LAPORAN PRAKTIKUM

MODUL III
SINGLE AND DOUBLE LINKED LIST

Disusun oleh:
Nadhif Atha Zaki
NIM: 2311102007

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

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

BAB I
TUJUAN PRAKTIKUM

1. Mahasiswa dapat memahami konsep single linked list dan double linked list

2. Mahasiswa dapat mengetahui single linked list dan double linked list dan cara penulisannya

3. Mahasiswa dapat mengimplementasikan single linked list dan double linked list pada kode program yang dibuat

BAB II

DASAR TEORI

Salah satu bentuk struktur data yang berisi kumpulan data yang tersusun secara sekuensial, saling bersambungan, dinamis dan terbatas adalah senarai berkait (linked list). Suatu senarai berkait (linked list) adalah suatu simpul (node) yang dikaitkan dengan simpul yang lain dalam suatu urutan tertentu. Suatu simpul dapat berbentuk suatu struktur atau class. Simpul harus mempunyai satu atau lebih elemen struktur atau class yang berisi data. Secara teori, linked list adalah sejumlah node yang dihubungkan secara linier dengan bantuan pointer. Senarai berkait lebih efisien di dalam melaksanakan penyisipan-penyisipan dan penghapusan-penghapusan. Senarai berkait juga menggunakan alokasi penyimpanan secara dinamis, yang merupakan penyimpanan yang dialokasikan pada runtime. Karena di dalam banyak aplikasi, ukuran dari data itu tidak diketahui pada saat kompile, hal ini bisa merupakan suatu atribut yang baik juga. Setiap node akan berbentuk struct dan memiliki satu buah field bertipe struct yang sama, yang berfungsi sebagai pointer. Dalam menghubungkan setiap node, kita dapat menggunakan cara first-create-first-access ataupun first-create-lastaccess. Yang berbeda dengan deklarasi struct sebelumnya adalah satu field bernama next, yang bertipe struct tnode. Hal ini sekilas dapat membingungkan. Namun, satu hal yang jelas, variabel next ini akan menghubungkan kita dengan node di sebelah kita, yang juga bertipe struct tnode. Hal inilah yang menyebabkan next harus bertipe struct tnode.

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 << ends;
            cout << bantu->kata<<ends;
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
```

```
    }
}
int main()
{
    init();
    insertDepan(3, "satu");

    insertBelakang(5, "dua");

    insertDepan(2, "tiga");

    insertDepan(1, "empat");

    hapusDepan();

    hapusBelakang();

    insertTengah(7,"lima", 2);

    hapusTengah(2);

    ubahDepan(1,"enam") ;

    ubahBelakang(8,"tujuh");

    ubahTengah(11,"delapan", 2);
    tampil();
    return 0;
}
```

Screenshoot program:

```
PROBLEMS    OUTPUT    TERMINAL    ...                    Code

PS D:\Nathhh\matkul\smst 2\praktikum struktur data\modul 3 - linked li
st> cd "d:\Nathhh\matkul\smst 2\praktikum struktur data\modul 3 - link
ed list\" ; if ($?) { g++ guided1.cpp -o guided1 } ; if ($?) { .\guide
d1 }
1enam11tujuh
PS D:\Nathhh\matkul\smst 2\praktikum struktur data\modul 3 - linked li
st>
```

Deskripsi program:

Program yang diberikan adalah implementasi dari sebuah single linked list non-circular dalam bahasa pemrograman C++. Single linked list merupakan struktur data yang terdiri dari serangkaian node, dimana setiap node menyimpan elemen data serta pointer yang menunjuk ke node berikutnya.

2. Guided 2

Source code

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

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

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;

    DoublyLinkedList() {
        head = nullptr;
```

