LAPORAN PRAKTIKUM

MODUL 3

SINGLE AND DOUBLE LINKED LIST

Disusun oleh:
Nabila Shasya Sabrina
NIM: 2311102039

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

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

1. Single Linked List

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Setiap elemen dalam linked list dihubungkan ke elemen lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau node atau verteks. Pointer adalah alamat elemen. Setiap simpul pada dasarnya dibagi atas dua bagian pertama disebut bagian isi atau informasi atau data yang berisi nilai yang disimpan oleh simpul. Bagian kedua disebut bagian pointer yang berisi alamat dari node berikutnya atau sebelumnya. Dengan menggunakan struktur seperti ini, linked list dibentuk dengan cara menunjuk pointer next suatu elemen ke elemen yang mengikutinya. Pointer next pada elemen terakhir merupakan NULL, yang menunjukkan akhir dari suatu list. 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.

2. 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.

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;
```

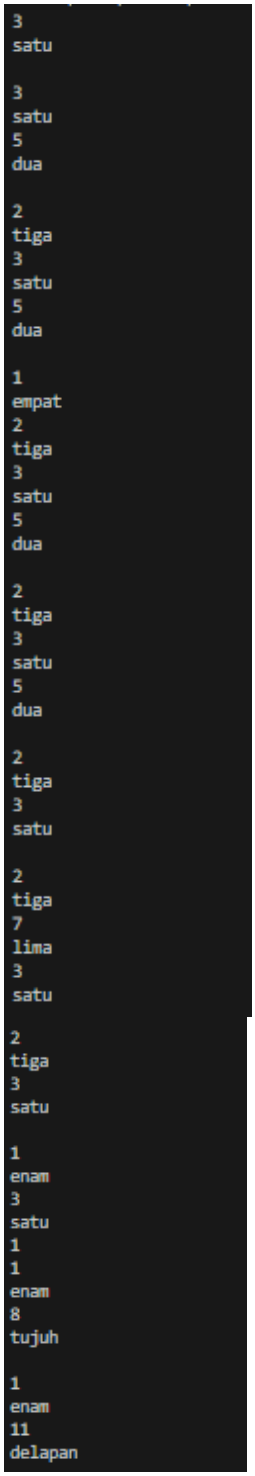
```
    }
}
// Hapus Depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    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 << endl;
            cout << bantu->kata << endl;
            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:



Deskripsi program:

Program yang diberikan adalah implementasi dari struktur data Linked List berbasis Node tunggal (single linked list non-circular). Program ini menyediakan fungsi-fungsi dasar untuk manipulasi Linked List seperti penambahan data di depan, di belakang, di tengah,

penghapusan data di depan, di belakang, di tengah, serta pembaruan data di depan, di belakang, dan di tengah. Setiap Node memiliki dua komponen yaitu data bertipe integer dan kata bertipe string. Program ini juga menyediakan fungsi untuk menampilkan jumlah elemen dalam list serta menampilkan isi dari list. Pada fungsi `main`, beberapa operasi dasar seperti penambahan, penghapusan, dan pembaruan elemen dilakukan untuk demonstrasi fungsionalitas program.

2. **Guided 2**

Source Code:

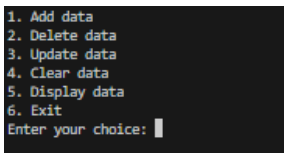
```
#include <iostream>
using namespace std;
class Node
{
public:
    int data;
    string kata;
    Node *prev;
    Node *next;
};
class DoublyLinkedList
{
public:
    Node *head;
    Node *tail;
    DoublyLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }
    void push(int data, 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;
```

