# LAPORAN PRAKTIKUM
# STRUKTUR DATA DAN ALGORITMA
# MODUL 3
# “SINGLE AND DOUBLE LINKED LIST”


**DISUSUN OLEH:**
MOHAMMAD HARITS. T
2311102016
S1 IF-11-A

DOSEN PENGAMPU:

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

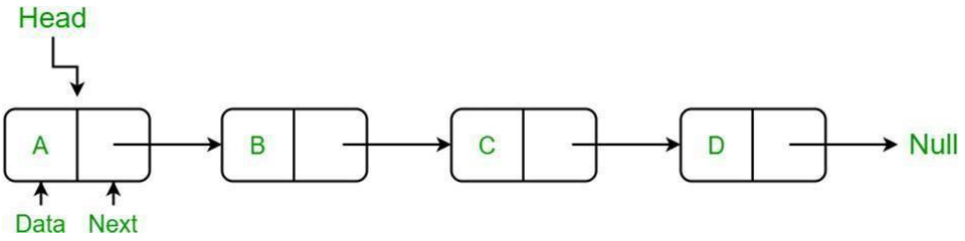
PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO

2024

## A. 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.



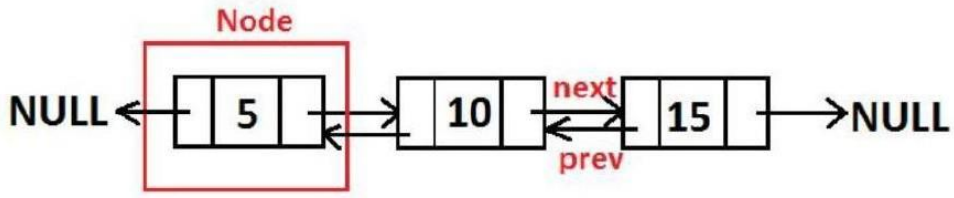

Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List umumnya 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 SingleLinked 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.

## B. GUIDED

### Guided 1 : Latihan Single Linked List

```
#include <iostream>
using namespace std;
struct Node{
    int data;
    Node *next;
};
Node  *head;
Node  *tail;
void init(){
    head = NULL;
    tail = NULL;
}
bool isEmpty(){
    if (head == NULL)
    return  true;
    else
    return false;
}
void insertDepan(int nilai){
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true){
        head = tail = baru;
        tail->next = NULL;
    }
    else{
        baru->next = head;
        head = baru;
    }
}
void insertBelakang(int nilai){
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true){
        head = tail = baru;
        tail->next = NULL;
    }
    else{
    tail->next = baru;
    tail = baru;
```

```
    }
}
int hitungList(){
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while( hitung != NULL ){
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}
void insertTengah(int data, 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;
        // tranversing
            bantu = head;
            int nomor = 1;
        while( nomor < posisi - 1 ){
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
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;
    }
}
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;
    }
}
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;
            }
```

