LAPORAN PRAKTIKUM
MODUL 3
SINGLE AND DOUBLE LINKED LIST

Disusun oleh:

Fahri Ramadhan

NIM : 2311102024

Dosen Pengampu:

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

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2023

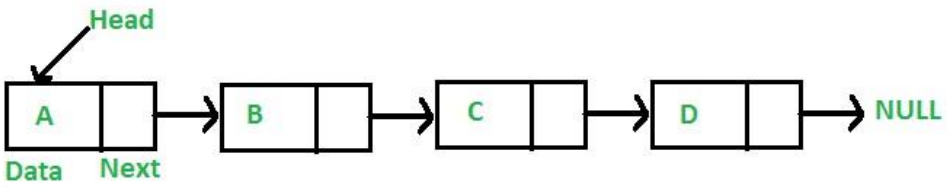
BAB I

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

Linked list adalah strukur data linier berbentuk rantai simpul di mana setiap simpul menyimpan 2 item, yaitu nilai data dan pointer ke simpul elemen berikutnya. Berbeda dengan array, elemen linked list tidak ditempatkan dalam alamat memori yang berdekatan melainkan elemen ditautkan menggunakan pointer.



Simpul pertama dari linked list disebut sebagai head atau simpul kepala. Apabila linked list berisi elemen kosong, maka nilai pointer dari head menunjuk ke NULL. Begitu juga untuk pointer berikutnya dari simpul terakhir atau simpul ekor akan menunjuk ke NULL. Ukuran elemen dari linked list dapat bertambah secara dinamis dan mudah untuk menyisipkan dan menghapus elemen karena tidak seperti array, kita hanya perlu mengubah pointer elemen sebelumnya dan elemen berikutnya untuk menyisipkan atau menghapus elemen. Linked list biasanya digunakan untuk membuat file system, adjacency list, dan hash table.

BAB III

GUIDED

1. GUIDED 1 SOURCE CODE

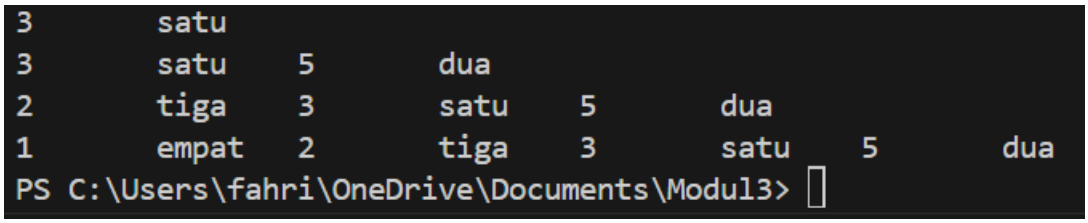
```
#include <iostream>
using namespace std;

// Declaration of struct
Node
struct Node {
    int data;
    string kata;
    Node *next;
};

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

// Initialize List
void init() {
    head = nullptr;
    tail = nullptr;
}

// Check if List is
Empty
bool isEmpty() {
    return head ==
nullptr;
}
```

```
// Insert at the
Beginning
void insertDepan(int
nilai, string kata) {
    Node *baru = new
Node;
    baru->data = nilai;
    baru->kata = kata;
    baru->next =
nullptr;

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

// Insert at the End
void insertBelakang(int
nilai, string kata) {
    Node *baru = new
Node;
    baru->data = nilai;
    baru->kata = kata;
    baru->next =
nullptr;

    if (isEmpty()) {
```

```
        head = tail =
baru;
    } else {
        tail->next =
baru;
        tail = baru;
    }
}

// Display List
void tampil() {
    if (isEmpty()) {
        cout << "List
masih kosong!" << endl;
        return;
    }

    Node *bantu = head;
    while (bantu !=
nullptr) {
        cout << bantu-
>data << "\t" << bantu-
>kata << "\t";
        bantu = bantu-
>next;
    }
    cout << endl;
}

int main() {
    init();
```

```
    insertDepan(3,
"satu");
    tampil();

    insertBelakang(5,
"dua");
    tampil();

    insertDepan(2,
"tiga");
    tampil();

    insertDepan(1,
"empat");
    tampil();

    // Add more
operations as needed

    return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Dalam program ini, kita telah berhasil mengimplementasikan Linked List tunggal non-circular menggunakan bahasa pemrograman C++. Kita dapat menambahkan, menghapus, mengubah, dan menampilkan simpul dalam Linked List menggunakan fungsi-fungsi yang telah kita buat. Linked List adalah struktur data yang fleksibel dan efisien untuk mengelola kumpulan data yang dinamis. Dengan pemahaman yang baik tentang Linked List, kita dapat mengembangkan program yang lebih kompleks dan efisien.

2. GUIDED 2

SOURCE CODE

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

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

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

    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

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

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

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

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

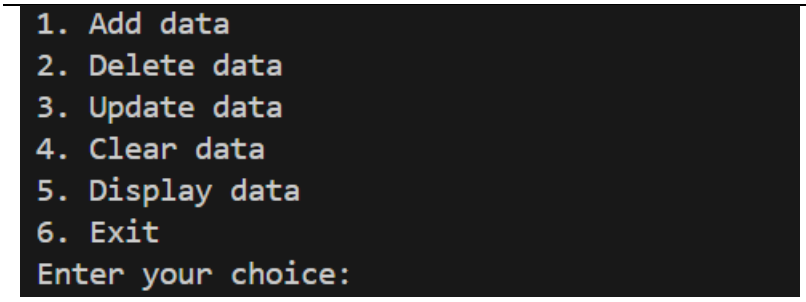
```
    bool update(int
oldData, int newData,
string newKata) {
        Node* current =
head;
        while (current
!= nullptr) {
            if (current-
>data == oldData) {
                current-
>data = newData;
                current-
>kata = newKata;
                return
true;
           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

