

**LAPORAN PRAKTIKUM STRUKTUR  
DATA DAN ALGORITMA**

**MODUL 3  
SINGLE AND DOUBLE LINKED LIST**



**Disusun Oleh :**  
MARSHELY AYU ISWANTO  
2311102073

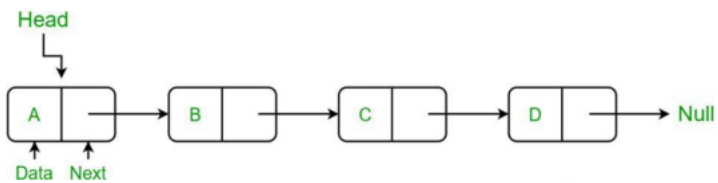
**Dosen**  
Wahyu Andi Saputra, S.Pd., M.Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

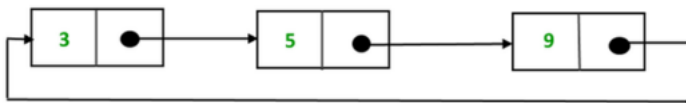
## A. Dasar Teori

### a) Single Linked List

Single linked list merupakan struktur data yang terdiri dari elemen data yang disusun secara berurutan. Pada setiap elemen dalam linked list disebut dengan node dan mempunyai dua bagian utama yaitu data yang menyimpan satu nilai serta informasi, dan juga pointer yang menyimpan alamat dari node selanjutnya yang biasanya diberi nama variabel next. Setiap node memiliki satu pointer. Setiap node memiliki pointer yang menunjukkan pada node berikutnya dalam urutan. Node terakhir dalam linked list menunjuk ke null yaitu menandakan bahwa akhir dari linked list. Elemen pada awal suatu list disebut head dan elemen terakhir dari suatu list disebut tail.



Pada single linked list dapat membuat linked list kosong, penambahan elemen, penghapusan elemen, dan pencarian elemen. 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. Single linked list juga sering digunakan pada implementasi berbagai struktur data seperti stack, queue, dan hash table.

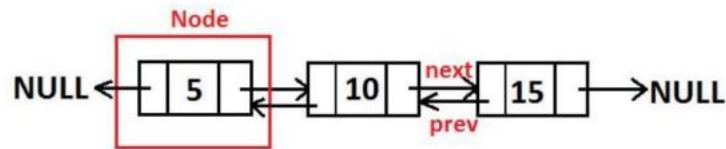


### b) Double Linked List

Double linked list adalah struktur data yang sama mirip dengan linked list, namun setiap node nya dalam double linked list mempunyai dua pointer. Dua pointer tersebut yaitu pointer yang menunjuk pada node sebelumnya (prev) dan pointer yang menunjuk ke node berikutnya (next). 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 menggunakan double linked list adalah karena memungkinkan langkah traversal dengan mudah dan tepat. Dalam Hal ini berguna dalam beberapa aplikasi seperti aplikasi yang memerlukan operasi penyisipan atau penghapusan

di depan dan belakang linked list dengan cepat. Terdapat kekurangan juga pada double linked list yaitu membutuhkan lebih banyak memori untuk menyimpan pointer tambahan untuk setiap node dan operasi pada double linked list memerlukan penanganan yang kompleksitas karena perlu mengatur dua pointer untuk setiap operasi.



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 : Single Linked List

#### Source Code

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

// Deklarasi Struct Node
struct Node {
    int data;
    Node* next;
};

Node* head;
Node* tail;

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

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

// Tambah Node di depan
void insertDepan(int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        baru->next = head;
        head = baru;
    }
}

// Tambah Node di belakang
void insertBelakang(int nilai) {
    Node* baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty()) {
        head = tail = baru;
    } else {
        tail->next = baru;
        tail = baru;
    }
}
```

```

    }
}

// Hitung jumlah Node di list
int hitungList() {
    Node* hitung = head;
    int jumlah = 0;
    while (hitung != NULL) {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