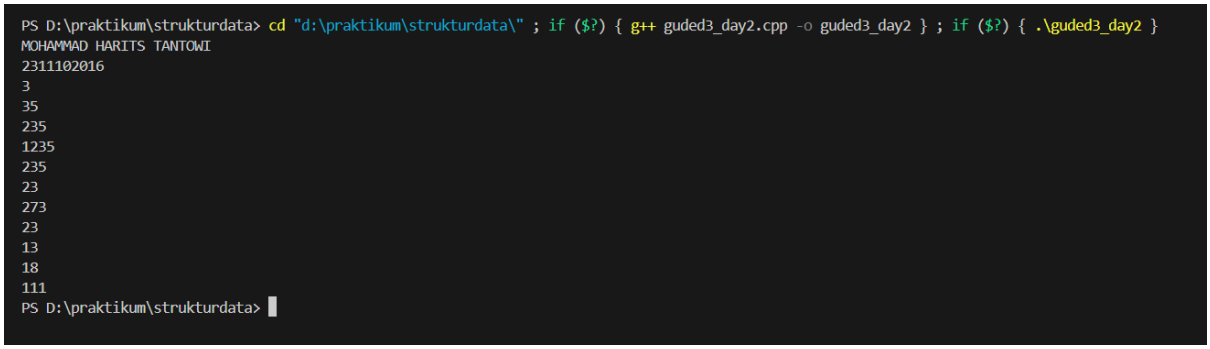
```
            bantu = bantu->next;
            nomor++;
        }
        bantu2->next = bantu;
    delete hapus;
    }
}
void ubahDepan(int data){
    if (isEmpty() == false){
        head->data = data;
    }
    else{
        cout << "List masih kosong!" << endl;
    }
}
void ubahTengah(int data, 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;
        }
    }
    else{
        cout << "List masih kosong!" << endl;
    }
}
void ubahBelakang(int data){
    if (isEmpty() == false){
        tail->data = data;
    }
    else{
        cout << "List masih kosong!" << endl;
    }
}
```

```
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;
}
void tampil(){
    Node *bantu;
    bantu = head;
    if (isEmpty() == false){
        while (bantu != NULL){
            cout << bantu->data << ends;
            bantu = bantu->next;
        }
        cout << endl;
    }
    else{
        cout << "List masih kosong!" << endl;
    }
}
int main(){
    cout << "MOHAMMAD HARITS TANTOWI" << endl;
    cout << "2311102016" << endl;

    init();
    insertDepan(3);tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7,2);
    tampil();
    hapusTengah(2);
    tampil();
```

```
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();
return 0;
}
```

Screenshots Output




Deskripsi :
Program yang diberikan merupakan implementasi dari struktur data linked list dalam bahasa pemrograman C++. Linked list adalah struktur data yang terdiri dari serangkaian simpul yang terhubung satu sama lain melalui penggunaan pointer. Dalam program ini, digunakan single linked list non-circular, yang mana setiap simpul hanya memiliki satu pointer yang menunjuk ke simpul berikutnya tanpa adanya simpul yangkembali menunjuk ke simpul sebelumnya.

Program ini memiliki beberapa fungsi utama, seperti `insertDepan()`, `insertBelakang()`, dan `insertTengah()` untuk menambahkan simpul pada berbagai posisi dalam linked list. Fungsi-fungsi `hapusDepan()`, `hapusBelakang()`, dan
`hapusTengah()` digunakan untuk menghapus simpul dari linked list. Selain itu, terdapat juga fungsi-fungsi `ubahDepan()`, `ubahBelakang()`, dan `ubahTengah()` untuk mengubah data pada simpul tertentu. Fungsi lain seperti `clearList()` digunakan untuk menghapus semua simpul dalam linked list, sedangkan `tampil()` digunakan untuk menampilkan isi dari linked list.

Dalam program ini, fungsi-fungsi tersebut diuji dengan memanggilnya secara berurutan dan menampilkan isi linked list setiap kali dilakukan operasi tambah, hapus,atau ubah data. Dengan demikian, program ini memberikan implementasi sederhana dari struktur data linked list dalam bahasa C++ yang dapat digunakan untuk berbagai keperluan pengolahan data.