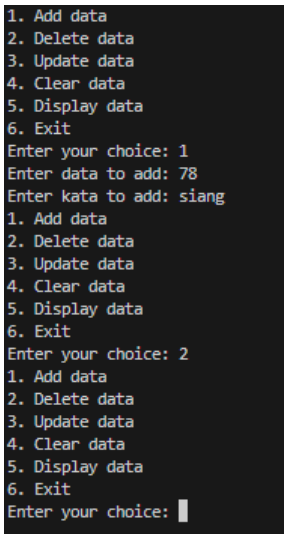
```
        }
        Node *temp = head;
        head = head->next;
        if (head != nullptr)
        {
            head->prev = nullptr;
        }
        else
        {
            tail = nullptr;
        }
        delete temp;
    }
    bool update(int oldData, int newData, string oldKata, string
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 << " " << current->kata;
            current = current->next;
        }
        cout << endl;
    }
};
int main()
{
    DoublyLinkedList list;
    while (true)
    {
```

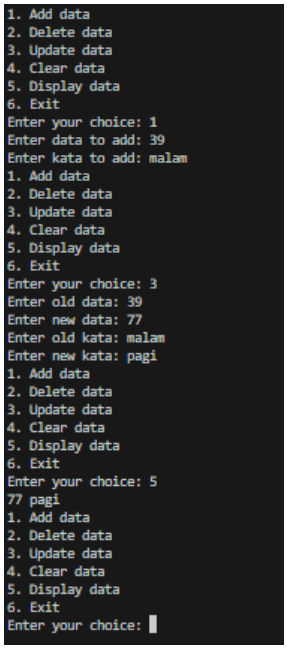
```
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Exit" << endl;
        int choice;
        cout << "Enter your choice: ";
        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;
        }
```

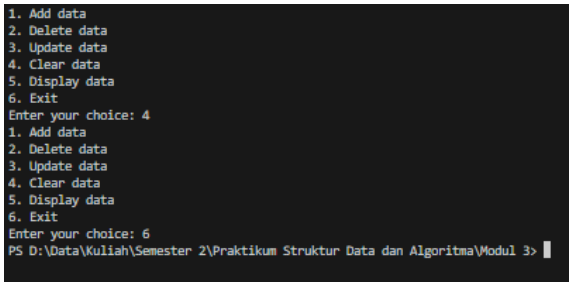
```
        case 6:
        {
            return 0;
        }
        default:
        {
            cout << "Invalid choice" << endl;
            break;
        }
        }
    }
    return 0;
}
```

Screenshoot program:









Deskripsi Program:

Program tersebut adalah implementasi dari Doubly Linked List. Program ini menyediakan kelas `Node` yang merepresentasikan setiap elemen dalam list, dengan atribut data (bertipe integer) dan kata (bertipe string), serta pointer ke node sebelumnya `prev` dan node berikutnya `next`.

Kelas `DoublyLinkedList` merupakan kelas yang menyediakan operasi-operasi dasar pada Doubly Linked List, seperti penambahan elemen di awal `push`, penghapusan elemen di awal `pop`, pembaruan elemen `update`, penghapusan seluruh elemen dalam list `deleteAll`, dan menampilkan isi dari list `display`.

Pada fungsi `main`, program memberikan menu kepada pengguna untuk memilih operasi yang ingin dilakukan, seperti menambahkan data, menghapus data, memperbarui data, membersihkan data, menampilkan data, dan keluar dari program. Program tersebut akan terus berjalan hingga pengguna memilih untuk keluar (pilihan 6).

UNGUIDED

1. Unguided 1

Soal mengenai Single Linked List

Buatlah program menu Single Linked List Non-Circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputandari 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 <iomanip>
using namespace std;

// Deklarasi Struct Node
struct Node {
    string namaNya;
    int umurNya;
    Node* next;
};

Node* head;
Node* tail;
```

```
// Inisialisasi Node
void inisialisasiNya() {
    head = NULL;
    tail = NULL;
}

// Pengecekan apakah linked list kosong
bool cekNya() {
    return head == NULL;
}

// Tambah Node di depan
void depanNya(string name, int age) {
    Node* baru = new Node;
    baru->namaNya = name;
    baru->umurNya = age;
    baru->next = NULL;