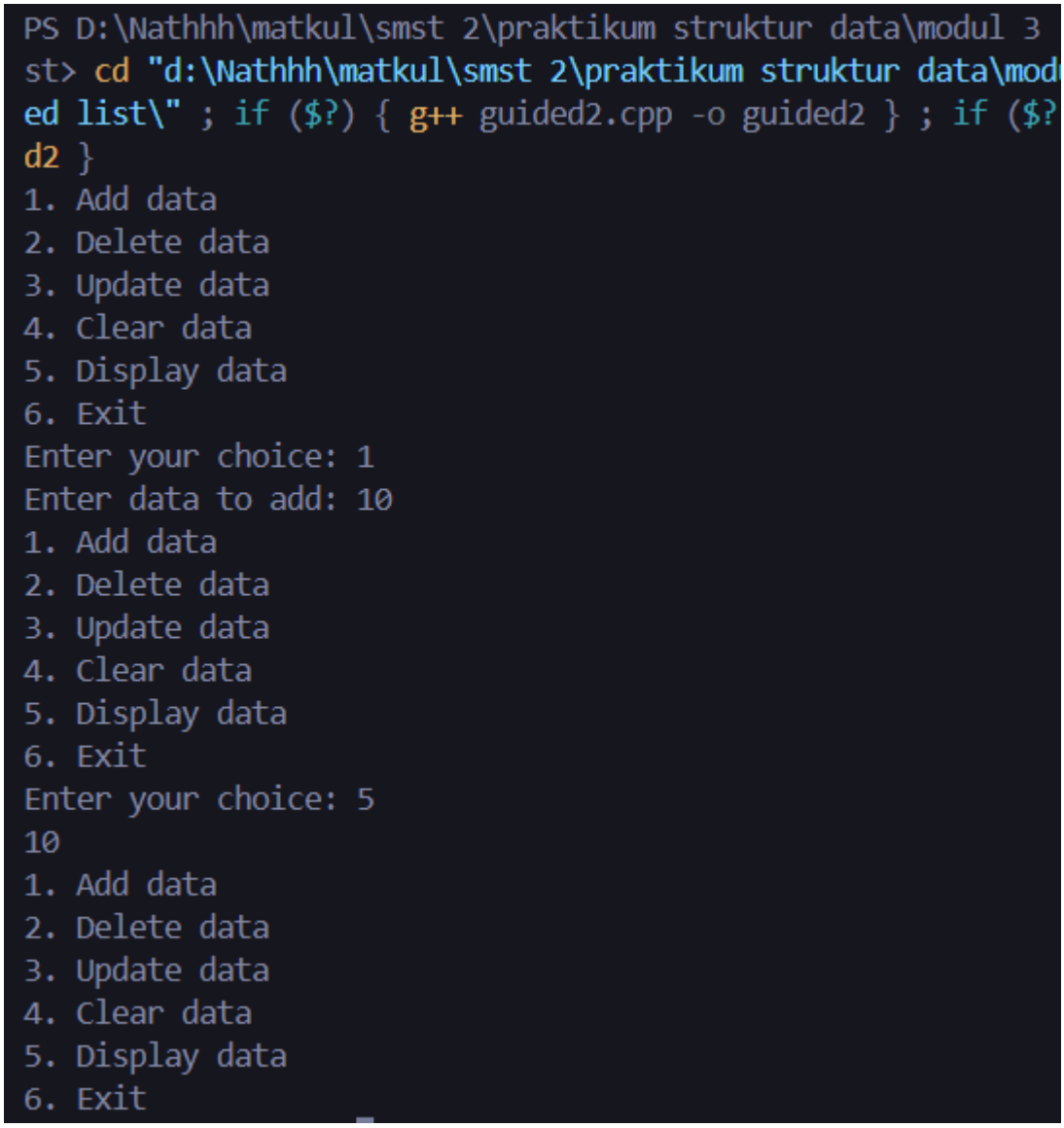
```
        tail = nullptr;
    }

    void push(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->prev = nullptr;
        newNode->next = head;

        if (head != nullptr) {
            head->prev = newNode;
        } else {
            tail = newNode;
        }

        head = newNode;
    }

    void pop() {
        if (head == nullptr) {
            return;
        }

        Node* temp = head;
        head = head->next;

        if (head != nullptr) {
            head->prev = nullptr;
        } else {
            tail = nullptr;
        }

        delete temp;
    }

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

```
            }
            current = current->next;
        }
        return false;
    }

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

    void display() {
        Node* current = head;
        while (current != nullptr) {
            cout << current->data << " ";
            current = current->next;
        }
        cout << endl;
    }
};

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

        int choice;
        cout << "Enter your choice: ";
        cin >> choice;
```

```
        switch (choice) {
            case 1: {
                int data;
                cout << "Enter data to add: ";
                cin >> data;
                list.push(data);
                break;
            }
            case 2: {
                list.pop();
                break;
            }
            case 3: {
                int oldData, newData;
                cout << "Enter old data: ";
                cin >> oldData;
                cout << "Enter new data: ";
                cin >> newData;
                bool updated = list.update(oldData, newData);
                if (!updated) {
                    cout << "Data not found" << endl;
                }
                break;
            }
            case 4: {
                list.deleteAll();
                break;
            }
            case 5: {
                list.display();
                break;
            }
            case 6: {
                return 0;
            }
            default: {
                cout << "Invalid choice" << endl;
                break;
            }
        }
    }
```