**Guided 2 : Latihan Double Linked List**

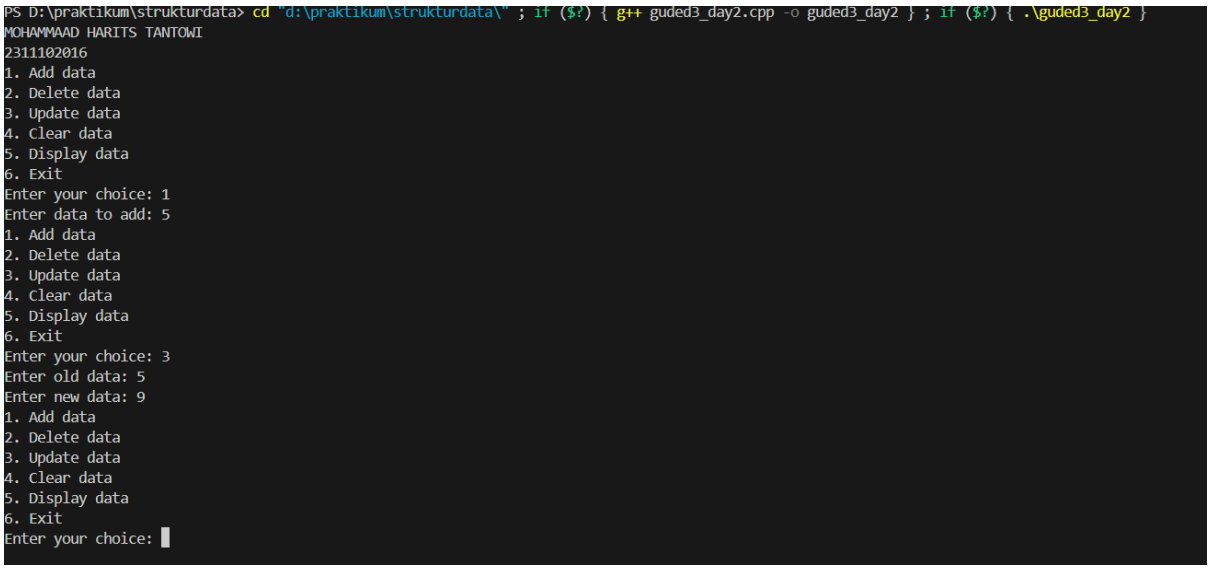
```
#include <iostream>
using namespace std;
class Node {
    public:int data;
    Node* prev;
    Node* next;
};
class DoublyLinkedList {
    public:
    Node* head;
    Node* tail;
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }
    void push(int data) {
        Node* 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() {
    cout << "MOHAMMAD HARITS TANTOWI" << endl;
    cout << "2311102016" << endl;

    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 ini merupakan implementasi Double Linked List sederhana dalam bahasa C++. Double Linked List adalah struktur data yang terdiri dari simpul-simpul yang saling terhubung, dimana setiap simpul memiliki data serta dua pointer, yaitu pointer ke simpul sebelumnya dan pointer ke simpul selanjutnya.

Kelas Node digunakan untuk merepresentasikan setiap simpul dalam Double Linked List. Setiap simpul memiliki data integer dan dua pointer, yaitu prev dan next, yang menunjuk ke simpul sebelumnya dan simpul selanjutnya.

Kelas DoubleLinkedList menyediakan fungsi-fungsi dasar untuk manipulasi Double Linked List, seperti menambahkan data di awal (push), menghapus data dari awal (pop), mengupdate data, menghapus semua data, dan menampilkan isi Double Linked List.

Fungsi main() merupakan titik awal program, dimana terdapat sebuah loop interaktif yang memungkinkan pengguna memilih operasi yang ingin dilakukan pada Double Linked List, seperti menambahkan data baru, menghapus data, mengupdate data, menghapus semua data, menampilkan data, dan keluar dari program. Program akan terus berjalan hingga pengguna memilih untuk keluar.

## C. UNGUIDED

### 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>
#include <string>

using namespace std;

struct Node {
    string nama;
    int usia;
    Node* next;
};
Node* head = nullptr;
void init() {
    head = nullptr;
}

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

void insertTengah(string nama, int usia, int posisi) {
    if (posisi <= 1 || head == nullptr) {
        insertDepan(nama, usia);
```

```
        return;
    }

    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;

    Node* current = head;
    for (int i = 1; i < posisi - 1 && current->next != nullptr;
i++) {
        current = current->next;
    }

    newNode->next = current->next;
    current->next = newNode;
}

void insertBelakang(string nama, int usia) {
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    newNode->next = nullptr;

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

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

    current->next = newNode;
}

void hapusData(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* current = head;
    while (current->next != nullptr && current->next->nama !=
nama) {
        current = current->next;
    }

    if (current->next == nullptr) {
        cout << "Data tidak ditemukan" << endl;
        return;
    }

    Node* temp = current->next;
    current->next = current->next->next;
    delete temp;
}

void ubahData(string nama, string newNama, int newUsia) {
    Node* current = head;
    while (current != nullptr) {
        if (current->nama == nama) {
            current->nama = newNama;
            current->usia = newUsia;
            return;
        }
        current = current->next;
    }
    cout << "Data tidak ditemukan" << endl;
}

void tampilkanData() {
    Node* current = head;
    if (current == nullptr) {
        cout << "Linked list kosong" << endl;
        return;
    }
    cout << current->nama << "\t" << current->usia << endl;
    while (current->next != nullptr) {
        current = current->next;
        cout << current->nama << "\t" << current->usia << endl;
    }
}
```