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

```
        head = nullptr;
        tail = nullptr;
    }

    void display() {
        if (head ==
nullptr) {
            cout <<
"List is empty." <<
endl;
            return;
        }

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

int main() {
    DoublyLinkedList
list;
    while (true) {
        cout << "1. Add
data" << endl;
```

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

        switch (choice)
{
            case 1: {
                int
data;
                string
kata;
                cout <<
"Enter data to add: ";
                cin >>
data;
                cout <<
"Enter kata to add: ";
                cin >>
kata;

list.push(data, kata);
                break;
```

```
            }
            case 2: {

list.pop();
                break;
            }
            case 3: {
                int
oldData, newData;
                string
newKata;
                cout <<
"Enter old data: ";
                cin >>
oldData;
                cout <<
"Enter new data: ";
                cin >>
newData;
                cout <<
"Enter new kata: ";
                cin >>
newKata;
                bool
updated =
list.update(oldData,
newData, newKata);
                if
(!updated) {
                    cout
<< "Data not found" <<
endl;
                }
```

```
                break;
            }
            case 4: {

list.deleteAll();
                break;
            }
            case 5: {

list.display();
                break;
            }
            case 6: {
                return
0;
            }
            default: {
                cout <<
"Invalid choice" <<
endl;
                break;
            }
        }
    }
    return 0;
}
```



2.1. Ketika program menambahkan kata

DESKRIPSI PROGRAM

Pertama, kita mendefinisikan class Node yang merepresentasikan simpul dalam Doubly Linked List. Setiap simpul memiliki atribut data dan kata untuk menyimpan data dan kata, serta pointer prev dan next untuk menghubungkan simpul dengan simpul sebelumnya dan setelahnya. Selanjutnya, kita mendefinisikan class DoublyLinkedList yang merepresentasikan Doubly Linked List itu sendiri. Class ini memiliki atribut head dan tail yang merupakan pointer ke simpul pertama dan terakhir dalam Doubly Linked List.

UNGUIDED

1. UNGUIDED 1

Buatlah program menu Single Linked List Non-Circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user. Lakukan operasi berikut:

a. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah). Data pertama yang dimasukkan adalah nama dan usia anda. [Nama_anda] [Usia_anda] John 19 Jane 20 Michael 18 Yusuke 19 Akechi 20 Hoshino 18 Karin

18

b. Hapus data Akechi

c. Tambahkan data berikut diantara John dan Jane : Futaba 18

d. Tambahkan data berikut diawal : Igor 20

e. Ubah data Michael menjadi : Reyn 18

f. Tampilkan seluruh data

SOURCE CODE

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

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

class LinkedList {
private:
    Node* head;

public:
    LinkedList() {
        head = nullptr;
    }

    void insertAwal(string
nama, int usia) {
        Node* newNode = new
Node;
        newNode->nama = nama;
        newNode->usia = usia;
        newNode->next = head;
        head = newNode;
    }

    void insertAkhir(string
nama, int usia) {
        Node* newNode = new
Node;
        newNode->nama = nama;
```

```
        newNode->usia = usia;
        newNode->next =
nullptr;

        if (head == nullptr) {
            head = newNode;
            return;
        }

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

    void insertSetelah(string
nama, int usia, string
namaSebelum) {
        Node* newNode = new
Node;
        newNode->nama = nama;
        newNode->usia = usia;

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

        if (temp == nullptr) {
```