```
    return 0;
}
```

Screenshoot program:



Deskripsi program:

Program tersebut adalah implementasi Doubly Linked List yang memungkinkan pengguna untuk menambah, menghapus, mengubah, membersihkan, dan menampilkan isi dari linked list. Struktur data Doubly Linked List digunakan untuk menyimpan data dengan setiap node memiliki dua pointer, yaitu pointer ke node

sebelumnya dan pointer ke node berikutnya. Kelas Node merepresentasikan setiap node dengan menyimpan data dan dua pointer, yaitu prev dan next. Kelas DoublyLinkedList mengatur operasi-operasi pada linked list seperti penambahan (push), penghapusan (pop), pengubahan (update), pembersihan (deleteAll), dan penampilan isi (display). Program berinteraksi dengan pengguna melalui menu yang memungkinkan pemilihan aksi untuk melakukan manipulasi pada linked list. Fungsi main mengatur loop utama untuk terus menampilkan menu hingga pengguna memilih untuk keluar dari program. Program memberikan kemudahan dalam pengelolaan data menggunakan Doubly Linked List.

LATIHAN KELAS - UNGUIDED

1.

Soal mengenai Single Linked List

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 <iomanip>
```

```cpp
using namespace std;

// Deklarasi Struct Node
struct Node {
    string Nama297;
    int Umur297;
    Node* next;
};

Node* head;
Node* tail;

// Inisialisasi Node
void inisialisasi297() {
    head = NULL;
    tail = NULL;
}

// Pengecekan apakah linked list kosong
bool cek297() {
    return head == NULL;
}

// Tambah Node di depan
void depan297(string name, int age) {
    Node* baru = new Node;
    baru->Nama297 = name;
    baru->Umur297 = age;
    baru->next = NULL;

    if (cek297()) {
        head = tail = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

// Tambah Node di belakang
void belakang297(string name, int age) {
    Node* baru = new Node;
```

```
    baru->Nama297 = name;
    baru->Umur297 = age;
    baru->next = NULL;

    if (cek297()) {
        head = tail = baru;
    } else {
        tail->next = baru;
        tail = baru;
    }
}

// Hitung Jumlah Node
int jumlahlist297() {
    int jumlah = 0;
    Node* hitung = head;

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

    return jumlah;
}

// Tambah Node di tengah
void tengah297(string name, int age, int posisi) {
    if (posisi < 1 || posisi > jumlahlist297()) {
        cout << "Tidak terjangkau!" << endl;
    } else if (posisi == 1) {
        cout << "Bukan di tengah." << endl;
    } else {
        Node* baru = new Node();
        baru->Nama297 = name;
        baru->Umur297 = 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 hapus297() {
    if (!cek297()) {
        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 hapusbelakang297() {
    if (!cek297()) {
        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 hapustengah297(int posisi) {
    if (posisi < 1 || posisi > jumlahlist297()) {
        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 ubahdepan297(string name, int age) {
    if (!cek297()) {
```

```
        head->Nama297 = name;
        head->Umur297 = age;
    } else {
        cout << "Tidak ada yang berubah!" << endl;
    }
}

// Ubah data di tengah
void ubahtengah297(string name, int age, int posisi) {
    if (!cek297()) {
        if (posisi < 1 || posisi > jumlahlist297()) {
            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->Nama297 = name;
            bantu->Umur297 = age;
        }
    } else {
        cout << "Kosong!" << endl;
    }
}

// Ubah data di belakang
void ubahbelakang297(string name, int age) {
    if (!cek297()) {
        tail->Nama297 = name;
        tail->Umur297 = age;
    } else {
        cout << "Kosong" << endl;
    }
}
```

```cpp
// Hapus semua Node
void hapuslist297() {
    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 tampillist297() {
    Node* bantu = head;

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

    if (!cek297()) {
        while (bantu != NULL) {
            cout << left << setw(15) << bantu->Nama297 << right
<< setw(4) << bantu->Umur297 << endl;
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "Kosong!" << endl;
    }
}

int main() {
    inisialisasi297(); // Inisialisasi Linked List

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

    depan297("Karin", 18);
```

```
    depan297("Hoshino", 18);
    depan297("Akechi", 20);
    depan297("Yusuke", 19);
    depan297("Michael", 18);
    depan297("Jane", 20);
    depan297("John", 19);
    depan297("Nadhif", 19); // Mengubah "Albar" menjadi "Nadhif"

    tampillist297();

    // Menjawab poin b
    cout << "-=-=-= (B) Hapus data 'Akechi' =-=-=-" << endl;
    hapustengah297(6);
    tampillist297();

    // Menjawab poin c
    cout << "-=-=-= (C) Tambah data 'Futaba (18)' diantara John &
Jane =-=-=-" << endl;
    tengah297("Futaba", 18, 3);
    tampillist297();

    // Menjawab poin d
    cout << "-=-=-= (D) Tambah data 'Igor (20)' di awal =-=-=-"
<< endl;
    depan297("Igor", 20);
    tampillist297();

    // Menjawab poin e & f
    cout << "-=-=-= (E) Ubah data 'Michael' menjadi 'Reyn (18)'
=-=-=-" << endl;
    cout << "-=-=-= (F) Tampilan Akhir =-=-=-" << endl;
    ubahtengah297("Reyn", 18, 6);
    tampillist297();

    return 0;
}
```