// Tambah Node di posisi tengah
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 = new Node();
        baru->data = data;
        Node* bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1) {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Hapus Node di depan
void hapusDepan() {
    if (!isEmpty()) {
        Node* hapus = head;
        if (head->next != NULL) {
            head = head->next;
            delete hapus;
        } else {
            head = tail = NULL;
            delete hapus;
        }
    } else {
        cout << "List kosong!" << endl;
    }
}

// Hapus Node di belakang

```

```

void hapusBelakang() {
    if (!isEmpty()) {
        if (head != tail) {
            Node* hapus = tail;
            Node* 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 Node di posisi tengah
void hapusTengah(int posisi) {
    if (posisi < 1 || posisi > hitungList()) {
        cout << "Posisi diluar jangkauan" << endl;
    } else if (posisi == 1) {
        cout << "Posisi bukan posisi tengah" << endl;
    } else {
        Node* hapus;
        Node* bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++) {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

// Ubah data Node di depan
void ubahDepan(int data) {
    if (!isEmpty()) {
        head->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah data Node di posisi tengah
void ubahTengah(int data, int posisi) {
    if (!isEmpty()) {
        if (posisi < 1 || posisi > hitungList()) {
            cout << "Posisi di luar jangkauan" << endl;

```

```

        } else if (posisi == 1) {
            cout << "Posisi bukan posisi tengah" << endl;
        } else {
            Node* bantu = head;
            for (int nomor = 1; nomor < posisi; nomor++) {
                bantu = bantu->next;
            }
            bantu->data = data;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah data Node di belakang
void ubahBelakang(int data) {
    if (!isEmpty()) {
        tail->data = data;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus semua Node di list
void clearList() {
    Node* bantu = head;
    while (bantu != NULL) {
        Node* hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

// Tampilkan semua data Node di list
void tampil() {
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != NULL) {
            cout << bantu->data << " ";
            bantu = bantu->next;
        }
        cout << endl;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

int main() {
    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

```

{ g++ unguided1_27mar2024.cpp -o unguided1_27mar2024 } ; if ($?) { .\ungui
ded1_27mar2024 }
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS E:\TugasAlproSmt2_6Mar2024>

```

MARSHELY AYU ISWANTO  
2311102073

Ln 2, Col 11 | 31 characters | 100% | Window | UTF-8

### Deskripsi Program:

Program tersebut merupakan program c++ yaitu pengimplementasian dari linked list. Di dalam program tersebut ada penyisipan (insertion), penghapusan (deletion) dan pembaruan (update). Program ini menggunakan Struct 'node' yaitu untuk menampilkan sebuah simpul dalam linked list. Output dari program tersebut adalah hasil dari setiap operasi yang dilakukan oleh linked list sesuai dengan urutan yang ditentukan oleh fungsi main (). Pada operasi tersebut, operasi pertama yaitu penyisipam node dengan nilai 3 di depan, linked list mempunyai satu node dengan nilai 3 didepan. Kemudian setelah operasi kedua (Penyisipan node dengan nilai 5 dibelakang). Linked list memiliki dua node dengan nilai 3 dan 5, dan seterusnya. Pada program tersebut juga terdapat penghapusan node pertama dan penghapusan node terakhir. Pembaruan pada node pertama, terakhir, dan pembaruan node pada posisi ke-2

### Guided 2 : Double Linked List

#### Source Code



```

#include <iostream>
using namespace std;

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

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

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

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

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

        head = newNode;
    }

    void pop() {
        if (head == nullptr) {
            return;
        }
        Node* temp = head;
        head = head->next;

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

        delete temp;
    }
}

```

```

bool update(int oldData, int newData) {
    Node* current = head;

    while (current != nullptr) {
        if (current->data == oldData) {
            current->data = newData;
            return true;
        }
        current = current->next;
    }
    return false;
}

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

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

};

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

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

        switch (choice) {
            case 1: {
                int data;