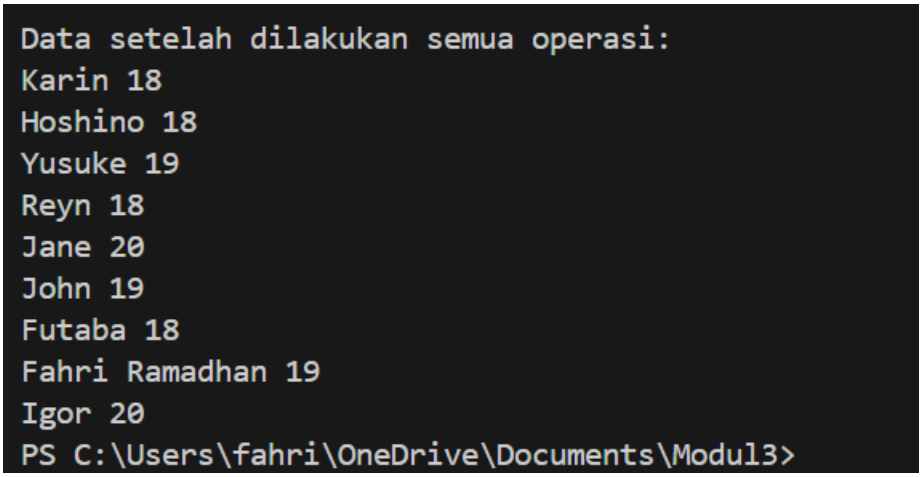
```
            cout << "Node
dengan nama " << namaSebelum
<< " tidak ditemukan." <<
endl;
            return;
        }

        newNode->next = temp-
>next;
        temp->next = newNode;
    }

    void hapus(string nama) {
        if (head == nullptr) {
            cout << "Linked
list kosong." << endl;
            return;
        }

        if (head->nama ==
nama) {
            Node* temp = head;
            head = head->next;
            delete temp;
            return;
        }

        Node* prev = head;
        Node* temp = head-
>next;
        while (temp != nullptr
&& temp->nama != nama) {
            prev = temp;
            temp = temp->next;
```

```
        }

        if (temp == nullptr) {
            cout << "Node
dengan nama " << nama << "
tidak ditemukan." << endl;
            return;
        }

        prev->next = temp-
>next;
        delete temp;
    }

    void ubah(string nama,
string namaBaru, int usiaBaru)
{
        Node* temp = head;
        while (temp != nullptr
&& temp->nama != nama) {
            temp = temp->next;
        }

        if (temp == nullptr) {
            cout << "Node
dengan nama " << nama << "
tidak ditemukan." << endl;
            return;
        }

        temp->nama = namaBaru;
        temp->usia = usiaBaru;
    }
```

```
    void tampilkan() {
        Node* temp = head;
        while (temp !=
nullptr) {
            cout << temp->nama
<< " " << temp->usia << endl;
            temp = temp->next;
        }
    }
};

int main() {
    LinkedList myList;

    myList.insertAwal("Fahri
Ramadhan", 19);
    myList.insertAwal("John",
19);
    myList.insertAwal("Jane",
20);

myList.insertAwal("Michael",
18);

myList.insertAwal("Yusuke",
19);

myList.insertAwal("Akechi",
20);

myList.insertAwal("Hoshino",
18);
    myList.insertAwal("Karin",
18);
```

```
    cout << "Data setelah
langkah (a):" << endl;
    myList.tampilkan();
    cout << endl;

    myList.hapus("Akechi");

myList.insertSetelah("Futaba",
18, "John");
    myList.insertAkhir("Igor",
20);
    myList.ubah("Michael",
"Reyn", 18);
    cout << "Data setelah
dilakukan semua operasi:" <<
endl;
    myList.tampilkan();

    return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

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

2. UNGUIDED 2

Modifikasi Guided Double Linked List dilakukan dengan penambahan operasi untuk menambah data, menghapus, dan update di tengah / di urutan tertentu yang diminta. Selain itu, buatlah agar tampilannya menampilkan Nama produk dan harga.

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Skintific	100.000
Wardah	50.000
Hanasui	30.000

Case:

1. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific
2. Hapus produk wardah
3. Update produk Hanasui menjadi Cleora dengan harga 55.000
4. Tampilkan menu seperti dibawah ini

Toko Skincare Purwokerto

1. Tambah Data

2. Hapus Data

3. Update Data

4. Tambah Data Urutan Tertentu

5. Hapus Data Urutan Tertentu

6. Hapus Seluruh Data

7. Tampilkan Data

8. Exit

Pada menu 7, tampilan akhirnya akan menjadi seperti dibawah ini :

Nama Produk	Harga
Originote	60.000
Somethinc	150.000
Azarine	65.000
Skintific	100.000
Cleora	55.000

SOURCE CODE

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

class Node {
public:
    string namaProduk;
    int harga;
    Node* prev;
    Node* next;
};

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

    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

    void push(string namaProduk,
int harga) {
        Node* newNode = new Node;
        newNode->namaProduk =
namaProduk;
        newNode->harga = harga;
        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(string
oldNamaProduk, string
newNamaProduk, int newHarga) {
        Node* current = head;
        while (current != nullptr)
{
            if (current-
>namaProduk == oldNamaProduk) {
                current-
>namaProduk = newNamaProduk;
                current->harga =
newHarga;
                return true;
            }
```