Screenshoot program

```
PS D:\Nathhh\matkul\smst 2\praktikum struktur data\modul 3 - linked list> cd "d:\Nathhh\matkul\s
m struktur data\modul 3 - linked list\" ; if ($?) { g++ unguided1.cpp -o unguided1 } ; if ($?) {

(A.)-=-=-= SELAMAT DATANG =-=-=-
-Nama-          -Usia-
Nadhif            19
John              19
Jane              20
Michael           18
Yusuke            19
Akechi            20
Hoshino           18
Karin             18

-=-=-= (B) Hapus data 'Akechi' =-=-=-
-Nama-          -Usia-
Nadhif            19
John              19
Jane              20
Michael           18
Yusuke            19
Hoshino           18
Karin             18

-=-=-= (C) Tambah data 'Futaba (18)' diantara John & Jane =-=-=-
-Nama-          -Usia-
Nadhif            19
John              19
Futaba            18
Jane              20
Michael           18
Yusuke            19
Hoshino           18
Karin             18

-=-=-= (D) Tambah data 'Igor (20)' di awal =-=-=-
-Nama-          -Usia-
Igor              20
Nadhif            19
John              19
Futaba            18
Jane              20
Michael           18
Yusuke            19
Hoshino           18
Karin             18

-=-=-= (E) Ubah data 'Michael' menjadi 'Reyn (18)' =-=-=-
-=-=-= (F) Tampilan Akhir =-=-=-
-Nama-          -Usia-
Igor              20
Nadhif            19
John              19
Futaba            18
Jane              20
Reyn              18
Yusuke            19
Hoshino           18
Karin             18
```

Deskripsi program:

Program ini menggunakan linked list singly linked list dengan struktur data `Node` yang mengelompokkan data bersama dengan pointer ke elemen berikutnya. `head` dan `tail` menyimpan alamat elemen pertama dan terakhir dalam linked list. Berbagai fungsi disediakan untuk menambah, menghapus, dan mengubah data dalam linked list.

Fungsi `depan297()` menambah data di depan, sementara `belakang297()` menambah data di belakang. `jumlahlist297()` menghitung jumlah elemen dalam linked list. Fungsi `tengah297()` menambah data di posisi tengah dengan parameter posisi yang menunjukkan posisi data yang akan ditambah.

Fungsi `hapus297()` menghapus data di depan, `hapusbelakang297()` menghapus data di belakang, dan `hapustengah297()` menghapus data di tengah.

Fungsi `ubahdepan297()` mengubah data di depan, `ubahtengah297()` mengubah data di tengah, dan `ubahbelakang297()` mengubah data di belakang. `hapuslist297()` menghapus semua data dalam linked list, sementara `tampillist297()` menampilkan semua data dalam linked list.

Dalam `main()`, program melakukan berbagai operasi pada linked list seperti menambah, menghapus, dan mengubah data. Hasil dari operasi ini ditampilkan melalui `tampillist297()`. Program juga menyediakan menu untuk memilih operasi dan meminta input data untuk operasi tersebut.

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	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

script:

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

```cpp
#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;
}
```

Output:

```
PS D:\Nathhh\matkul\smst 2\praktikum struktur data\modul 3 - linked list> cd "d:\Nathhh\matkul\
3 - linked list\" ; if ($?) { g++ unguided2.cpp -o unguided2 } ; if ($?) { .\unguided2 }

==========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: 5
Input posisi yang ingin dihapus: 1

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
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:
```

Deskripsi Program:

Program yang diberikan adalah sebuah simulasi sederhana dari sebuah toko skincare menggunakan struktur data double linked list. Program ini memungkinkan pengguna untuk melakukan berbagai operasi seperti menambah, menghapus, memperbarui, menampilkan, serta menyisipkan dan menghapus data pada posisi tertentu dalam daftar produk skincare yang tersedia. Setiap operasi dapat diakses melalui menu yang disediakan dalam sebuah loop utama. Ini adalah contoh implementasi dasar dari struktur data double linked list untuk aplikasi pengelolaan data produk dalam sebuah toko.

