LAPORAN PRAKTIKUM

MODUL III
SINGLE AND DOUBLE LINKED LIST

Disusun oleh:
Andika Indra Prastawa
NIM: 2311102033

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

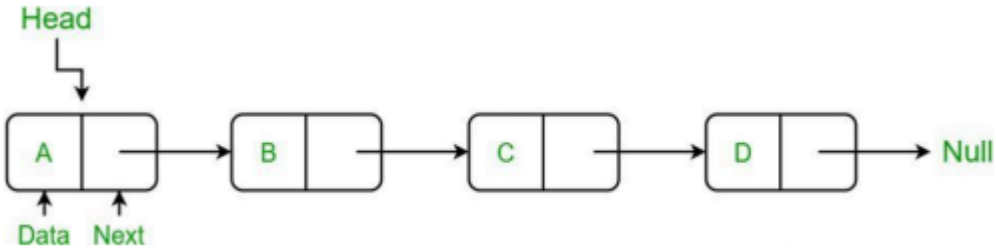
1. Mahasiswa memahami perbedaan konsep Single dan Double Linked List

2.Mahasiswa mampu menerapkan Single dan Double Linked List ke dalam pemrograman

BAB II
DASAR TEORI

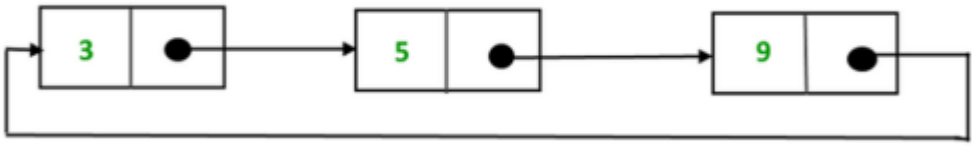
a) Single Linked List

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.



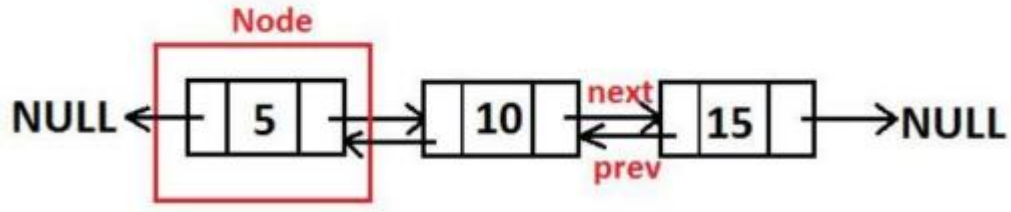
b) Double Linked List

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.

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

Source code

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    // komponen/member
    int data;
    string kata;
    Node *next;
};
Node *head;
Node *tail;
// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}
// Pengecekan
bool isEmpty()
{
    if (head == NULL)
        return true;
    else
        return false;
}
// Tambah Depan
void insertDepan(int nilai, string kata)
```

```
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->kata = kata;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}
// Tambah Belakang
void insertBelakang(int nilai, string kata)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->kata = kata;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        tail->next = baru;
        tail = baru;
```

```
    }
}
// Hitung Jumlah List
int hitungList()
{
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}
// Tambah Tengah
void insertTengah(int data, string kata, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;
        baru->kata = kata;
        // tranversing
        bantu = head;
```

```
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
// Hapus Depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}
// Hapus Belakang
void hapusBelakang()
```

```
{
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false)
    {
        if (head != tail)
        {
            hapus = tail;
            bantu = head;
            while (bantu->next != tail)
            {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}
// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *hapus, *bantu, *bantu2;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
```

