperti ini, linked list dibentuk dengan cara menunjuk pointer next suatu elemen ke elemen yang mengikutinya seperti gambar 2.1. 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.



Untuk mengakses elemen dalam linked list, dimulai dari head dan menggunakan pointer next dari elemen selanjutnya untuk berpindah dari elemen ke elemen berikutnya sampai elemen yang diminta dicapai. Dengan single linke list, list dapat dilintasi hanya satu arah dari head ke tail karena masing-masing elemen tidak terdapat link dengan elemen sebelumnya. Sehingga, apabila kita mulai dari head dan berpindah ke beberapa elemen dan berharap dapat mengakses elemen sebelumnya, kita harus mulai dari head.



Elemen-elemen dihubungkan dengan dua pointer dalam satu elemen. Struktur ini menyebabkan list melintas baik ke depan maupun ke belakang. Masing-masing elemen pada double linked list terdiri dari tiga bagian, disamping data dan pointer next, masing-masing elemen dilengkapi dengan pointer prev yang menunjuk ke elemen sebelumnya. Double linked list dibentuk dengan menyusun sejumlah elemen sehingga pointer next menunjuk ke elemen yang mengikutinya dan pointer prev menunjuk ke elemen yang mendahuluinya. Untuk menunjukkan head dari double linked list, maka pointer prev dari elemen pertama menunjuk NULL. Untuk menunjukkan tail dari double linked list tersebut, maka pointer next dari elemen terakhir menunjuk NULL.



Untuk melintas kembali melalui double linked list, kita gunakan pointer prev dari elemen yang berurutan pada arah tail ke head. Double linked list mempunyai fleksibilitas yang lebih tinggi daripada single linked list dalam perpindahan pada list. Bentuk ini sangat berguna ketika akan meletakkan suatu elemen pada list dan dapat memilih dengan lebih bijaksana bagaimana memindahkannya.

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;
        }
    }
    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;
    }
}
```

```cpp
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;
}
```

Screenshot Output

```
PS D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Pertemuan 4> cd "
++ Guided1_Single_Lingked_List.cpp -o Guided1_Single_Lingked
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 D:\Matkul\SEMESTER 2\PRAKTIKUM STRUKDAT\Pertemuan 4>
```

Deskripsi Program

Program tersebut merupakan single linked list non-circular memiliki fungsi-fungsi untuk melakukan operasi dasar pada linked list seperti penambahan, penghapusan, dan pengubahan elemen. struktur data Node yang memiliki dua bagian, yaitu data dan pointer next yang menunjuk ke elemen berikutnya. Fungsi-fungsi seperti insertDepan(), insertBelakang(), dan insertTengah() digunakan untuk menambahkan elemen baru ke dalam linked list. Fungsi-fungsi seperti hapusDepan(), hapusBelakang(), dan hapusTengah() digunakan untuk menghapus elemen dari linked list.

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;
    }
```

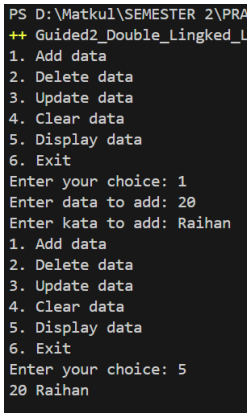
```
void push(int data, string kata)
{
    Node *newNode = new Node;
    newNode->data = data;
    newNode->kata = kata;
    newNode->prev = nullptr;
    newNode->next = head;
    if (head != nullptr)
    {
        head->prev = newNode;
    }
    else
    {
        tail = newNode;
    }
    head = newNode;
}

void pop()
{
    if (head == nullptr)
    {
        return;
    }
    Node *temp = head;
    head = head->next;
    if (head != nullptr)
    {
        head->prev = nullptr;
    }
    else
    {
        tail = nullptr;
    }
    delete temp;
}
```

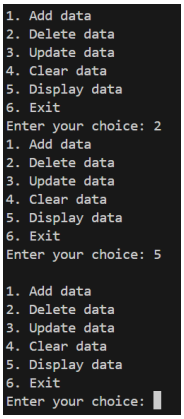
```
    bool update(int oldData, int newData, string newKata)
    {
        Node *current = head;
        while (current != nullptr)
        {
            if (current->data == oldData)
            {
                current->data = newData;
                current->kata = newKata;
                return 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;
    }
};
```

