LAPORAN PRAKTIKUM

MODUL III
SINGLE AND DOUBLE LINKED LIST

Disusun oleh:
Bayu Kuncoro Adi
NIM: 2311102031

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

A. TUJUAN PRAKTIKUM

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

BAB II

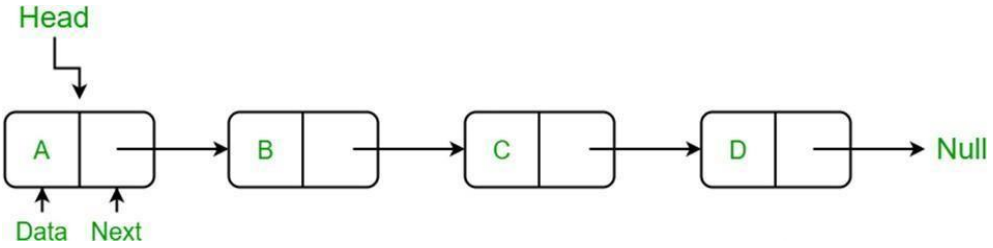
DASAR TEORI

A. DASAR TEORI

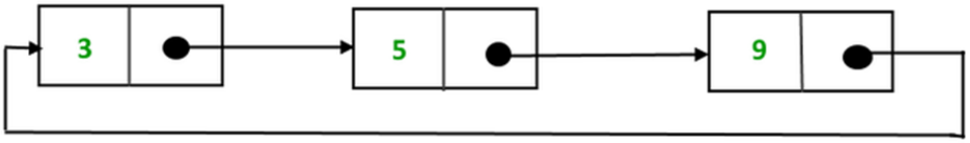
1. Single Linked List

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



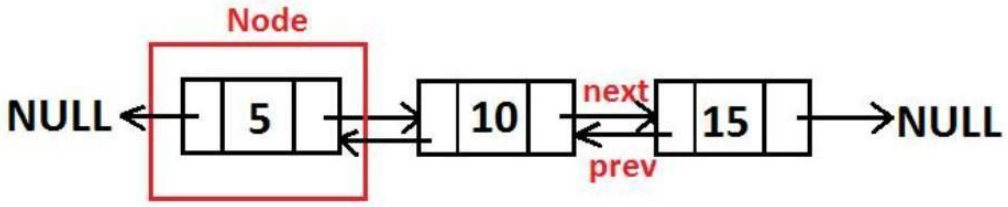
linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.

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 setiapsimpul yaitu pointer prev yang menunjuk ke simpul sebelumnya. Denganadanya pointer prev, Double Linked List memungkinkan untuk melakukanoperasi penghapusan dan penambahan pada simpul mana saja secara efisien.Setiap simpul pada Double Linked List memiliki tiga elemen penting, yaituelemen 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 yangmembutuhkan operasi tersebut. Selain itu, Double Linked List juga memungkinkan kita untuk melakukan traversal pada list baik dari depan (head) maupun dari belakang (tail) dengan mudah. Namun, kekurangan dari Double Linked List adalah penggunaan memori yang lebih besar dibandingkan dengan Single Linked List, karena setiap simpul membutuhkan satu pointer tambahan. Selain itu, Double Linked List juga membutuhkan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan jika dibandingkan dengan Single Linked List.

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



Di dalam sebuah linked list, ada 2 pointer yang menjadi penunjuk utama, yakni pointer HEAD yang menunjuk pada node pertama di dalam linked list itu sendiri dan pointer TAIL yang menunjuk pada node paling akhir di dalam linked list. Sebuah linked list dikatakan kosong apabila isi pointer head adalah NULL. Selain itu, nilai pointer prev dari HEAD selalu NULL, karena merupakan data pertama. Begitu pula dengan pointer next dari TAIL yang selalu bernilai NULL sebagai penanda data terakhir.