```
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        int nomor = 1;
        bantu = head;
        while (nomor <= posisi)
        {
            if (nomor == posisi - 1)
            {
                bantu2 = bantu;
            }
            if (nomor == posisi)
            {
                hapus = bantu;
            }
            bantu = bantu->next;
            nomor++;
        }
        bantu2->next = bantu;
        delete hapus;
    }
}
// Ubah Depan
void ubahDepan(int data, string kata)
{
    if (isEmpty() == false)
    {
        head->data = data;
        head->kata = kata;
    }
```

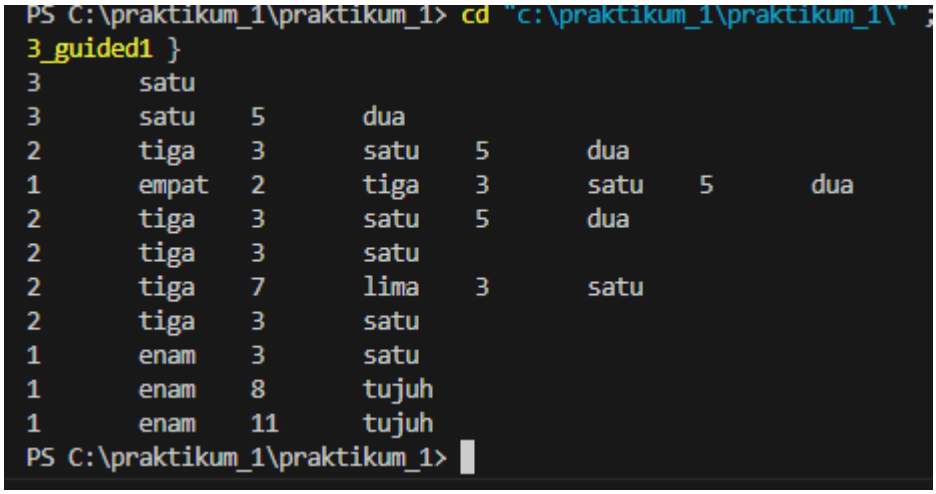
```
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Tengah
void ubahTengah(int data, string kata, int posisi)
{
    Node *bantu;
    if (isEmpty() == false)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {
            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
            bantu->kata;
        }
    }
    else
    {
```

```
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Belakang
void ubahBelakang(int data, string kata)
{
    if (isEmpty() == false)
    {
        tail->data = data;
        tail->kata = kata;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Hapus List
void clearList()
{
    Node *bantu, *hapus;
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}
// Tampilkan List
void tampil()
{
    Node *bantu;
```

```
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << "\t";
            cout << bantu->kata << "\t";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
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 ini memiliki fungsi dasar seperti menambahkan node di depan, di belakang, dan di tengah, menghapus node di depan, di belakang dan di tengah, serta mengubah nilai data pada node. Setiap node dalam linked list memiliki dua data yaitu integer dan string, dan program ini memberikan kemampuan untuk menampilkan seluruh isi linked list serta menghitung jumlah elemen dalam linked list. Program juga memberikan penanganan ketika linked list kosong atau ketika posisi yang diminta berada di luar jangkauan. Pada fungsi utama, terdapat contoh penggunaan dari setiap fungsi yang telah didefinisikan.

2. Guided 2

Source code

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

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

class DoublyLinkedList {
public:
    Node* head;
    Node* tail;
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }