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

    void insertAfter(string
namaProduk, int harga, string
afterNamaProduk) {
        Node* newNode = new Node;
        newNode->namaProduk =
namaProduk;
        newNode->harga = harga;
        Node* current = head;
        while (current != nullptr)
{
            if (current-
>namaProduk == afterNamaProduk) {
                newNode->prev =
current;
                newNode->next =
current->next;
                if (current->next
!= nullptr) {
                    current->next-
>prev = newNode;
                } else {
                    tail =
newNode;
                }
                current->next =
newNode;
                return;
            }
```

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

    void deleteAfter(string
afterNamaProduk) {
        Node* current = head;
        while (current != nullptr)
{
            if (current-
>namaProduk == afterNamaProduk) {
                Node* temp =
current->next;
                if (temp !=
nullptr) {
                    current->next
= temp->next;
                    if (temp->next
!= nullptr) {
                        temp-
>next->prev = current;
                    } else {
                        tail =
current;
                    }
                    delete temp;
                }
                return;
            }
            current = current-
>next;
        }
    }
```

```
    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;
        cout << "Nama
Produk\tHarga" << endl;
        while (current != nullptr)
{
            cout << current-
>namaProduk << "\t\t" << current-
>harga << endl;
            current = current-
>next;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList list;
    while (true) {
```