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

// Tambah Node di belakang
void belakangNya(string name, int age) {
    Node* baru = new Node;
    baru->namaNya = name;
    baru->umurNya = age;
    baru->next = NULL;

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

// Hitung Jumlah Node
int jumlahlistNya() {
    int jumlah = 0;
    Node* hitung = head;

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

    return jumlah;
}

// Tambah Node di tengah
void tengahNya(string name, int age, int posisi) {
    if (posisi < 1 || posisi > jumlahlistNya()) {
```

```
        cout << "Tidak terjangkau!" << endl;
    } else if (posisi == 1) {
        cout << "Bukan di tengah." << endl;
    } else {
        Node* baru = new Node();
        baru->namaNya = name;
        baru->umurNya = age;

        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 hapusNya() {
    if (!cekNya()) {
        Node* hapus = head;

        if (head->next != NULL) {
            head = head->next;
        } else {
            head = tail = NULL;
        }

        delete hapus;
    } else {
        cout << "Kosong!" << endl;
    }
}

// Hapus Node di belakang
void hapusbelakangNya() {
    if (!cekNya()) {
        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 << "Kosong!" << endl;
    }
}

// Hapus Node di tengah
void hapustengahNya(int posisi) {
    if (posisi < 1 || posisi > jumlahlistNya()) {
        cout << "Tidak terjangkau!" << endl;
    } else if (posisi == 1) {
        cout << "Bukan yang tengah." << endl;
    } else {
        Node* hapus;
        Node* bantu = head;
        Node* bantu2 = nullptr;
        int nomor = 1;

        while (nomor <= posisi) {
            if (nomor == posisi - 1) {
                bantu2 = bantu;
            }

            if (nomor == posisi) {
                hapus = bantu;
            }

            bantu = bantu->next;
            nomor++;
        }

        bantu2->next = bantu;
        delete hapus;
    }
}

// Ubah data di depan
void ubahdepanNya(string name, int age) {
    if (!cekNya()) {
        head->namaNya = name;
        head->umurNya = age;
    } else {
        cout << "Tidak ada yang berubah!" << endl;
    }
}

// Ubah data di tengah
void ubahtengahNya(string name, int age, int posisi) {
    if (!cekNya()) {
        if (posisi < 1 || posisi > jumlahlistNya()) {
            cout << "Tidak Terjangkau!" << endl;
        } else if (posisi == 1) {
            cout << "Bukan yang Tengah." << endl;
        } else {
            Node* bantu = head;
            int nomor = 1;

            while (nomor < posisi) {
```

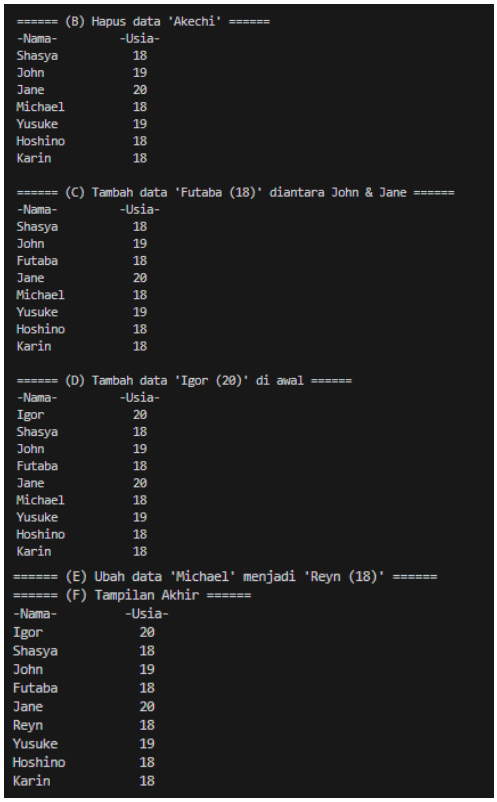
```
                bantu = bantu->next;
                nomor++;
            }

            bantu->namaNya = name;
            bantu->umurNya = age;
        }
    } else {
        cout << "Kosong!" << endl;
    }
}