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

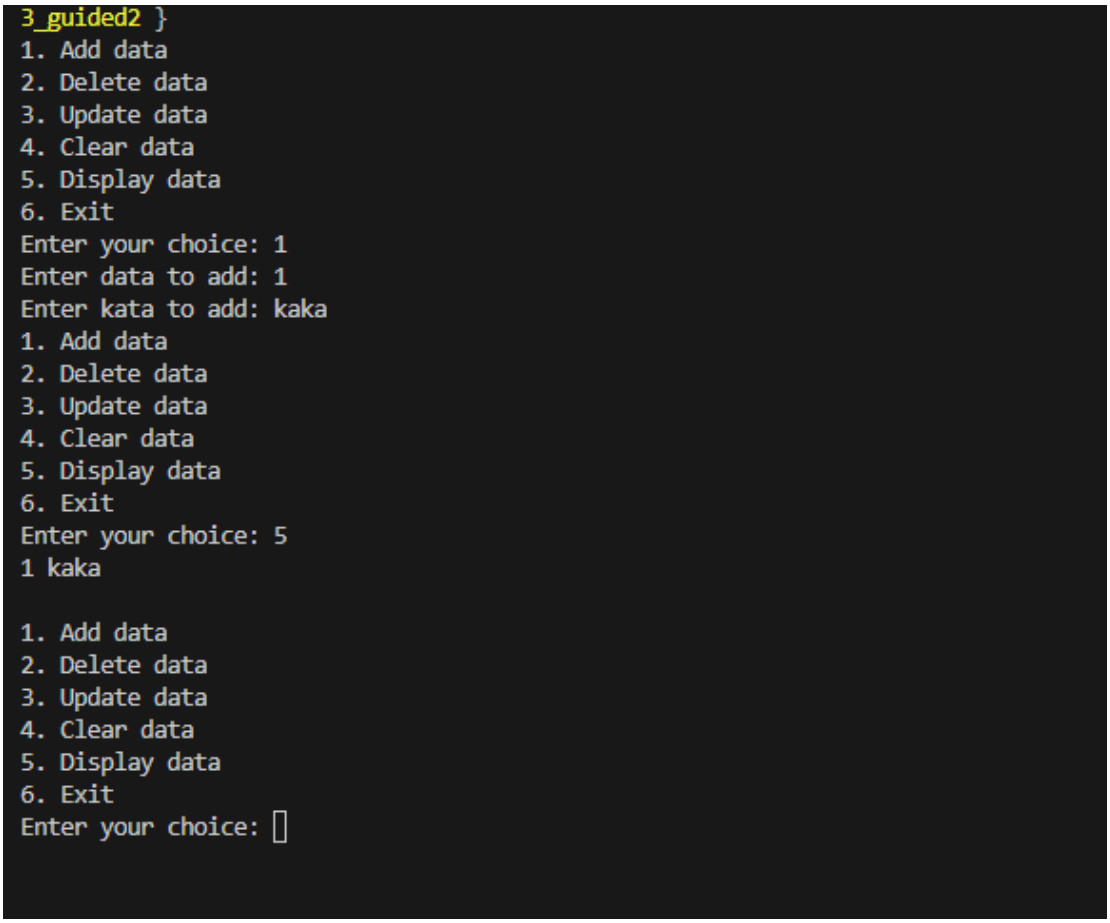
```
    }

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

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

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

```
        delete temp;
        }
        head = nullptr;
        tail = nullptr;
    }

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

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;
}
```

Screenshoot program



Deskripsi program

Awalnya, kita menentukan kelas Node yang menggambarkan node dalam Doubly Linked List. Setiap node memiliki properti data dan kata untuk menyimpan informasi dan teks, bersama dengan pointer prev dan next untuk menghubungkannya dengan node sebelumnya dan berikutnya. Kemudian, kita mendefinisikan kelas DoublyLinkedList yang mewakili Doubly Linked List itu sendiri. Kelas ini memiliki properti head dan tail yang menunjuk pada node pertama dan terakhir dalam Doubly Linked List.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

```
//Andika Indra Prastawa_2311102033_IF-11-A
#include <iostream>
using namespace std;

struct Node_2311102033 {
    string nama;
    int usia;
    Node_2311102033* next;
};

class LinkedList {
private:
    Node_2311102033* head;

public:
    LinkedList() {
        head = nullptr;
    }

    void insertAwal(string nama, int usia) {
        Node_2311102033* newNode = new Node_2311102033;
        newNode->nama = nama;
        newNode->usia = usia;
        newNode->next = head;
        head = newNode;
    }