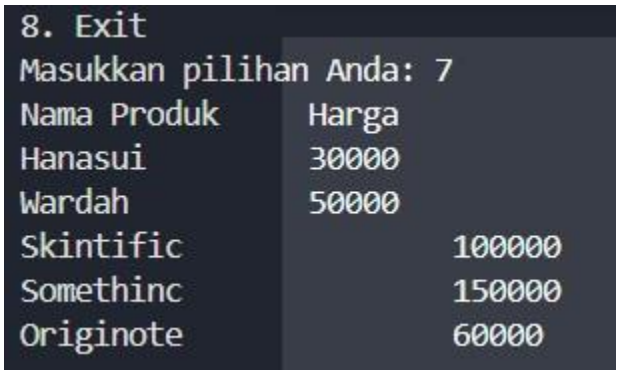
```
        cout << "Toko Skincare
Purwokerto" << endl;
        cout << "1. Tambah Data"
<< endl;
        cout << "2. Hapus Data" <<
endl;
        cout << "3. Update Data"
<< endl;
        cout << "4. Tambah Data
Urutan Tertentu" << endl;
        cout << "5. Hapus Data
Urutan Tertentu" << endl;
        cout << "6. Hapus Seluruh
Data" << endl;
        cout << "7. Tampilkan
Data" << endl;
        cout << "8. Exit" << endl;
        int choice;
        cout << "Masukkan pilihan
Anda: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                string namaProduk;
                int harga;
                cout << "Masukkan
nama produk: ";
                cin >> namaProduk;
                cout << "Masukkan
harga: ";
                cin >> harga;

list.push(namaProduk, harga);
                break;
```

```
            }
            case 2: {
                list.pop();
                break;
            }
            case 3: {
                string
oldNamaProduk, newNamaProduk;
                int newHarga;
                cout << "Masukkan
nama produk yang lama: ";
                cin >>
oldNamaProduk;
                cout << "Masukkan
nama produk yang baru: ";
                cin >>
newNamaProduk;
                cout << "Masukkan
harga yang baru: ";
                cin >> newHarga;
                bool updated =
list.update(oldNamaProduk,
newNamaProduk, newHarga);
                if (!updated) {
                    cout << "Data
tidak ditemukan" << endl;
                }
                break;
            }
            case 4: {
                string namaProduk,
afterNamaProduk;
                int harga;
```

```
                cout << "Masukkan
nama produk: ";
                cin >> namaProduk;
                cout << "Masukkan
harga: ";
                cin >> harga;
                cout << "Masukkan
nama produk setelahnya: ";
                cin >>
afterNamaProduk;

list.insertAfter(namaProduk,
harga, afterNamaProduk);
                break;
            }
            case 5: {
                string
afterNamaProduk;
                cout << "Masukkan
nama produk setelahnya: ";
                cin >>
afterNamaProduk;

list.deleteAfter(afterNamaProduk);
                break;
            }
            case 6: {
                list.deleteAll();
                break;
            }
            case 7: {
                list.display();
                break;
            }
```

```
            case 8: {
                return 0;
            }
            default: {
                cout << "Pilihan
tidak valid" << endl;
                break;
            }
        }
    }
    return 0;
}
```

SCREENSHOOT PROGRAM



2.1. Tampilan sebelum data diupdate

2.2.Tampilan setelah update data

DESKRIPSI PROGRAM

Dalam program ini, kita telah membuat sebuah program untuk mengelola data produk skincare di sebuah toko. Program ini menggunakan struktur data Doubly Linked List untuk menyimpan dan mengatur data produk. Program ini memungkinkan pengguna untuk menambah, menghapus, mengupdate, dan menampilkan data produk skincare. Dengan menggunakan struktur data Doubly

Linked List, kita dapat dengan mudah melakukan operasi pada data produk dengan efisien.

BAB IV
KESIMPULAN

Setelah melakukan pembelajaran mengenai tipe data di Bahasa Pemrograman C++ berikut poin utama yang telah dipelajari :

1. Single Linked List : Struktur data berupa kumpulan node yang terhubung secara sekuensial dengan menggunakan pointer.
2. Single Linked List : Setiap node memiliki dua bagian: bagian isi (data) dan bagian pointer yang menunjuk ke node berikutnya.

3. Double Linked List : Representasi struktur data Double Linked List mencakup pointer HEAD yang menunjuk pada node pertama dan pointer TAIL yang menunjuk pada node terakhir.
4. Double Linked List: Linked list dikatakan kosong jika nilai pointer head adalah NULL, dan nilai pointer prev dari HEAD serta nilai pointer next dari TAIL selalu NULL.

DAFTAR PUSTAKA

Trivusi. (2022, 16 September). Belajar C++ #9: Menggunakan Array untuk Menyimpan Banyak Data, Diakses pada 29 Maret 2024 dari https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html