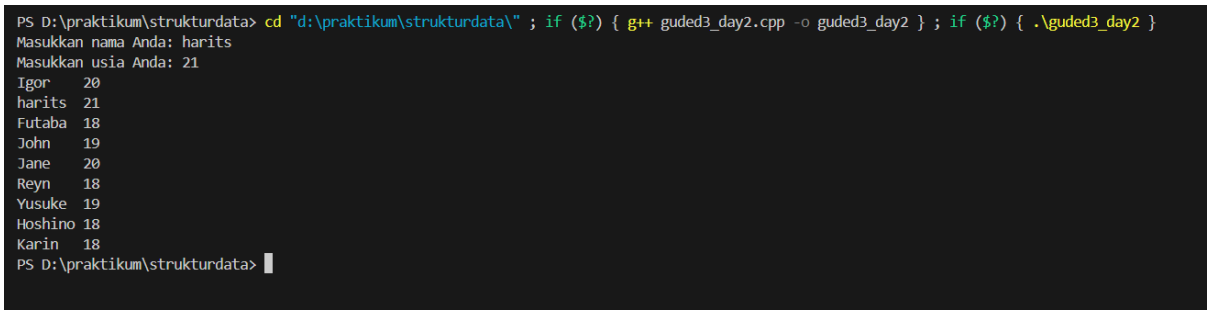
```cpp
int main() {
    init();

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

    insertDepan(namaAnda, usiaAnda);
    insertBelakang("John", 19);
    insertBelakang("Jane", 20);
    insertTengah("Futaba", 18, 2); // di antara John dan Jane
    insertBelakang("Michael", 18);
    insertBelakang("Yusuke", 19);
    insertBelakang("Akechi", 20);
    insertBelakang("Hoshino", 18);
    insertBelakang("Karin", 18);
    hapusData("Akechi");
    insertDepan("Igor", 20);
    ubahData("Michael", "Reyn", 18);
    tampilkanData();

    return 0;
}
```

Screenshots output



Deskripsi :

Program yang diberikan adalah implementasi sederhana dari struktur datalinked list dalam bahasa pemrograman C++. Linked list adalah struktur data linear di mana setiap elemen, yang disebut node, terdiri dari dua bagian: data itu sendiri dan alamat atau referensi ke node berikutnya dalam urutan. Dalam program ini, setiap node merepresentasikan informasi tentang seorang individu, termasuk nama dan usia mereka. Pada awal program, linked list kosong dideklarasikan dengan menggunakan pointer yang menunjuk ke elemen pertama dari linked list, yang disebut head. Selanjutnya, program menyediakan sejumlah fungsi untuk melakukan operasi dasar pada linked list. Fungsi-fungsi tersebut mencakup penambahan node baru di depan, di tengah, atau di belakanglinked list dengan menggunakan fungsi `insertDepan`, `insertTengah`, dan `insertBelakang` secara berturut-turut. Terdapat juga fungsi untuk menghapus node (`hapusData`) berdasarkan nama yang diberikan, mengubah informasi pada node (`ubahData`) berdasarkan nama, dan menampilkan seluruh data dalam linked list (`tampilkanData`). Di dalam fungsi `main()`, terdapat penggunaan fungsi-fungsi tersebut untuk melakukan manipulasi pada linked list, seperti menambahkan, menghapus, mengubah, dan menampilkan data. Keseluruhan operasi pada linked list dilakukan dengan memanipulasi alamat atau referensi antar node menggunakan pointer, sehingga memungkinkan fleksibilitas dalam penambahan, penghapusan, atau perubahan urutan data dalam linked list tanpa harus memindahkan seluruh elemen, yang merupakan kelebihan utama dari struktur data ini.