    void insertAkhir(string nama, int usia) {
        Node_2311102033* newNode = new Node_2311102033;
```

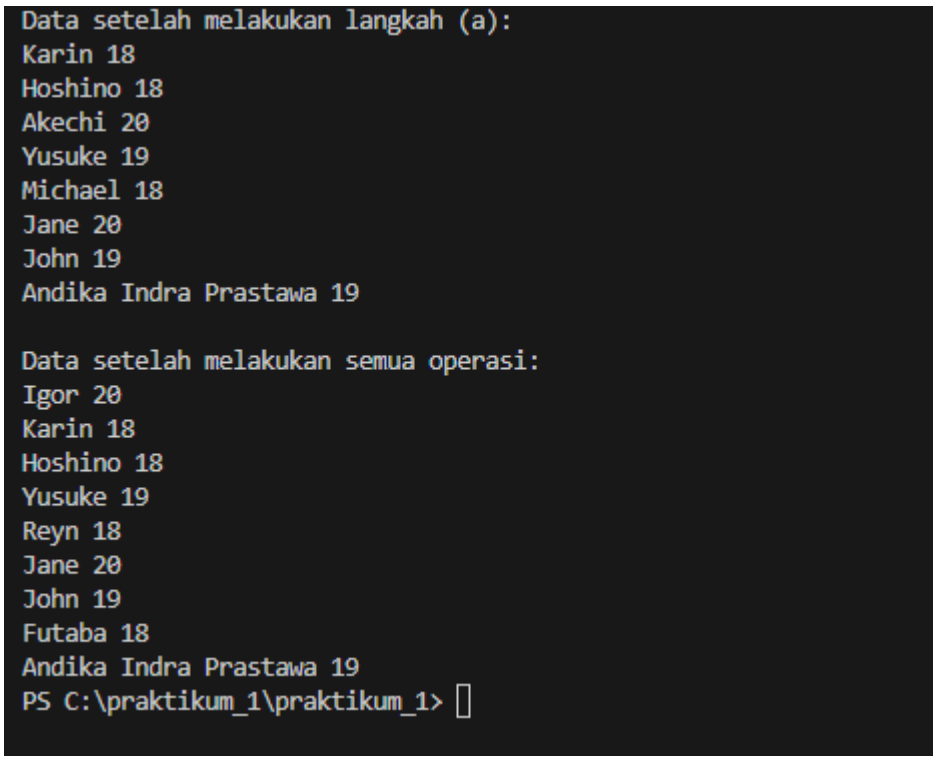
```
        newNode->nama = nama;
        newNode->usia = usia;
        newNode->next = nullptr;

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

        Node_2311102033* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }

    void insertSetelah(string nama, int usia, string
namaSebelum) {
        Node_2311102033* newNode = new Node_2311102033;
        newNode->nama = nama;
        newNode->usia = usia;

        Node_2311102033* 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_2311102033* temp = head;
            head = head->next;
            delete temp;
            return;
        }

        Node_2311102033* prev = head;
        Node_2311102033* 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_2311102033* 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_2311102033* temp = head;
        while (temp != nullptr) {
            cout << temp->nama << " " << temp->usia << endl;
            temp = temp->next;
        }
    }
};

int main() {
    LinkedList myList;

    myList.insertAwal("Andika Indra Prastawa", 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 melakukan langkah (a):" << endl;
    myList.tampilkan();
    cout << endl;

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

    return 0;
}
```

Screenshoot program



Deskripsi program

Program ini menggunakan struktur data linked list untuk menyimpan daftar nama beserta usia. Kelas LinkedList mengelola linked list dengan menyediakan fungsi-fungsi untuk memasukkan elemen di awal, di akhir, dan setelah suatu elemen tertentu, menghapus elemen, serta mengubah nilai elemen. Pada fungsi utama (main()), program melakukan serangkaian operasi seperti memasukkan beberapa nama beserta usianya ke dalam linked list, menghapus satu nama dari linked list, memasukkan nama baru setelah suatu nama tertentu, memasukkan nama baru di awal, dan mengubah usia seseorang. Setelah itu, program menampilkan isi linked list setelah semua operasi dilakukan.

2. Unguided 2

Source code