// Ubah data di belakang
void ubahbelakangNya(string name, int age) {
    if (!cekNya()) {
        tail->namaNya = name;
        tail->umurNya = age;
    } else {
        cout << "Kosong" << endl;
    }
}

// Hapus semua Node
void hapuslistNya() {
    Node* bantu = head;
    Node* hapus;

    while (bantu != NULL) {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }

    head = tail = NULL;
    cout << "Menghapus semua!" << endl;
}

// Tampilkan semua Node
void tampillistNya() {
    Node* bantu = head;

    cout << left << setw(15) << "-Nama-" << right << setw(4) << "-
Usia-" << endl;

    if (!cekNya()) {
        while (bantu != NULL) {
             cout << left << setw(15) << bantu->namaNya << right <<
setw(4) << bantu->umurNya << endl;
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "Kosong!" << endl;
    }
}

int main() {
```

```
    inisialisasiNya(); // Inisialisasi Linked List

     cout  <<  "\n(A.)======  SELAMAT  DATANG  ======"  <<  endl;  //
Menampilkan nama dan umur awal & menjawab poin a

    string nama, namaAnda;
    int usia, usiaAnda;
    cout << "Masukkan nama Anda: ";
    cin >> namaAnda;
    cout << "Masukkan usia Anda: ";
    cin >> usiaAnda;
    depanNya(namaAnda, usiaAnda);

    cout << "Masukkan nama mahasiswa lainnya:" << endl;
    cout << "Nama: ";
    cin >> nama;
    cout << "Usia: ";
    cin >> usia;
    belakangNya(nama, usia);

    cout << "Nama: ";
    cin >> nama;
    cout << "Usia: ";
    cin >> usia;
    belakangNya(nama, usia);

    cout << "Nama: ";
    cin >> nama;
    cout << "Usia: ";
    cin >> usia;
    belakangNya(nama, usia);

    cout << "Nama: ";
    cin >> nama;
    cout << "Usia: ";
    cin >> usia;
    belakangNya(nama, usia);

    cout << "Nama: ";
    cin >> nama;
    cout << "Usia: ";
    cin >> usia;
    belakangNya(nama, usia);

    cout << "Nama: ";
    cin >> nama;
    cout << "Usia: ";
    cin >> usia;
    belakangNya(nama, usia);

    cout << "Nama: ";
    cin >> nama;
    cout << "Usia: ";
    cin >> usia;
    belakangNya(nama, usia);

    tampillistNya();
```

```
    // Menjawab poin b
    cout << "====== (B) Hapus data 'Akechi' ======" << endl;
    hapustengahNya(6);
    tampillistNya();

    // Menjawab poin c
    cout << "====== (C) Tambah data 'Futaba (18)' diantara John & 
Jane ======" << endl;
    tengahNya("Futaba", 18, 3);
    tampillistNya();

    // Menjawab poin d
    cout << "====== (D) Tambah data 'Igor (20)' di awal ======" << 
endl;
    depanNya("Igor", 20);
    tampillistNya();

    // Menjawab poin e & f
    cout << "====== (E) Ubah data 'Michael' menjadi 'Reyn (18)' 
======" << endl;
    cout << "====== (F) Tampilan Akhir ======" << endl;
    ubahtengahNya("Reyn", 18, 6);
    tampillistNya();

    return 0;
}
```

Screenshoot program:

```
(A.)====== SELAMAT DATANG ======
Masukkan nama Anda: Shasya
Masukkan usia Anda: 18
Masukkan nama mahasiswa lainnya:
Nama: John
Usia: 19
Nama: Jane
Usia: 20
Nama: Michael
Usia: 18
Nama: Yusuke
Usia: 19
Nama: Akechi
Usia: 20
Nama: Hoshino
Usia: 18
Nama: Karin
Usia: 18
-Nama-        -Usia-
Shasya          18
John            19
Jane            20
Michael         18
Yusuke          19
Akechi          20
Hoshino         18
Karin           18
```



Deskripsi program:

Program tersebut merupakan implementasi dari linked list. Program tersebut mencakup beberapa fungsi dasar yang terkait dengan linked list, seperti inisialisasi linked list, penambahan node di depan dan di belakang, penghapusan node di depan, di belakang, dan di tengah, penghitungan jumlah node, serta pengubahan data di depan, di tengah, dan di belakang.

Selain itu, program tersebut juga menyediakan fungsi untuk menampilkan semua node yang ada dalam linked list. Program tersebut juga dilengkapi dengan fungsi main yang digunakan untuk menguji setiap operasi yang telah diimplementasikan dalam linked list, seperti penambahan, penghapusan, dan pengubahan data. Setiap operasi tersebut dijalankan sesuai dengan input yang diberikan oleh pengguna.

2. Unguided 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	50.000

Case:

1. Tambahkan produk Azarine dengan harga 65000 diantaraSomethinc 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 di bawah 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>
#include <string>
using namespace std;

class Node
{ // Deklarasi Class Node untuk Double Linked List
public:
    string Nama_Produk;
    int harga;
    Node *prev;
    Node *next;
};

class DoublyLinkedList
{ // Deklarasi Class DoublyLinkedList untuk Double Linked List
public:
    Node *head;
    Node *tail;
    DoublyLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }

    void tambahproduk140(string Nama_Produk, int harga)
    { // Menambahkan produk ke dalam linked list di bagian atas
        Node *newNode = new Node;
        newNode->Nama_Produk = Nama_Produk;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = head;
        if (head != nullptr)
        {
            head->prev = newNode;
        }
        else
        {
            tail = newNode;
        }
        head = newNode;
    }

    void hapusproduk140()
    { // Menghapus produk teratas dari linked list
        if (head == nullptr)
        {
            return;
        }
        Node *temp = head;
```

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

    bool ubahproduk140(string Nama_Produk_Lama, string
     Nama_Produk_Baru, int Harga_Baru)
    { // Mengubah data produk berdasarkan nama produk
        Node *current = head;
        while (current != nullptr)
        {
            if (current->Nama_Produk == Nama_Produk_Lama)
            {
                current->Nama_Produk = Nama_Produk_Baru;
                current->harga = Harga_Baru;
                return true;
            }
            current = current->next;
        }
        return false; // Mengembalikan false jika data produk
         tidak ditemukan
    }

    void sisipposisi140(string Nama_Produk, int harga, int
     posisi)
    { // Menambahkan data produk pada posisi tertentu
        if (posisi < 1)
        {
            cout << "Posisi tidak ada" << endl;
            return;
        }
        Node *newNode = new Node;
        newNode->Nama_Produk = Nama_Produk;
        newNode->harga = harga;
        if (posisi == 1)
        { // Jika posisi adalah 1 maka tambahkan data produk di
         depan linked list
            newNode->next = head;
            newNode->prev = nullptr;
            if (head != nullptr)
            {
                head->prev = newNode;
            }
            else
            {
                tail = newNode;
            }
            head = newNode;
            return;
        }
```

```
        Node *current = head;
         for (int i = 1; i < posisi - 1 && current != nullptr;
          ++i)
        { // Looping sampai posisi sebelum posisi yang diinginkan
          (Posisi - 1)
            current = current->next;
        }
        if (current == nullptr)
        {
            cout << "Posisi tidak ada" << endl;
            return;
        }
        newNode->next = current->next;
        newNode->prev = current;
        if (current->next != nullptr)
        {
            current->next->prev = newNode; // Pointer prev node
         setelah current menunjuk ke newNode jika node setelah
         current tidak nullptr
        }
        else
        {
            tail = newNode;
        }
        current->next = newNode;
    }