BAB III

GUIDED

1. Guided 1

Single Linked List

```
#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;
```

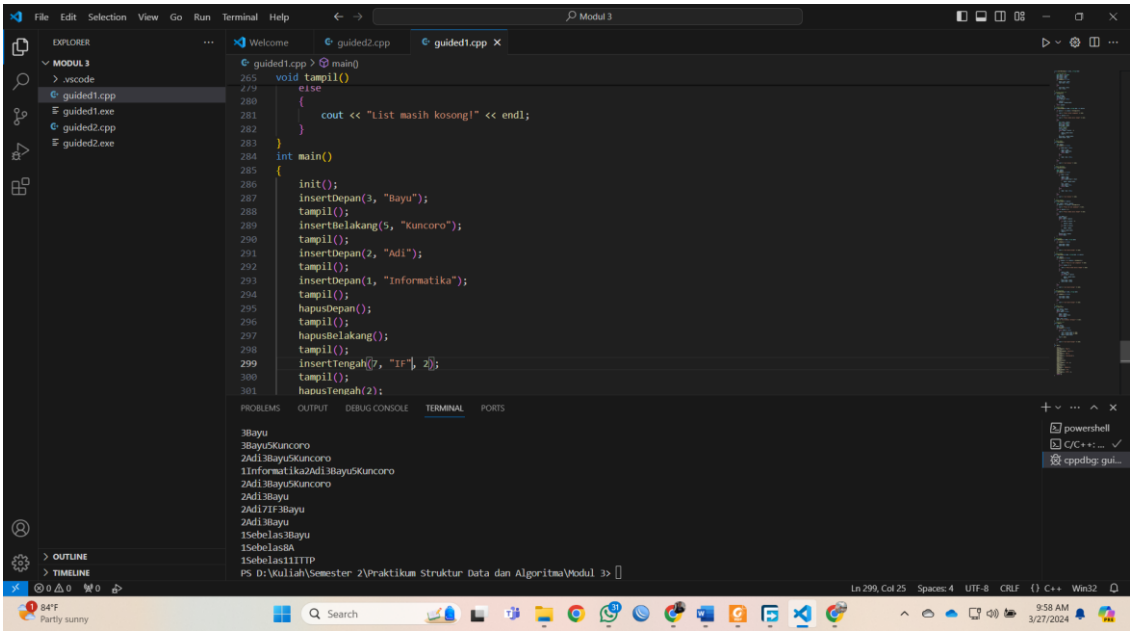
```
            }
            tail = bantu;
            tail->next = NULL;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}
// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *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 << ends;
            cout << bantu->kata << ends;
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
int main()
{
    init();
    insertDepan(3, "Bayu");
    tampil();
    insertBelakang(5, "Kuncoro");
    tampil();
    insertDepan(2, "Adi");
    tampil();
    insertDepan(1, "Informatika");
```

```
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, "IF", 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1, "Sebelas");
    tampil();
    ubahBelakang(8, "A");
    tampil();
    ubahTengah(11, "ITTP", 2);
    tampil();
    return 0;
}
```

Screenshoot Program:



Deskripsi Program

Program di atas merupakan implementasi dari struktur data linked list berbasis node tunggal (single linked list non-circular) dalam bahasa C++. Program ini memungkinkan pengguna untuk melakukan operasi penambahan (depan, belakang, dan tengah), penghapusan (depan, belakang, dan tengah), serta perubahan data pada node-node yang ada dalam linked list. Terdapat fungsi-fungsi seperti inisialisasi linked list, pengecekan apakah linked list kosong, menghitung jumlah node dalam linked list, dan menampilkan isi linked list. Selain itu, program ini juga memiliki fungsi main yang melakukan serangkaian operasi untuk menguji fungsionalitas dari linked list yang telah diimplementasikan.