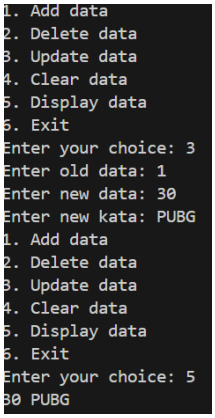
```cpp
int main()
{
    DoublyLinkedList list;
    while (true)
    {
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Exit"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

Screenshot Output







Deskripsi Program

Program tersebut adalah implementasi sederhana dari Double Linked List dalam C++, yang memungkinkan pengguna untuk menambah, menghapus, mengubah, dan menampilkan data dalam linked list. Fungsi push() digunakan untuk menambahkan data ke depan linked list, fungsi pop() menghapus data dari depan, fungsi update() mengubah data yang sudah ada, fungsi deleteAll() menghapus seluruh data, dan fungsi display() menampilkan semua data.Program memberikan menu pilihan yang berulang kepada pengguna untuk melakukan operasi-operasi tersebut hingga pengguna memilih untuk keluar.

BAB IV

UNGUIDED

1. Unguided 1

Source Code

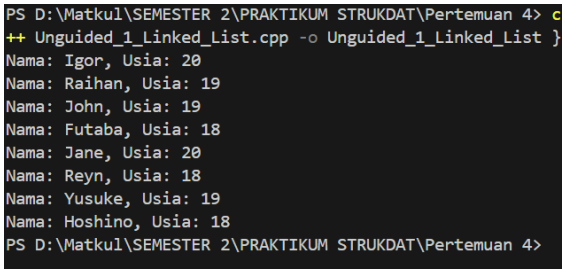
```
#include <iostream>
using namespace std;

// Raihan Ramadhan
// 231110204
// IF-11-A

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

class LinkedList
{
private:
    Node *head, *tail;

public:
    LinkedList()
    {
        head = NULL;
        tail = NULL;
    }

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

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

    void insertTengah(string nama, int usia, string namaSetelah)
    {
        Node *temp = new Node;
        temp->nama = nama;
        temp->usia = usia;
        Node *current = head;
        while (current != NULL && current->nama != namaSetelah)
        {
            current = current->next;
        }
        if (current != NULL)
        {
            temp->next = current->next;
            current->next = temp;
        }
        else
        {
            cout << "Node dengan nama " << namaSetelah << " tidak
ditemukan." << endl;
        }
    }
```

```
    void hapusData(string nama)
    {
        Node *temp = head, *prev = NULL;
        while (temp != NULL && temp->nama != nama)
        {
            prev = temp;
            temp = temp->next;
        }
        if (temp == NULL)
            return;
        if (prev == NULL)
        {
            head = temp->next;
        }
        else
        {
            prev->next = temp->next;
        }
        delete temp;
    }

    void ubahData(string namaLama, string namaBaru, int usiaBaru)
    {
        Node *current = head;
        while (current != NULL && current->nama != namaLama)
        {
            current = current->next;
        }
        if (current != NULL)
        {
            current->nama = namaBaru;
            current->usia = usiaBaru;
        }
    }

    void tampilkanData()
    {
        Node *current = head;
        while (current != NULL)
        {
            cout << "Nama: " << current->nama << ", Usia: " << current-
>usia << endl;
            current = current->next;
        }
    }
};
```

```cpp
int main()
{
    LinkedList list;
    // Masukkan data sesuai instruksi
    list.insertDepan("Raihan", 19);
    list.insertBelakang("John", 19);
    list.insertBelakang("Jane", 20);
    list.insertBelakang("Michael", 18);
    list.insertBelakang("Yusuke", 19);
    list.insertBelakang("Akechi", 20);
    list.insertBelakang("Hoshino", 18);