**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>
#include <string>

using namespace std;

struct Node {
    string namaProduk;
    int harga;
    Node* prev;
    Node* next;
};

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

```
void init() {
    head = nullptr;
    tail = nullptr;
}

void tambahData(string namaProduk, int harga) {
    Node* newNode = new Node;
    newNode->namaProduk = namaProduk;
    newNode->harga = harga;
    newNode->prev = nullptr;
    newNode->next = nullptr;

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

void hapusData(string namaProduk) {
    if (head == nullptr) {
        cout << "Linked list kosong" << endl;
        return;
    }

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

```
            cout << "Data " << namaProduk << " berhasil dihapus"
<< endl;
            return;
        }


        current = current->next;
    }
    cout << "Data tidak ditemukan" << endl;
}

void updateData(string namaProduk, string newNamaProduk, int
newHarga) {
    Node* current = head;
    while (current != nullptr) {
        if (current->namaProduk == namaProduk) {
            current->namaProduk = newNamaProduk;
            current->harga = newHarga;
            cout << "Data berhasil diupdate" << endl;
            return;
        }
        current = current->next;
    }
    cout << "Data tidak ditemukan" << endl;
}

void tambahDataUrutanTertentu(string namaProduk, int harga, int
posisi) {
    if (posisi <= 1 || head == nullptr) {
        tambahData(namaProduk, harga);
        return;
    }

    Node* newNode = new Node;
    newNode->namaProduk = namaProduk;
    newNode->harga = harga;

    Node* current = head;
    for (int i = 1; i < posisi - 1 && current->next != nullptr;
i++) {
        current = current->next;
    }

    newNode->next = current->next;
    newNode->prev = current;
    if (current->next != nullptr) {
        current->next->prev = newNode;
```

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

void hapusDataUrutanTertentu(int posisi) {
    if (head == nullptr) {
        cout << "Linked list kosong" << endl;
        return;
    }

    if (posisi == 1) {
        Node* temp = head;
        head = head->next;
        if (head != nullptr) {
            head->prev = nullptr;
        }
        delete temp;
        cout << "Data pada posisi 1 berhasil dihapus" << endl;
        return;
    }

    Node* current = head;
    for (int i = 1; i < posisi && current != nullptr; i++) {
        current = current->next;
    }

    if (current == nullptr) {
        cout << "Posisi tidak valid" << endl;
        return;
    }

    if (current == tail) {
        tail = tail->prev;
        tail->next = nullptr;
    } else {
        current->prev->next = current->next;
        current->next->prev = current->prev;
    }
    delete current;
    cout << "Data pada posisi " << posisi << " berhasil dihapus"
<< endl;
}

void hapusSeluruhData() {
    while (head != nullptr) {
```

```
        Node* temp = head;
        head = head->next;
        delete temp;
    }
    tail = nullptr;
    cout << "Seluruh data berhasil dihapus" << endl;
}

void tampilkanData() {
    if (head == nullptr) {
        cout << "Linked list kosong" << endl;
        return;
    }

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


int main() {
    cout << "MOHAMMAD HARITS TANTOWI" << endl;
    cout << "2311102016" << endl;

    init();

    tambahData("Originote", 60000);
    tambahData("Somethinc", 150000);
    tambahData("Skintific", 100000);
    tambahData("Wardah", 50000);
    tambahData("Hanasui", 30000);

    int pilihan;
    do {
        cout << "\nToko 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 << "Pilihan Anda: ";
        cin >> pilihan;
        cin.ignore();

        switch (pilihan) {
            case 1: {
                string namaProduk;
                int harga;
                cout << "Masukkan nama produk: ";
                getline(cin, namaProduk);
                cout << "Masukkan harga produk: ";
                cin >> harga;
                tambahData(namaProduk, harga);
                break;
            }
            case 2: {
                string namaProduk;
                cout << "Masukkan nama produk yang ingin dihapus:
";
                getline(cin, namaProduk);
                hapusData(namaProduk);
                break;
            }
            case 3: {
                string namaProduk, newNamaProduk;
                int newHarga;
                cout << "Masukkan nama produk yang ingin
diupdate: ";
                getline(cin, namaProduk);
                cout << "Masukkan nama baru produk: ";
                getline(cin, newNamaProduk);
                cout << "Masukkan harga baru produk: ";
                cin >> newHarga;
                updateData(namaProduk, newNamaProduk, newHarga);
                break;
            }
            case 4: {
                string namaProduk;
                int harga, posisi;
                cout << "Masukkannama produk yang ingin
ditambahkan: ";
                getline(cin, namaProduk);
                cout << "Masukkan harga produk: ";
                cin >> harga;
```

```
                cout << "Masukkan posisi penambahan data: ";
                cin >> posisi;
                tambahDataUrutanTertentu(namaProduk, harga,
posisi);
                break;
            }
            case 5: {
                int posisi;
                cout << "Masukkan posisi data yang ingin dihapus:
";
                cin >> posisi;
                hapusDataUrutanTertentu(posisi);
                break;
            }
            case 6: {
                hapusSeluruhData();
                break;
            }
            case 7: {
                tampilkanData();
                break;
            }
            case 8: {
                cout << "Terima kasih telah menggunakan layanan
kami" << endl;
                break;
            }
            default:
                cout << "Pilihan tidak valid" << endl;
                break;
        }
    } while (pilihan != 8);

    return 0;
}
```