```
#include <iostream>
#include <string>
using namespace std;
struct Node
{
    string nama;
    int harga;
    Node *prev;
    Node *next;
};
```

```
class DoubleLinkedList
{
private:
    Node *head;
    Node *tail;
    int size;

public:
    DoubleLinkedList()
    {
        head = NULL;
        tail = NULL;
        size = 0;
    }
    void addData(string nama, int harga)
    {
        Node *node = new Node;
        node->nama = nama;
        node->harga = harga;
        node->prev = tail;
        node->next = NULL;
        if (head == NULL)
        {
            head = node;
            tail = node;
        }
        else
        {
            tail->next = node;
            tail = node;
        }
        size++;
    }
    void addDataAt(int index, string nama, int harga)
```

```
    {
        if (index < 0 || index > size)
        {
            cout << "Index tidak di temukan" << endl;
            return;
        }
        Node *node = new Node;
        node->nama = nama;
        node->harga = harga;
        if (index == 0)
        {
            node->prev = NULL;
            node->next = head;
            head->prev = node;
            head = node;
        }
        else if (index == size)
        {
            node->prev = tail;
            node->next = NULL;
            tail->next = node;
            tail = node;
        }
        else
        {
            Node *current = head;
            for (int i = 0; i < index - 1; i++)
            {
                current = current->next;
            }
            node->prev = current;
            node->next = current->next;
            current->next->prev = node;
            current->next = node;
```

```
        }
        size++;
    }
    void deleteDataAt(int index)
    {
        if (index < 0 || index >= size)
        {
            cout << "Index out of bounds" << endl;
            return;
        }

        if (index == 0)
        {
            Node *temp = head;
            head = head->next;
            head->prev = NULL;
            delete temp;
        }
        else if (index == size - 1)
        {
            Node *temp = tail;
            tail = tail->prev;
            tail->next = NULL;
            delete temp;
        }
        else
        {
            Node *current = head;
            for (int i = 0; i < index; i++)
            {
                current = current->next;
            }
            current->prev->next = current->next;
            current->next->prev = current->prev;
```

```
            delete current;
        }
        size--;
    }
    void clearData()
    {
        while (head != NULL)
        {
            Node *temp = head;
            head = head->next;
            delete temp;
        }
        tail = NULL;
        size = 0;
    }
    void displayData()
    {
        cout << "Nama Produk\tHarga" << endl;
        Node *current = head;
        while (current != NULL)
        {
            cout << current->nama << "\t\t" << current->harga <<
endl;
            current = current->next;
        }
    }
    void updateDataAt(int index, string nama, int harga)
    {
        if (index < 0 || index >= size)
        {
            cout << "Index out of bounds" << endl;
            return;
        }
        Node *current = head;
```

```
        for (int i = 0; i < index; i++)
        {
            current = current->next;
        }
        current->nama = nama;
        current->harga = harga;
    }
};
int main()
{
    DoubleLinkedList dll;
    int choice;
    string nama;
    int harga;
    int index;
    do
    {
       cout << "Toko Skincare Purwokerto" << endl;
        cout << "1. Tambah Data" << endl;
        cout << "2. Hapus Data" << endl;
        cout << "3. Update Data" << endl;
        cout << "4. Tambah Data pada Urutan Tertentu" << endl;
        cout << "5. Hapus Data pada Urutan Tertentu" << endl;
        cout << "6. Hapus Semua Data" << endl;
        cout << "7. Tampilkan Data" << endl;
        cout << "8. Keluar " << endl;
        cout << "Pilih: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            cout << "Nama Produk: ";
            cin >> nama;
            cout << "Harga: ";
```