    // Operasi sesuai instruksi
    list.hapusData("Akechi");
    list.insertTengah("Futaba", 18, "John");
    list.insertDepan("Igor", 20);
    list.ubahData("Michael", "Reyn", 18);
    list.tampilkanData();

    return 0;
}
```

Screenshot Program



Deskripsi Program

Program tersebut merupakan implementasi dari Single Linked List Non-Circular dalam. Program tersebut memiliki kelas `LinkedList` yang mendefinisikan operasi-operasi dasar seperti `insertDepan`, `insertBelakang`, `insertTengah`, `hapusData`, `ubahData`, dan `tampilkanData`. Setiap node dalam linked list menyimpan data mahasiswa berupa nama dan usia. Data mahasiswa dimasukkan sesuai dengan instruksi menggunakan metode `insertDepan` dan `insertBelakang`. Dilakukan operasi-operasi seperti menghapus, menambahkan, mengubah data, dan menampilkan seluruh data sesuai instruksi. Hasil akhirnya adalah seluruh data mahasiswa yang sudah dimodifikasi sesuai dengan instruksi.

2. Unguided 2

Source Code

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

// Raihan Ramadhan
// 2311102040
// IF-11-A

struct Node
{
    string namaProduk;
    int harga;
    Node *next;
    Node *prev;
};

class DoubleLinkedList
{
private:
    Node *head, *tail;

public:
    DoubleLinkedList()
    {
        head = NULL;
        tail = NULL;
    }
```

```
// Fungsi untuk menambahkan data di depan
void tambahDataDepan(string namaProduk, int harga)
{
    Node *temp = new Node;
    temp->namaProduk = namaProduk;
    temp->harga = harga;
    temp->next = head;
    temp->prev = NULL;
    if (head != NULL)
    {
        head->prev = temp;
    }
    head = temp;
    if (tail == NULL)
    {
        tail = head;
    }
}

// Fungsi untuk menambahkan data di belakang
void tambahDataBelakang(string namaProduk, int harga)
{
    Node *temp = new Node;
    temp->namaProduk = namaProduk;
    temp->harga = harga;
    temp->next = NULL;
    temp->prev = tail;
    if (tail != NULL)
    {
        tail->next = temp;
    }
    tail = temp;
    if (head == NULL)
    {
        head = tail;
    }
}
```

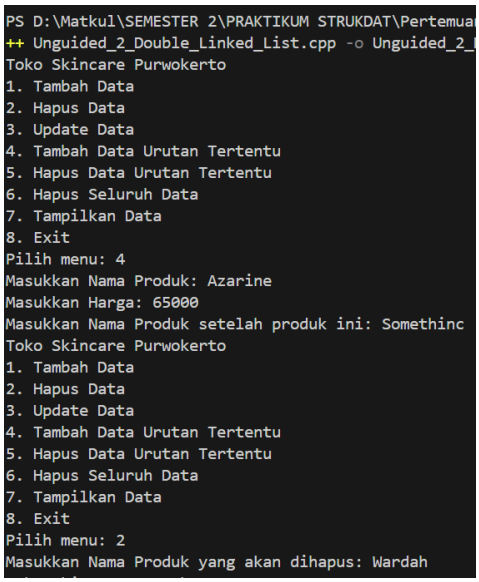
```
    // Fungsi untuk menambahkan data di urutan tertentu
    void tambahDataTertentu(string namaProduk, int harga, string
namaProdukSetelah)
    {
        Node *current = head;
        while (current != NULL && current->namaProduk !=
namaProdukSetelah)
        {
            current = current->next;
        }
        if (current != NULL && current->next != NULL)
        {
            Node *temp = new Node;
            temp->namaProduk = namaProduk;
            temp->harga = harga;
            temp->next = current->next;
            temp->prev = current;
            current->next->prev = temp;
            current->next = temp;
        }
    }

    // Fungsi untuk menghapus data
    void hapusData(string namaProduk)
    {
        Node *temp = head;
        while (temp != NULL && temp->namaProduk != namaProduk)
        {
            temp = temp->next;
        }
        if (temp == NULL)
            return;
        if (temp->prev != NULL)
        {
            temp->prev->next = temp->next;
        }
        else
        {
            head = temp->next;
        }
        if (temp->next != NULL)
        {
            temp->next->prev = temp->prev;
        }
        else
        {
            tail = temp->prev;
        }
        delete temp;
    }
```