```

```

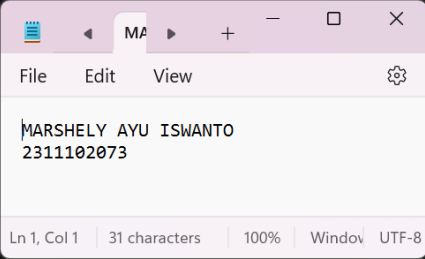
        cout << "Enter data to add: ";
        cin >> data;
        list.push(data);
        break;
    }
    case 2: {n
        list.pop();
        break;
    }
    case 3: {
        int oldData, newData;
        cout << "Enter old data: ";
        cin >> oldData;
        cout << "Enter new data: ";
        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;
}

```

Screenshots Output

```
PS E:\TugasAlproSmt2_6Mar2024> cd "e:\TugasAlproSmt2_6Mar2024\" ; if ($?) { g++ unguided2_27mar2024.cpp -o unguided2_27mar2024 } ; if ($?) { .\unguided2_27mar2024 }
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 5
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 7
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
7 5
```



## Deskripsi Program

Program tersebut merupakan program c++ yaitu pengimplementasian dari double linked list. Double linked list merupakan struktur data yang pada setiap node nya memiliki dua pointer (satu menunjuk ke node sebelumnya (prev) dan satu menunjuk pada node selanjutnya (next)). Pada program tersebut melakukan operasi-operasi pada kelas double linked list. Program tersebut memberikan pilihan kepada pengguna untuk dapat memilih operasi yang diinginkan yaitu ada

1. menambah 'push (int data)' menambah node baru ke depan linked list
2. menghapus 'pop()' menghapus node pertama dari linked list
3. memperbarui 'update (int oldData, int newData)' mencari node yang sama dengan oldData, lalu menggantinya dengan newData
4. menghapus semua 'deleteAll()' semua node akan dihapus dari linked list
5. menampilkan 'display ()' menampilkan isi linked list
6. ingin keluar dari program

Output dari program tersebut adalah sesuai hasil operasi yang dipilih. Contoh output diatas yaitu pengguna menambahkan dua data yaitu 7 dan 5

## 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>

using namespace std;

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

// Deklarasi global pointer head
Node* head = nullptr;

// Fungsi untuk menyisipkan 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;
}

// Fungsi untuk menyisipkan 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 (head == nullptr) {
            head = newNode;
        } else {
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }

    // Fungsi untuk menyisipkan data di tengah linked list
    void insertTengah(string nama, int usia, string
nama_sebelum, string nama_setelah) {
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->usia = usia;
        Node* temp = head;
        while (temp != nullptr && temp->nama != nama_sebelum)
        {
            temp = temp->next;
        }
        if (temp == nullptr) {
            cout << "Data " << nama_sebelum << " tidak
ditemukan" << endl;
            return;
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }

    // Fungsi untuk menghapus data dari linked list
    void hapusData(string nama) {
        if (head == nullptr) {
            cout << "Linked list kosong" << endl;
            return;
        }
        Node* temp = head;
        Node* prev = nullptr;
        while (temp != nullptr && temp->nama != nama) {
            prev = temp;
            temp = temp->next;
        }
        if (temp == nullptr) {
            cout << "Data tidak ditemukan" << endl;
            return;
        }
        if (prev == nullptr) {
            head = head->next;
        } else {
            prev->next = temp->next;
        }
    }

```

```

    }
    delete temp;
}

// Fungsi untuk mengubah data pada linked list
void ubahData(string nama_lama, string nama_baru, int
usia_baru) {
    Node* temp = head;
    while (temp != nullptr && temp->nama != nama_lama) {
        temp = temp->next;
    }
    if (temp == nullptr) {
        cout << "Data tidak ditemukan" << endl;
        return;
    }
    temp->nama = nama_baru;
    temp->usia = usia_baru;
}