2. **Guided 2**

Double Linked List

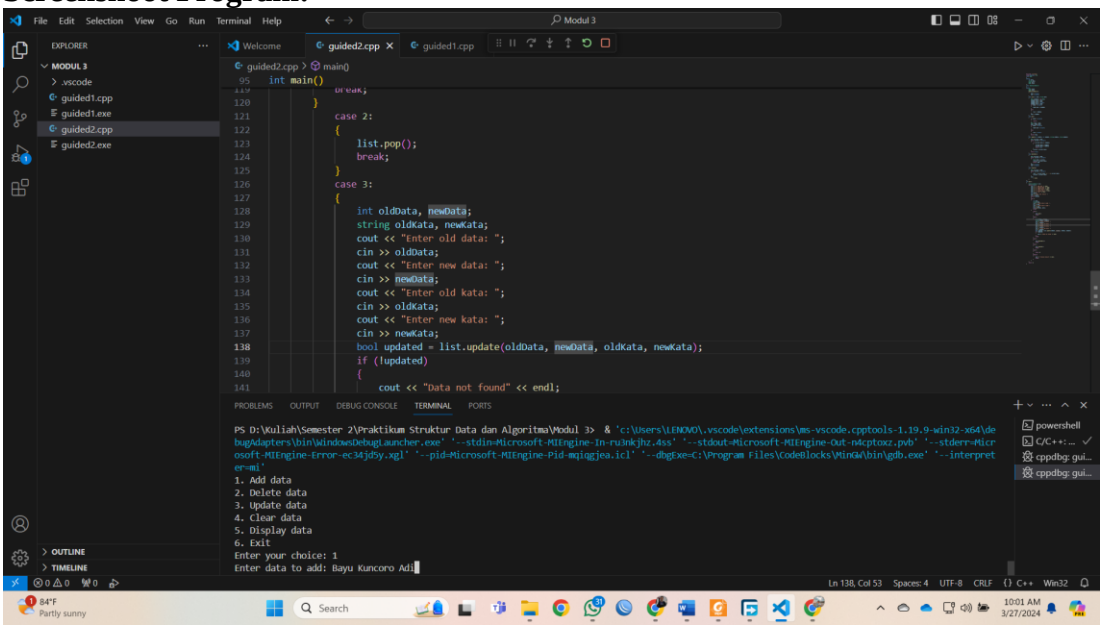
```
#include <iostream>
using namespace std;
class Node
{
public:
    int data;
    string kata;
    Node *prev;
    Node *next;
};
class DoublyLinkedList
{
public:
    Node *head;
    Node *tail;
    DoublyLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }
    void push(int data, 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 di atas merupakan implementasi dari struktur data Doubly Linked List dalam bahasa C++. Struktur data Doubly Linked List memiliki node yang memiliki dua pointer, yaitu pointer prev yang menunjuk ke node sebelumnya dan pointer next yang menunjuk ke node selanjutnya. Program ini menggunakan dua kelas, yaitu kelas Node untuk merepresentasikan setiap node dalam Doubly Linked List, dan kelas DoublyLinkedList untuk mengelola operasi-operasi yang terkait dengan Doubly Linked List. Kelas DoublyLinkedList memiliki beberapa fungsi seperti push untuk menambahkan data ke depan linked list, pop untuk menghapus data dari depan linked list, update untuk mengubah data pada linked list, deleteAll untuk menghapus seluruh data dalam linked list, dan display untuk menampilkan isi linked list.

Selain itu, program ini juga memiliki fungsi main yang memberikan antarmuka pengguna sederhana untuk berinteraksi dengan linked list yang telah diimplementasikan. Pengguna diberikan opsi untuk menambahkan data baru, menghapus data, mengubah data, menghapus seluruh data, menampilkan isi linked list, dan keluar dari program. Setiap opsi yang dipilih pengguna akan memicu fungsi terkait dalam kelas DoublyLinkedList untuk melakukan operasi yang sesuai. Dengan demikian, program ini memberikan kontrol penuh kepada pengguna untuk mengelola dan memanipulasi data dalam Doubly Linked List.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

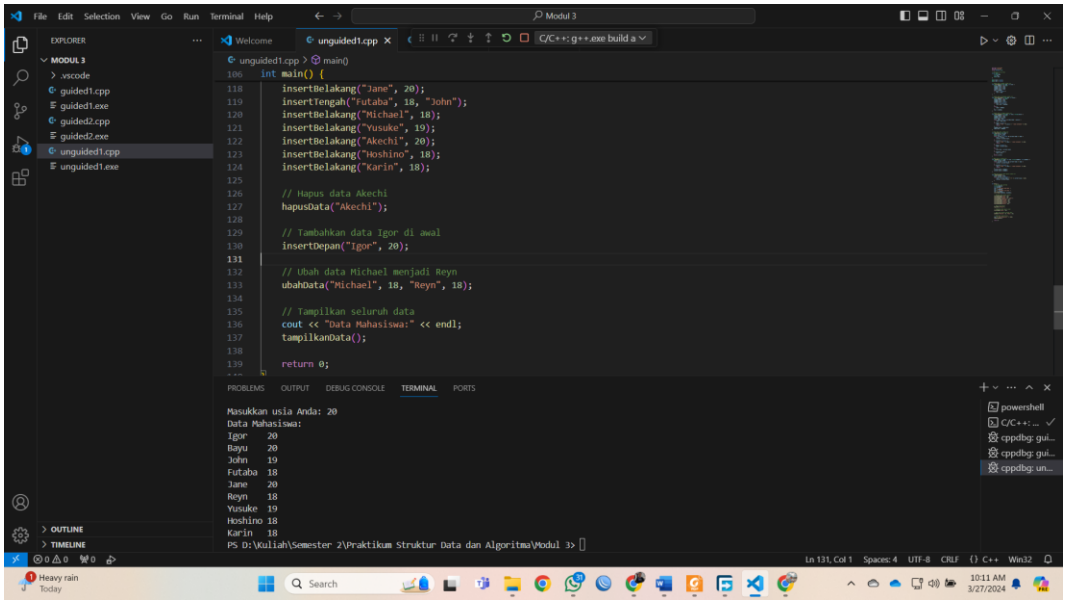
```
#include <iostream>
#include <string>
using namespace std;