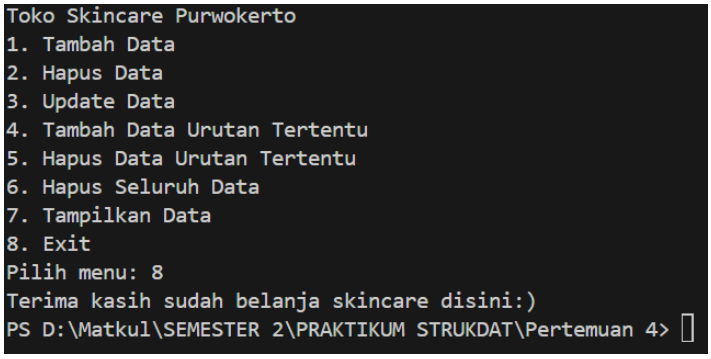
```
    // Fungsi untuk mengupdate data
    void updateData(string namaProdukLama, string namaProdukBaru, int
hargaBaru)
    {
        Node *current = head;
        while (current != NULL && current->namaProduk != namaProdukLama)
        {
            current = current->next;
        }
        if (current != NULL)
        {
            current->namaProduk = namaProdukBaru;
            current->harga = hargaBaru;
        }
    }

    // Fungsi untuk menampilkan semua data
    void tampilkanData()
    {
        Node *current = head;
        cout << "Toko Skincare Purwokerto\n";
        cout << left << setw(20) << "Nama Produk"
             << "Harga\n";
        while (current != NULL)
        {
            cout << left << setw(20) << current->namaProduk << current-
>harga << endl;
            current = current->next;
        }
    }

    // Fungsi untuk menghapus seluruh data
    void hapusSemuaData()
    {
        while (head != NULL)
        {
            Node *temp = head;
            head = head->next;
            delete temp;
        }
        tail = NULL;
    }
};
```

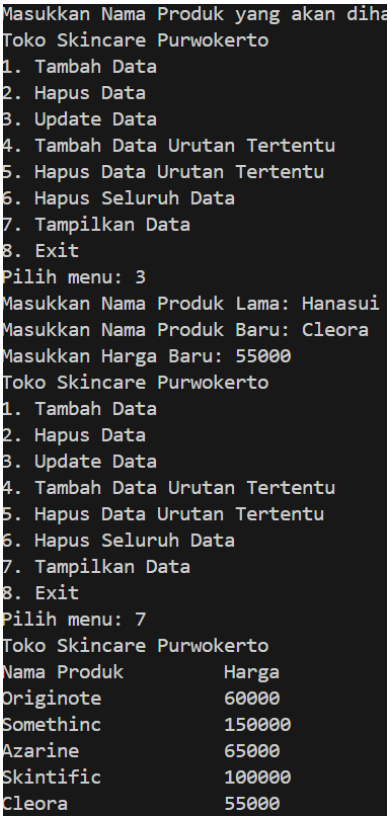
```
// Fungsi untuk menampilkan menu
void tampilkanMenu()
{
    cout << "Toko Skincare Purwokerto\n";
    cout << "1. Tambah Data\n";
    cout << "2. Hapus Data\n";
    cout << "3. Update Data\n";
    cout << "4. Tambah Data Urutan Tertentu\n";
    cout << "5. Hapus Data Urutan Tertentu\n";
    cout << "6. Hapus Seluruh Data\n";
    cout << "7. Tampilkan Data\n";
    cout << "8. Exit\n";
}

int main()
{
    DoubleLinkedList list;
    int pilihan, harga;
    string namaProduk, namaProdukSetelah, namaProdukLama,
namaProdukBaru;

    // Tambahkan data awal
    list.tambahDataBelakang("Originote", 60000);
    list.tambahDataBelakang("Somethinc", 150000);
    list.tambahDataBelakang("Skintific", 100000);
    list.tambahDataBelakang("Wardah", 50000);
    list.tambahDataBelakang("Hanasui", 30000);

    do
    {
        tampilkanMenu();
        cout << "Pilih menu: ";
        cin >> pilihan;
        switch (pilihan)
        {
        case 1:
            cout << "Masukkan Nama Produk: ";
            cin >> namaProduk;
            cout << "Masukkan Harga: ";
            cin >> harga;
            list.tambahDataBelakang(namaProduk, harga);
            break;
        case 2:
            cout << "Masukkan Nama Produk yang akan dihapus: ";
            cin >> namaProduk;
            list.hapusData(namaProduk);
            break;
```

```
        case 3:
            cout << "Masukkan Nama Produk Lama: ";
            cin >> namaProdukLama;
            cout << "Masukkan Nama Produk Baru: ";
            cin >> namaProdukBaru;
            cout << "Masukkan Harga Baru: ";
            cin >> harga;
            list.updateData(namaProdukLama, namaProdukBaru, harga);
            break;
        case 4:
            cout << "Masukkan Nama Produk: ";
            cin >> namaProduk;
            cout << "Masukkan Harga: ";
            cin >> harga;
            cout << "Masukkan Nama Produk setelah produk ini: ";
            cin >> namaProdukSetelah;
            list.tambahDataTertentu(namaProduk, harga,
namaProdukSetelah);
            break;
        case 5:
            cout << "Masukkan Nama Produk yang akan dihapus: ";
            cin >> namaProduk;
            list.hapusData(namaProduk);
            break;
        case 6:
            list.hapusSemuaData();
            cout << "Seluruh data telah dihapus.\n";
            break;
        case 7:
            list.tampilkanData();
            break;
        case 8:
            cout << "Terima kasih sudah belanja skincare disini:)\n";
            break;
        default:
            cout << "Pilihan tidak valid. Silakan coba lagi.\n";
        }
    } while (pilihan != 8);

    return 0;
}
```