```
            cin >> harga;
            dll.addData(nama, harga);
            break;
        case 2:
            cout << "Index: ";
            cin >> index;
            dll.deleteDataAt(index);
            break;
        case 3:
            cout << "Index: ";
            cin >> index;
            cout << "Nama Produk: ";
            cin >> nama;
            cout << "Harga: ";
            cin >> harga;
            dll.updateDataAt(index, nama, harga);
            break;
        case 4:
            cout << "Index: ";
            cin >> index;
            cout << "Nama Produk: ";
            cin >> nama;
            cout << "Harga: ";
            cin >> harga;
            dll.addDataAt(index, nama, harga);
            break;
        case 5:
            cout << "Index: ";
            cin >> index;
            dll.deleteDataAt(index);
            break;
        case 6:
            dll.clearData();
            break;
```

```
        case 7:
            dll.displayData();
            break;
        case 8:
            break;
        default:
            cout << "Pilihan tidak valid" << endl;
            break;
        }
        cout << endl;
    } while (choice != 8);
    return 0;
}
```

Screenshoot program

1) Menambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific

```
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 4
Index: 2
Nama Produk: Azarine
Harga: 65000
```

Tampilan data

```
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7
Nama Produk     Harga
Originote               60000
Somethinc               150000
Azarine         65000
Skintific               100000
Wardah          50000
Hanasui         30000
```

2) Hapus produk wardah

```
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 5
Index: 4
```

Tampilan data

```
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7
Nama Produk     Harga
Originote               60000
Somethinc               150000
Azarine         65000
Skintific               100000
Hanasui         30000
```

3) Update produk Hanasui menjadi Cleora dengan harga 55.000

```
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 3
Index: 4
Nama Produk: Cleora
Harga: 55000
```

Tampilan data

```
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7
Nama Produk     Harga
Originote               60000
Somethinc               150000
Azarine         65000
Skintific               100000
Cleora          55000
```

4) Tampilkan menu seperti dibawah ini

```
Toko Skincare Purwokerto
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih:
```

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

```
Pilih: 7
Nama Produk     Harga
Originote               60000
Somethinc               150000
Azarine         65000
Skintific               100000
Cleora          55000
```

Deskripsi program

Program ini menggunakan struktur data double linked list untuk menyimpan informasi tentang nama produk dan harganya. Program ini memungkinkan pengguna untuk menambahkan, menghapus, mengubah, serta menampilkan data produk. Pengguna dapat memilih operasi yang ingin dilakukan melalui menu interaktif yang disediakan. Setiap produk disimpan dalam bentuk node pada double linked list, yang memungkinkan pengguna untuk menambahkan atau menghapus produk di awal, di akhir, atau pada posisi tertentu dalam daftar. Program berjalan dalam loop hingga pengguna memilih opsi untuk keluar dari program.

BAB IV
KESIMPULAN

Setelah menjalani pembelajaran tentang konsep tipe data dalam C++. Saya memahami beberapa poin penting yang perlu dicatat. Pertama, dalam Single Linked List, kami belajar bahwa struktur data ini terdiri dari sejumlah node yang saling terhubung secara berurutan melalui penggunaan pointer. Kedua, kami menyadari bahwa setiap node dalam Single Linked List memiliki dua komponen utama: bagian data yang menyimpan informasi, dan bagian pointer yang menunjukkan ke node berikutnya dalam urutan. Sementara itu, dalam Double Linked List, kami mempelajari representasi struktur data yang lebih kompleks, yang melibatkan pointer HEAD yang merujuk pada node pertama dan pointer TAIL yang menunjuk pada node terakhir. Akhirnya, kami memahami bahwa Double Linked List dianggap kosong ketika nilai pointer head memiliki nilai NULL, dan baik nilai pointer prev dari HEAD maupun nilai pointer next dari TAIL selalu memiliki nilai NULL secara konsekuen.