// Deklarasi Struct Node
struct Node {
    string nama;
    int usia;
    Node* next;
};

Node* head = nullptr;
Node* tail = nullptr;

// Tambah Data di Depan Linked List
void insertDepan(string nama, int usia) {
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    newNode->next = head;
    head = newNode;
    if (tail == nullptr) {
        tail = head;
    }
}

// Tambah Data di Belakang Linked List
void insertBelakang(string nama, int usia) {
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    newNode->next = nullptr;
    if (tail != nullptr) {
        tail->next = newNode;
    }
```

```
    else {
        head = newNode;
    }
    tail = newNode;
}

// Tambah Data di Tengah Linked List
void insertTengah(string nama, int usia, string posisi) {
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    Node* temp = head;
    while (temp != nullptr && temp->nama != posisi) {
        temp = temp->next;
    }
    if (temp == nullptr) {
        cout << "Data " << posisi << " tidak ditemukan" << endl;
        return;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}

// Hapus Data dari Linked List
void hapusData(string nama) {
    Node* current = head;
    Node* prev = nullptr;
    while (current != nullptr && current->nama != nama) {
        prev = current;
        current = current->next;
    }
    if (current == nullptr) {
        cout << "Data " << nama << " tidak ditemukan" << endl;
        return;
    }
    if (prev == nullptr) {
        head = current->next;
    }
    else {
        prev->next = current->next;
    }
```

```
    if (current == tail) {
        tail = prev;
    }
    delete current;
}

// Ubah Data di Linked List
void ubahData(string nama, int usia, string namaBaru, int
usiaBaru) {
    Node* current = head;
    while (current != nullptr && current->nama != nama) {
        current = current->next;
    }
    if (current == nullptr) {
        cout << "Data " << nama << " tidak ditemukan" << endl;
        return;
    }
    current->nama = namaBaru;
    current->usia = usiaBaru;
}

// Tampilkan Seluruh Data dalam Linked List
void tampilkanData() {
    Node* current = head;
    while (current != nullptr) {
        cout << current->nama << "\t" << current->usia << endl;
        current = current->next;
    }
}

int main() {
    // Masukkan data pertama
    string namaAnda;
    int usiaAnda;
    cout << "Masukkan nama Anda: ";
    cin >> namaAnda;
    cout << "Masukkan usia Anda: ";
    cin >> usiaAnda;
    insertDepan(namaAnda, usiaAnda);

    // Masukkan data sesuai urutan
```

```
    insertBelakang("John", 19);
    insertBelakang("Jane", 20);
    insertTengah("Futaba", 18, "John");
    insertBelakang("Michael", 18);
    insertBelakang("Yusuke", 19);
    insertBelakang("Akechi", 20);
    insertBelakang("Hoshino", 18);
    insertBelakang("Karin", 18);

    // Hapus data Akechi
    hapusData("Akechi");

    // Tambahkan data Igor di awal
    insertDepan("Igor", 20);

    // Ubah data Michael menjadi Reyn
    ubahData("Michael", 18, "Reyn", 18);