// Fungsi untuk menampilkan seluruh data pada linked list
void tampilkanData() {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->nama << " " << temp->usia << endl;
        temp = temp->next;
    }
}

int main() {
    // Menambahkan data pertama (nama dan usia Anda)
    string nama_anda;
    int usia_anda;
    cout << "Masukkan nama Anda: ";
    cin >> nama_anda;
    cout << "Masukkan usia Anda: ";
    cin >> usia_anda;
    insertDepan(nama_anda, usia_anda);
    insertBelakang("John", 19);
    insertBelakang("Jane", 20);
    insertBelakang("Michael", 18);
    insertBelakang("Yusuke", 19);
    insertBelakang("Akechi", 20);
    insertBelakang("Hoshino", 18);
    insertBelakang("Karin", 18);
    cout << endl;

    // Menampilkan data awal
    cout << "Data awal:" << endl;
    tampilkanData();

    // Menghapus data Akechi

```

```

hapusData("Akechi");

// Menambahkan data lain sesuai urutan yang diminta
insertTengah("Futaba", 18, "John", "Jane");
insertDepan("Igor", 20);

// Menampilkan seluruh data
cout << "Data setelah operasi tambah dan dihapus:" << endl;
tampilkanData();

// Mengubah data Michael menjadi Reyn dengan usia 18
ubahData("Michael", "Reyn", 18);

// Menampilkan seluruh data setelah operasi hapus dan
ubah
cout << "\nData setelah operasi diubah :" << endl;
tampilkanData();

return 0;
}

```

## Screenshots Output

The screenshot shows a Windows terminal window with a dark background. The command prompt is at the top, showing the directory path and the execution of a C++ program. The program prompts for a name and an age, then displays the initial data list. It then performs operations (insertion and deletion) and displays the updated data list. A Notepad window is overlaid on the terminal, showing the user's input: 'MARSHELY AYU ISWANTO' and '2311102073'.

```

PS E:\TugasAlproSmt2_6Mar2024> cd "e:\TugasAlproSmt2_6Mar2024\"; if ($?) { g++ unguided1_27mar2024.cpp -o unguided1_27mar2024 }; if ($?) { .\unguided1_27mar2024 }
Masukkan nama Anda: Shely
Masukkan usia Anda: 19

Data awal:
Shely 19
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18
Data setelah operasi tambah dan dihapus:
Igor 20
Shely 19
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18

Data setelah operasi diubah :
Igor 20
Shely 19
John 19
Futaba 18
Jane 20
Reyn 18
Yusuke 19
Hoshino 18
Karin 18
PS E:\TugasAlproSmt2_6Mar2024>

```

Notepad window content:

```

MARSHELY AYU ISWANTO
2311102073

```

Notepad window status bar: Ln 2, Col 11 | 31 characters | 100% | Window | UTF-8



### Deskripsi Program :

Program tersebut merupakan program c++ yaitu menggunakan linked list untuk menyimpan data dan memanipulasi data pada urutan tertentu. Program tersebut menyimpan nama mahasiswa dan usia mahasiswa. Program ini dimulai dengan mendefinisikan struct 'node' yang memiliki tiga anggota yaitu 'nama' (string), 'usia' (int), dan 'next' (pointer ke node berikutnya). Deklarasi variable pointer global 'head' menunjuk pada node pertama dalam linked list. Output dari kode program ini adalah menampilkan nama dan usia mahasiswa sesuai dengan perintah operasi yang diminta.

### 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. 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 :

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

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

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

//menambahkan 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;
    else
        tail = newNode;
    head = newNode;
}

//menambahkan 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;
    newNode->prev = tail;
    if (tail != nullptr)
        tail->next = newNode;
    else
        head = newNode;
    tail = newNode;
}

//menambahkan data di urutan tertentu
void insertTengah(string nama_produk, int harga,
string nama_sebelum, string nama_setelah) {
    Node* newNode = new Node;
```

```

        newNode->nama_produk = nama_produk;
        newNode->harga = harga;
        Node* temp = head;
        while (temp != nullptr && temp->nama_produk !=
nama_sebelum) {
            temp = temp->next;
        }
        if (temp == nullptr) {
            cout << "Data " << nama_sebelum << " tidak
ditemukan" << endl;
            return;
        }
        if (temp->next == nullptr) {
            insertBelakang(nama_produk, harga);
        } else {
            newNode->next = temp->next;
            newNode->prev = temp;
            temp->next->prev = newNode;
            temp->next = newNode;
        }
    }
}

//menghapus data dari linked list
void hapusData(string nama_produk) {
    if (head == nullptr) {
        cout << "Linked list kosong" << endl;
        return;
    }
    Node* temp = head;
    while (temp != nullptr && temp->nama_produk !=
nama_produk) {
        temp = temp->next;
    }
    if (temp == nullptr) {
        cout << "Data tidak ditemukan" << endl;
        return;
    }
    if (temp == head) {
        head = head->next;
        if (head != nullptr)
            head->prev = nullptr;
        else
            tail = nullptr;
    } else if (temp == tail) {
        tail = tail->prev;
        tail->next = nullptr;
    } else {
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
    }
    delete temp;
}