    void hapusposisi140(int posisi)
    { // Menghapus data produk pada posisi tertentu
        if (posisi < 1 || head == nullptr)
        {
            cout << "Posisi tidak ada atau list kosong" << endl;
            return;
        }
        Node *current = head;
        if (posisi == 1)
        {
            head = head->next;
            if (head != nullptr)
            {
                head->prev = nullptr;
            }
            else
            {
                tail = nullptr;
            }
            delete current;
            return;
        }
        for (int i = 1; current != nullptr && i < posisi; ++i)
        { // Looping sampai posisi yang diinginkan
            current = current->next;
        }
        if (current == nullptr)
        {
            cout << "Posisi tidak ada" << endl;
            return;
```

```
        }
        if (current->next != nullptr)
        {
            current->next->prev = current->prev;
        }
        else
        {
            tail = current->prev;
        }
        current->prev->next = current->next;
        delete current;
    }

    void hapussemua140()
    { // Menghapus semua data produk
        Node *current = head;
        while (current != nullptr)
        {
            Node *temp = current;
            current = current->next;
            delete temp;
        }
        head = nullptr;
        tail = nullptr;
    }

    void tampilan140()
    { // Menampilkan data produk
        Node *current = head;
        cout << "\nBerikut daftar Produk dan harga yang tersedia
          saat ini:" << endl;
        cout << left << setw(20) << "Nama Produk" << "Harga" <<
          endl;
        while (current != nullptr)
        {
            cout << left << setw(20) << current->Nama_Produk <<
         current->harga << endl;
            current = current->next;
        }
        cout << endl;
    }
};

int main()
{
     DoublyLinkedList list; // Deklarasi objek list dari class
          DoublyLinkedList

    list.tambahproduk140("Hanasui", 30000);
    list.tambahproduk140("Wardah", 50000);
    list.tambahproduk140("Skintific", 100000);
    list.tambahproduk140("Somethinc", 150000);
    list.tambahproduk140("Originote", 60000);

      cout  <<  "\n==========Selamat  datang  di  Toko  Skincare
          Purwokerto==========" << endl;
    list.tampilan140();
```

```
    while (true)
    { // Looping menu utama
        cout << "\nMenu 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 pilihan;
        cout << "Pilih Menu: ";
        cin >> pilihan;
        switch (pilihan)
        { // Switch case untuk memilih menu
        case 1:
        {
            string Nama_Produk;
            int harga;
            cout << "Masukkan nama produk: ";
            cin >> Nama_Produk;
            cout << "Masukkan harga: ";
            cin >> harga;
                 list.tambahproduk140(Nama_Produk,  harga);  //
         Memanggil fungsi tambah_produk
             cout << "Produk berhasil ditambahkan teratas" <<
         endl;
           break;
        }
        case 2:
        {
                  list.hapusproduk140();  //  Memanggil  fungsi
         hapus_produk
           cout << "Produk teratas berhasil dihapus" << endl;
           break;
        }
        case 3:
        {
            string Nama_Produk_Lama, Nama_Produk_Baru;
            int Harga_Baru;
            cout << "Input nama produk lama: ";
            cin >> Nama_Produk_Lama;
            cout << "Input nama produk baru: ";
            cin >> Nama_Produk_Baru;
            cout << "Input harga baru: ";
            cin >> Harga_Baru;
           bool updated = list.ubahproduk140(Nama_Produk_Lama,
         Nama_Produk_Baru,  Harga_Baru);  //  Memanggil  fungsi
         ubah_produk
           if (!updated)
           {
               cout << "Data produk tidak ditemukan" << endl;
           }
           else
           {
```