    // Tampilkan seluruh data
    cout << "Data Mahasiswa:" << endl;
    tampilkanData();

    return 0;
}
```

Screenshoot Program:



Penjelasan Program:

Program di atas adalah implementasi dari Single Linked List Non-Circular dalam bahasa C++. Program ini memiliki beberapa fungsi yang memungkinkan pengguna untuk memanipulasi data dalam linked list. Pertama-tama, terdapat fungsi-fungsi untuk menambahkan data ke linked list, baik di depan, belakang, maupun di tengah. Fungsi-fungsi tersebut adalah `insertDepan`, `insertBelakang`, dan `insertTengah`. Selanjutnya, program juga menyediakan fungsi untuk menghapus data dari linked list, yaitu fungsi `hapusData`. Kemudian, terdapat fungsi `ubahData` yang digunakan untuk mengubah data yang sudah ada dalam linked list. Terakhir, program memiliki fungsi `tampilkanData` untuk menampilkan semua data yang ada dalam linked list.

Pada fungsi `main`, program pertama-tama meminta pengguna untuk memasukkan nama dan usia mereka sendiri. Kemudian, program memasukkan data sesuai dengan urutan yang ditentukan menggunakan fungsi-fungsi untuk menambahkan data. Setelah itu, program menghapus data Akechi dan menambahkan data Igor di awal linked list. Selanjutnya, program mengubah data Michael menjadi Reyn. Terakhir, program menampilkan seluruh data mahasiswa yang ada dalam linked list.

Dengan menggunakan linked list, program ini memungkinkan penyimpanan dan manipulasi data yang fleksibel, serta memberikan kontrol kepada pengguna untuk mengatur data sesuai dengan kebutuhan mereka. Dengan demikian, program ini dapat digunakan sebagai dasar untuk pengembangan sistem yang lebih kompleks yang melibatkan pengelolaan data mahasiswa.

2. Unguided 2

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

// Deklarasi Struct Node
struct Node {
    string nama_produk;
    int harga;
    Node* prev;
    Node* next;
};

Node* head = nullptr;
Node* tail = nullptr;

// Tambah Data di Depan Linked List
void insertDepan(string nama_produk, int harga) {
    Node* newNode = new Node;
    newNode->nama_produk = nama_produk;
    newNode->harga = harga;
    newNode->prev = nullptr;
    newNode->next = head;
    if (head != nullptr) {
        head->prev = newNode;
    }
    head = newNode;
    if (tail == nullptr) {
        tail = head;
    }
}

// Tambah Data di Belakang Linked List
void insertBelakang(string nama_produk, int harga) {
    Node* newNode = new Node;
    newNode->nama_produk = nama_produk;
    newNode->harga = harga;
    newNode->next = nullptr;
    if (tail != nullptr) {
        tail->next = newNode;
    }
```

```
    else {
        head = newNode;
    }
    newNode->prev = tail;
    tail = newNode;
}

// Tambah Data di Tengah Linked List
void insertTengah(string nama_produk, int harga, string posisi) {
    Node* newNode = new Node;
    newNode->nama_produk = nama_produk;
    newNode->harga = harga;

    Node* temp = head;
    while (temp != nullptr && temp->nama_produk != posisi) {
        temp = temp->next;
    }

    if (temp == nullptr) {
        cout << "Produk " << posisi << " tidak ditemukan" <<
endl;
        return;
    }

    newNode->next = temp->next;
    newNode->prev = temp;
    if (temp->next != nullptr) {
        temp->next->prev = newNode;
    }
    temp->next = newNode;
}

// Hapus Data dari Linked List
void hapusData(string nama_produk) {
    Node* current = head;
    while (current != nullptr && current->nama_produk !=
nama_produk) {
        current = current->next;
    }

    if (current == nullptr) {
```