```

```

}

//mengubah data pada linked list
void ubahData(string nama_produk_lama, string
nama_produk_baru, int harga_baru) {
    Node* temp = head;
    while (temp != nullptr && temp->nama_produk !=
nama_produk_lama) {
        temp = temp->next;
    }
    if (temp == nullptr) {
        cout << "Data tidak ditemukan" << endl;
        return;
    }
    temp->nama_produk = nama_produk_baru;
    temp->harga = harga_baru;
}

//menampilkan seluruh data pada linked list
void printtampilkanData(){
    if ( head == NULL){
        cout << "Buat linked list terlebih dahulu" <<
endl;
    }else{
        cout << "Data Produk : " << endl;
        Node* cur = head;
        cout << "
_____ " << endl;
        cout << "| Nama Produk \t\t| Harga Produk\t|"
<< endl;
        cout << "
_____ " << endl;
        while ( cur -> next != head){
            // print
            cout << "| " << cur -> nama_produk << "\t\t|"
" << cur -> harga << endl;
            //step
            cur = cur -> next;
        }
    }
}

//menampilkan data awal
void tampilkanDataAwal() {
    cout << "
_____ " <<
endl;
    cout << "|Nama Produk   |\t\t\tHarga   |" << endl;
    cout << "
_____ " <<
endl;
    cout << "|Originote       |\t\t\t60.000 |" << endl;
    cout << "|Somethinc         |\t\t\t150.000|" << endl;
}

```

```

        cout << "|Skintific      |\t\t100.000|" << endl;
        cout << "|Wardah        |\t\t50.000 |" << endl;
        cout << "|Hanasui         |\t\t30.000 |" << endl;
        cout << " _____" << endl;
    }

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

        // Menampilkan data awal
        tampilkanDataAwal();

        // Menu program
        int menu;
        string nama_produk, nama_sebelum, nama_setelah,
nama_produk_baru;
        int harga, harga_baru;

        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 << "Pilih menu: ";
            cin >> menu;

            switch (menu) {
                case 1:
                    cout << "Masukkan nama produk: ";
                    cin >> nama_produk;
                    cout << "Masukkan harga: ";
                    cin >> harga;
                    insertBelakang(nama_produk, harga);
                    break;
                case 2:
                    cout << "Masukkan nama produk yang
akan dihapus: ";
                    cin >> nama_produk;

```

```

        hapusData(nama_produk);
        break;
    case 3:
        cout << "Masukkan nama produk yang
akan diupdate: ";
        cin >> nama_produk;
        cout << "Masukkan nama produk baru:
";
        cin >> nama_produk_baru;
        cout << "Masukkan harga baru: ";
        cin >> harga_baru;
        ubahData(nama_produk,
nama_produk_baru, harga_baru);
        break;
    case 4:
        cout << "Masukkan nama produk baru:
";
        cin >> nama_produk;
        cout << "Masukkan harga: ";
        cin >> harga;
        cout << "Masukkan nama produk
sebelumnya: ";
        cin >> nama_sebelum;
        cout << "Masukkan nama produk
setelahnya: ";
        cin >> nama_setelah;
        insertTengah(nama_produk, harga,
nama_sebelum, nama_setelah);
        break;
    case 5:
        cout << "Masukkan nama produk yang
akan dihapus: ";
        cin >> nama_produk;
        hapusData(nama_produk);
        break;
    case 6:
        while (head != nullptr) {
            hapusData(head->nama_produk);
        }
        break;
    case 7:
        printtampilkanData();
        break;

    case 8:
        cout << "Program selesai." << endl;
        break;
    default:
        cout << "Menu tidak valid." << endl;
}
} while (menu != 8);