Screenshot Output







Deskripsi Program

Program tersebut merupakan aplikasi toko skincare sederhana yang menggunakan konsep Double Linked List untuk menyimpan data produk skincare beserta harganya. Program ini memungkinkan pengguna untuk menambahkan data produk, menghapus data produk, mengubah data produk, menambahkan data di urutan tertentu, menghapus data di urutan tertentu, menghapus seluruh data, dan menampilkan seluruh data produk yang tersedia. Setiap produk skincare memiliki atribut nama dan harga. Pengguna dapat memilih menu yang diinginkan dari menu yang disediakan. Program akan terus berjalan hingga pengguna memilih untuk keluar.

BAB V

KESIMPULAN

Node adalah tempat penyimpanan data pada Linked List dimana terdiri dari dua bagian/field yakni field data dan pointer. Pointer (link) adalah field untuk menyimpan alamat tertentu. Link adalah tempat penyimpanan alamat simpulnya (pointer). Linked list adalah sebuah struktur untuk menyimpan data yang bersifat dinamik Beberapa operasi dapat diterapkan pada linked list seperti sisip(insert),hapus(delete) . Null adalah suatu kondisi khusus dimana pointer itu belum di set dengan sebuah address tertentu, artinya pointer tidak mrnunjuk ke alamat manapun. Double linked list dibentuk dengan menyusun sejumlah elemen sehingga pointer next menunjuk ke elemen yang mengikutinya dan pointer back menunjuk ke elemen yang mendahuluinya. Double Linked List, di sisi lain, memiliki dua pointer untuk setiap simpul: prev (menunjuk ke simpul sebelumnya) dan next (menunjuk ke simpul berikutnya). Keuntungan double linked list adalah kemampuan untuk mengakses elemen baik dari depan maupun belakang, sehingga operasi seperti pencarian atau penghapusan dapat dilakukan dengan lebih efisien. Namun, kelemahannya adalah kompleksitas lebih tinggi karena perlu mengelola dua pointer per simpul.

DAFTAR PUSTAKA

Asisten Praktikum, “MODUL 3 SINGLE AND DOUBLE LINKED LIST", Learning Management System, 31 Maret 2024.

Yuliana,” BAB II Senarai Berantai (Linked List) ”, https://yuliana.lecturer.pens.ac.id/Struktur%20Data%20C/Teori%20SD%20-%20pdf/Data%20Structure%20-%20Bab%202.pdf ,1 April 2024