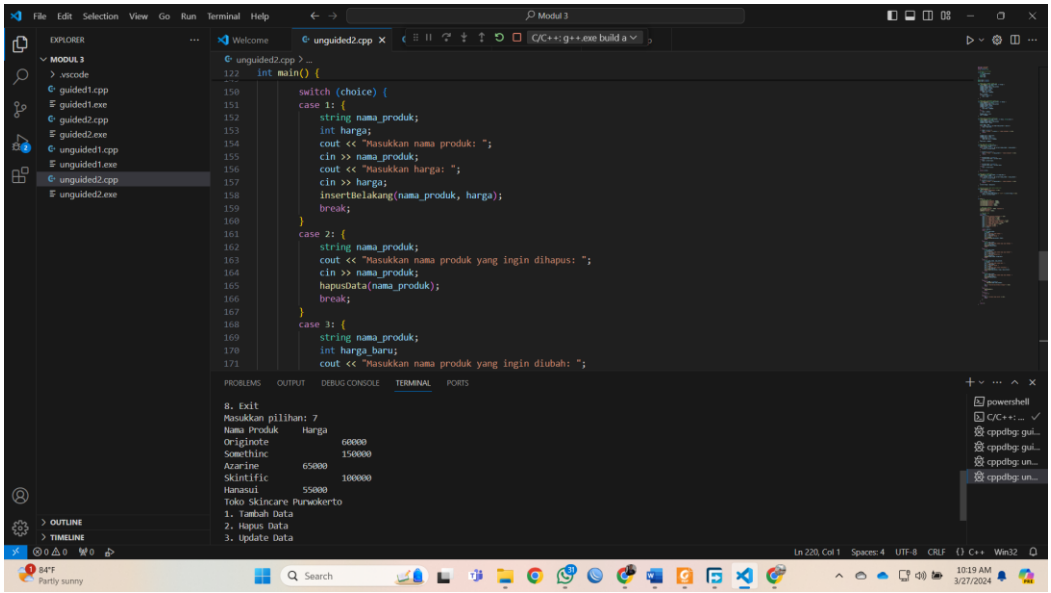
```
        cout << "Produk " << nama_produk << " tidak ditemukan" <<
endl;
        return;
    }

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

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

    delete current;
}

// Ubah Data di Linked List
void ubahData(string nama_produk, int harga_baru) {
    Node* current = head;
    while (current != nullptr && current->nama_produk !=
nama_produk) {
        current = current->next;
    }
    if (current == nullptr) {
        cout << "Produk " << nama_produk << " tidak ditemukan" <<
endl;
        return;
    }
    current->harga = harga_baru;
}

// Tampilkan Seluruh Data dalam Linked List
void tampilkanData() {
    cout << "Nama Produk\tHarga" << endl;
    Node* current = head;
    while (current != nullptr) {
```

```
        cout << current->nama_produk << "\t\t" << current->harga
<< endl;
        current = current->next;
    }
}

int main() {
    // Masukkan data awal
    insertBelakang("Originote", 60000);
    insertBelakang("Somethinc", 150000);
    insertBelakang("Skintific", 100000);
    insertBelakang("Wardah", 50000);
    insertBelakang("Hanasui", 30000);

    // Modifikasi operasi
    insertTengah("Azarine", 65000, "Somethinc");
    hapusData("Wardah");
    ubahData("Hanasui", 55000);

    // Tampilkan menu
    int choice;
    while (true) {
        cout << "Toko Skincare Purwokerto" << endl;
        cout << "1. Tambah Data" << endl;
        cout << "2. Hapus Data" << endl;
        cout << "3. Update Data" << endl;
        cout << "4. Tambah Data Urutan Tertentu" << endl;
        cout << "5. Hapus Data Urutan Tertentu" << endl;
        cout << "6. Hapus Seluruh Data" << endl;
        cout << "7. Tampilkan Data" << endl;
        cout << "8. Exit" << endl;
        cout << "Masukkan pilihan: ";
        cin >> choice;

        switch (choice) {
        case 1: {
            string nama_produk;
            int harga;
            cout << "Masukkan nama produk: ";
            cin >> nama_produk;
            cout << "Masukkan harga: ";
```

```
            cin >> harga;
            insertBelakang(nama_produk, harga);
            break;
        }
        case 2: {
            string nama_produk;
            cout << "Masukkan nama produk yang ingin dihapus: ";
            cin >> nama_produk;
            hapusData(nama_produk);
            break;
        }
        case 3: {
            string nama_produk;
            int harga_baru;
            cout << "Masukkan nama produk yang ingin diubah: ";
            cin >> nama_produk;
            cout << "Masukkan harga baru: ";
            cin >> harga_baru;
            ubahData(nama_produk, harga_baru);
            break;
        }
        case 4: {
            string nama_produk, nama_setelah;
            int harga;
            cout << "Masukkan nama produk baru: ";
            cin >> nama_produk;
            cout << "Masukkan harga: ";
            cin >> harga;
            cout << "Masukkan nama produk setelahnya: ";
            cin >> nama_setelah;
            insertTengah(nama_produk, harga, nama_setelah);
            break;
        }
        case 5: {
            string nama_produk;
            cout << "Masukkan nama produk yang ingin dihapus: ";
            cin >> nama_produk;
            hapusData(nama_produk);
            break;
        }
        case 6: {
```

```
            while (head != nullptr) {
                hapusData(head->nama_produk);
            }
            cout << "Seluruh data berhasil dihapus" << endl;
            break;
        }
        case 7: {
            tampilkanData();
            break;
        }
        case 8: {
            return 0;
        }
        default: {
            cout << "Pilihan tidak valid" << endl;
            break;
        }
        }
    }

    return 0;
}
```