```

```
        return 0;
    }
```

## Screenshost Output

```
PS E:\TugasAlproSmt2_6Mar2024> cd "e:\TugasAlproSmt2_6Mar2024\" ; if ($?) { g++ cobaaaa.cpp -o cobaaaa } ; if ($?) { .\cobaaaa }
```

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

Toko Skincare Purwokerto

1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit

Pilih menu: 4

Masukkan nama produk baru: Azarine

Masukkan harga: 65000

Masukkan nama produk sebelumnya: Somethinc

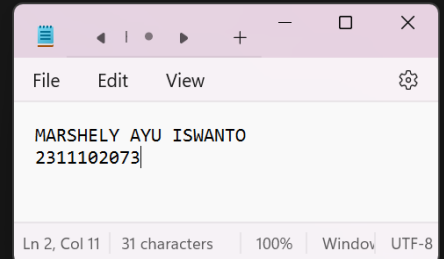
Masukkan nama produk setelahnya: Skintific

Toko Skincare Purwokerto

1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan Data
8. Exit

Pilih menu: 2

Masukkan nama produk yang akan dihapus: 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

Pilih menu: 3

Masukkan nama produk yang akan diupdate: Hanasui

Masukkan nama produk baru: Cleora

Masukkan harga baru: 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

Pilih menu: 7

Data Produk :

Nama Produk	Harga Produk
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Cleora	55000

```
PS E:\TugasAlproSmt2_6Mar2024>
```

### Deskripsi Program:

Program tersebut merupakan program c++ yaitu pengimplementasian dari struktur linked list. Program tersebut menyimpan daftar produk skincare beserta harganya. Mengelola daftar produk skincare yaitu seperti penambahan produk, menghapus produk, menambah produk pada urutan tertentu, menghapus produk pada urutan tertentu, menghapus seluruh data, atau menampilkan seluruh data. Struktur data 'node' dalam program tersebut memiliki 4 anggota yaitu 'nama produk' bertipe string, 'harga' bertipe 'int', 'prev' bertipe pointer 'node' untuk menunjukan ke node sebelum linked list, lalu ada 'next' yang bertipe pointer 'node' untuk menunjukan ke node berikutnya dalam linked list. Variabel head dan tail digunakan untuk menunjuk node pertama dan terakhir dalam linked list, awalnya diatur sebagai 'nullptr'. Output dari kode program ini adalah pengguna bisa memilih menu sesuai dengan kebutuhan untuk operasi data.

### Kesimpulan

Singkatnya, perbandingan antara single linked list dan double linked list tergantung pada kebutuhan dan karakteristik. Single linked list lebih efisien dalam penggunaan memori, namun Double linked list lebih memudahkan untuk mengakses nilai sebelumnya dan selanjutnya.

### D. Referensi

Laden. (2022, April 29). *Linked List #2: Single Linked List*. From <https://dryladen.netlify.app>: <https://dryladen.netlify.app/linkedlist-singlelinked/>

Pkthapa. (2023, Juli 13). *Program to implement Singly Linked List in C++ using class*. From <https://www.geeksforgeeks.org>: <https://www.geeksforgeeks.org/program-to-implement-singly-linked-list-in-c-using-class/>

Thakur, A. (2023, Oktober 5). *C++ Program to Implement Doubly Linked List*. From <https://www.tutorialspoint.com>: <https://www.tutorialspoint.com/cplusplus-program-to-implement-doubly-linked-list>