SCREENSHOOT OUTPUT

a. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific

```
PS D:\praktikum\strukturdata> cd "d:\praktikum\strukturdata\" ; if ($?) { g++ guded3_day2.cpp -o guded3_day2 } ; if ($?) { .\guded3_day2 }
MOHAMMAD HARITS TANTOWI
2311102016

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
Pilihan Anda: 1
Masukkan nama produk: Azarine
Masukkan harga produk: 65000

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
Pilihan Anda: 7
Nama Produk     Harga
Originote       60000
Somethinc       150000
Skintific       100000
Wardah  50000
Hanasui 30000
Azarine 65000
```

b. Hapus produk wardah

```
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
Pilihan Anda: 5
Masukkan posisi data yang ingin dihapus:4
Data pada posisi 4 berhasil dihapus

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
Pilihan Anda: 7
Nama Produk     Harga
Originote       60000
Somethinc       150000
Skintific       100000
Hanasui 30000
Azarine 65000
```

c. Update produk Hanasui menjadi Cleora dengan harga 55000 dan tampilkan seluruh data

```
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
Pilihan Anda: 1
Masukkan nama produk: Cleora
Masukkan harga produk: 55000

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
Pilihan Anda: 7
Nama Produk     Harga
Originote       60000
Somethinc       150000
Skintific       100000
Hanasui 30000
Azarine 65000
Cleora  55000
```

Deskripsi :
Program yang diberikan merupakan implementasi sederhana dari struktur data double linked list dalam bahasa pemrograman C++. Double linked list memungkinkan pengguna untuk menyimpan dan mengelola data dalam urutan linear dengan kemampuan untuk bergerak maju dan mundur di dalam list. Setiap elemen data disebut node, yang memiliki dua pointer: satu yang menunjuk ke node sebelumnya (prev) dan satu lagi yang menunjuk ke node berikutnya (next).

Dalam program ini, setiap node memiliki informasi tentang produk, seperti nama produk dan harganya, serta dua pointer ke node sebelumnya dan node berikutnya. Fungsi-fungsi utama dalam program, seperti tambahData, hapusData, updateData, tambahDataUrutanTertentu,

hapusDataUrutanTertentu, hapusSeluruhData, dan tampilkanData, memungkinkan pengguna untuk menambah, menghapus, memperbarui, dan menampilkan data dalam linked list.

Operasi-operasi ini memanfaatkan sifat dari double linked list dengan memperhatikan dan memperbarui kedua pointer prev dan next untuk menjaga konsistensi struktur linked list. Dengan demikian, program ini menyediakan cara yang efisien untuk mengelola daftar produk dengan menggunakan struktur double linked list.

# DAFTAR PUSTAKA

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts, problems, Interview Questions*. CareerMonk Publications.

TylerMSFT. (n.d.). Collections (C++/CX). diakses dari https://learn.microsoft.com/en-us/cpp/cppcx/collections-c-cx?view=msvc-170