Screenshoot Program:



Penjelasan Program:

Program di atas merupakan sebuah implementasi dari double linked list yang digunakan untuk mengelola data produk pada sebuah toko skincare. Setiap node dalam linked list menyimpan informasi mengenai nama produk dan harganya. Program ini dilengkapi dengan berbagai fungsi seperti menambah data di belakang, menghapus data, mengubah data, menambah data di tengah atau di urutan tertentu, menghapus data di urutan tertentu, menghapus seluruh data, dan menampilkan data.

Di bagian **main()**, program dimulai dengan memasukkan data produk awal seperti Originote, Somethinc, Skintific, Wardah, dan Hanasui menggunakan fungsi **insertBelakang()**. Kemudian, dilakukan modifikasi operasi seperti menambah produk Azarine di antara Somethinc dan Skintific menggunakan fungsi **insertTengah()**, menghapus produk Wardah dengan fungsi **hapusData()**, dan mengubah produk Hanasui menjadi Cleora dengan harga 55000 menggunakan fungsi **ubahData()**.

Setelahnya, program menampilkan menu dengan pilihan operasi yang tersedia seperti tambah data, hapus data, update data, tambah data di urutan tertentu, hapus data di urutan tertentu, hapus seluruh data, dan tampilkan data. Setiap operasi diproses sesuai dengan pilihan pengguna dengan menggunakan struktur switch-case. Terakhir, program berjalan dalam loop hingga pengguna memilih untuk keluar dari program.

BAB IV

KESIMPULAN

Program-program di atas merupakan implementasi dari struktur data linked list dalam bahasa pemrograman C++.

Pertama, program pertama adalah implementasi dari Single Linked List Non-Circular yang digunakan untuk mengelola data mahasiswa. Program ini menyediakan fungsi-fungsi untuk menambahkan data mahasiswa ke linked list, menghapus data, mengubah data, dan menampilkan seluruh data yang tersimpan dalam linked list. Program ini memberikan fleksibilitas kepada pengguna untuk mengelola data mahasiswa sesuai kebutuhan.

Kedua, program kedua adalah implementasi dari Double Linked List yang digunakan untuk mengelola data produk pada sebuah toko skincare. Program ini memiliki fungsi-fungsi untuk menambahkan data produk di belakang, menghapus data, mengubah data, menambah data di tengah atau di urutan tertentu, menghapus data di urutan tertentu, menghapus seluruh data, dan menampilkan data. Dengan demikian, program ini memberikan pengguna kontrol penuh dalam pengelolaan data produk toko skincare.

Kedua program ini memanfaatkan konsep linked list yang memungkinkan penyimpanan dan manipulasi data dengan efisien serta fleksibilitas. Penggunaan linked list dalam kedua program ini memungkinkan penyimpanan data dalam urutan yang dinamis, sehingga data dapat ditambahkan, dihapus, dan diubah dengan mudah tanpa harus memindahkan seluruh elemen data.

Dengan adanya kedua program ini, pengguna dapat memanfaatkan struktur data linked list untuk mengelola data dengan lebih efektif dan efisien sesuai dengan kebutuhan aplikasi atau sistem yang sedang dikembangkan.

DAFTAR PUSTAKA

Modul 3 Single and Double Linked List Praktikum Struktur Data dan Algoritma

Modul 6 Single dan Double Linked List diakses dari https://elektro.um.ac.id/wp-content/uploads/2016/04/ASD-Modul-6-Linked-List.pdf

Single dan Double Linked List diakses dari https://www.academia.edu/82277141/SINGLE_LINKED_LIST_DOUBLE_ENDED_LIST_DOUBLY_LINKED_LIST_CIRCULAR_LINKED_LIST_DAN_ITERATOR?f_ri=1292408