```
                cout << "Data produk berhasil diupdate" << endl;
            }
            break;
        }
        case 4:
        {
            string Nama_Produk;
            int harga, position;
            cout << "Input nama produk: ";
            cin >> Nama_Produk;
            cout << "Input harga: ";
            cin >> harga;
            cout << "Input posisi: ";
            cin >> position;
             list.sisipposisi140(Nama_Produk, harga, position);
         // Memanggil fungsi sisipkan_posisi_tertentu
             cout << "Produk berhasil ditambahkan pada posisi "
         << position << endl;
            break;
        }
        case 5:
        {
            int position;
            cout << "Input posisi yang ingin dihapus: ";
            cin >> position;
             list.hapusposisi140(position); // Memanggil fungsi
         hapus_posisi_tertentu

            break;
        }
        case 6:
        {
           list.hapussemua140(); // Memanggil fungsi hapus_semua
            break;
        }
        case 7:
        {
            list.tampilan140(); // Memanggil fungsi display
            break;
        }
        case 8:
        {
            return 0;
        }
        default:
        {
            cout << "Input Invalid" << endl;
            break;
        }
        }
    }
    return 0;
}
```

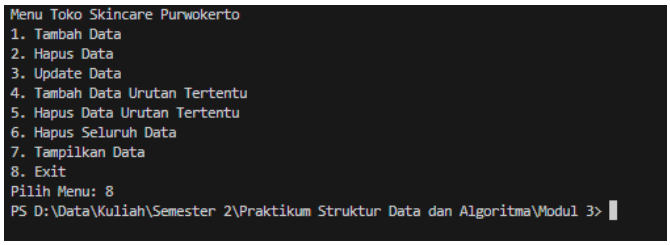
Screenshot Program:

```
==========Selamat datang di Toko Skincare Purwokerto==========

Berikut daftar Produk dan harga yang tersedia saat ini:
Nama Produk         Harga
Originote           60000
Somethinc           150000
Skintific           100000
Wardah              50000
Hanasui             30000

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: 4
Input nama produk: Azarine
Input harga: 65000
Input posisi: 3
Produk berhasil ditambahkan pada posisi 3
```

```
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: 5
Input posisi yang ingin dihapus: 5
```

```
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: 3
Input nama produk lama: Hanasui
Input nama produk baru: Cleora
Input harga baru: 55000
Data produk berhasil diupdate
```

```
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: 7

Berikut daftar Produk dan harga yang tersedia saat ini:
Nama Produk         Harga
Originote           60000
Somethinc           150000
Azarine             65000
Skintific           100000
Cleora              55000
```



Deskripsi program:

Program tersebut merupakan implementasi dari double linked list. Program tersebut mencakup dua kelas utama: `Node` dan `DoublyLinkedList`. Kelas `Node` digunakan untuk merepresentasikan setiap elemen dalam linked list, sedangkan kelas `DoublyLinkedList` digunakan untuk mengelola linked list secara keseluruhan.

Kelas `Node` memiliki atribut yang menyimpan informasi tentang nama produk dan harga, serta pointer yang menunjukkan ke node sebelumnya dan node selanjutnya dalam linked list.

Kelas `DoublyLinkedList` memiliki atribut `head` dan `tail` yang menunjukkan ke node pertama dan terakhir dalam linked list. Kelas ini menyediakan berbagai fungsi untuk memanipulasi linked list, seperti menambahkan produk ke atas, menghapus produk teratas, mengubah produk berdasarkan nama, menyisipkan produk pada posisi tertentu, menghapus produk pada posisi tertentu, menghapus semua produk, dan menampilkan semua produk yang tersedia.

Fungsi `main()` digunakan untuk menguji fungsi-fungsi yang telah diimplementasikan dalam kelas `DoublyLinkedList`. Program ini menyediakan menu interaktif yang memungkinkan pengguna untuk melakukan operasi pada linked list, seperti menambah, menghapus, mengubah, dan menampilkan produk. Setiap operasi tersebut dijalankan sesuai dengan input yang diberikan oleh pengguna.

DAFTAR PUSTAKA

Asisten Praktikum. 2024. "*MODUL III SINGLE AND DOUBLE LINKED LIST*". Learning Management System.