Berikut adalah penjelasan dan deskripsi singkat dari setiap bagian program:

1. Deklarasi Class Node: Mendefinisikan struktur node yang berisi informasi tentang nama produk (Nama_Produk), harga produk (harga), serta pointer ke node sebelumnya (prev) dan node selanjutnya (next).
2. Deklarasi Class DoublyLinkedList: Mendefinisikan struktur dari linked list itu sendiri, yang memiliki dua pointer, yaitu head dan tail. head adalah pointer yang menunjuk ke node pertama dalam linked list, sedangkan tail adalah pointer yang menunjuk ke node terakhir.
3. Metode tambahproduk140: Menambahkan sebuah produk baru ke dalam linked list di bagian atas (head). Setiap kali menambahkan produk baru, program membuat sebuah node baru, mengatur nilainya, dan memperbarui pointer head dan prev.
4. Metode hapusproduk140: Menghapus produk teratas dari linked list. Program menghapus node yang ditunjuk oleh head, memperbarui head ke node selanjutnya (jika ada), dan membersihkan memori dari node yang dihapus.
5. Metode ubahproduk140: Mengubah data produk berdasarkan nama produk. Program melakukan pencarian linear di dalam linked list untuk menemukan node dengan nama produk yang sesuai, lalu mengubah nilai produk tersebut.
6. Metode sisipposisi140: Menambahkan data produk pada posisi tertentu dalam linked list. Program melakukan pencarian linear untuk menemukan posisi yang dituju, lalu menyisipkan node baru di antara node-node yang sudah ada.
7. Metode hapusposisi140: Menghapus data produk pada posisi tertentu dalam linked list. Program melakukan pencarian linear untuk menemukan posisi yang dituju, lalu menghapus node pada posisi tersebut dan mengatur ulang pointer yang terlibat.
8. Metode hapussemua140: Menghapus semua data produk dari linked list. Program melakukan iterasi melalui setiap node dalam linked list, menghapusnya satu per satu, dan mengatur pointer head dan tail ke nullptr.
9. Metode tampilan140: Menampilkan semua data produk yang ada dalam linked list. Program melakukan iterasi melalui setiap node dalam linked list dan mencetak informasi nama produk dan harganya ke layar.
10. Fungsi main: Tempat program dimulai. Pada bagian ini, objek list dari kelas DoublyLinkedList dibuat. Produk-produk awal ditambahkan ke dalam linked list. Kemudian, program memasuki loop tak terbatas yang menampilkan menu

utama kepada pengguna, memproses input pengguna, dan menjalankan operasi yang sesuai berdasarkan pilihan pengguna.

BAB IV

KESIMPULAN

Secara singkat, perbedaan antara single linked list dan double linked list adalah sebagai berikut:

1. **Single Linked List**:
    - Keunggulan: Penggunaan ruang penyimpanan yang lebih efisien karena hanya menggunakan satu pointer (**next**) untuk menunjukkan ke node berikutnya.
    - Keterbatasan: Terbatas dalam navigasi maju saja, sehingga sulit untuk melakukan operasi seperti mencari node sebelumnya atau menghapus node tertentu dengan mudah.
2. **Double Linked List**:
    - Keunggulan: Menawarkan fleksibilitas yang lebih besar karena setiap node memiliki dua pointer (**prev** dan **next**), memungkinkan navigasi maju dan mundur serta operasi penghapusan dan penyisipan di tengah-tengah list dengan lebih mudah.
    - Keterbatasan: Memerlukan penggunaan ruang penyimpanan yang lebih besar karena setiap node menyimpan dua pointer.

Pemilihan antara kedua jenis linked list tergantung pada kebutuhan spesifik aplikasi dan prioritas performa yang diinginkan. Jika ruang penyimpanan menjadi perhatian utama dan navigasi maju saja sudah cukup untuk kebutuhan aplikasi, maka single linked list dapat menjadi pilihan yang baik. Namun, jika aplikasi memerlukan fleksibilitas navigasi maju dan mundur serta operasi penghapusan dan penyisipan di tengah-tengah list, maka double linked list mungkin lebih sesuai meskipun dengan pengorbanan sedikit ruang penyimpanan tambahan.