

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL III
SINGLE AND DOUBLE LINKED LIST**



Disusun Oleh :

**NAMA : WISNU RANANTA RADITYA
PUTRA
NIM : 2311102013**

Dosen :

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

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

BAB I

TUJUAN PRAKTIKUM

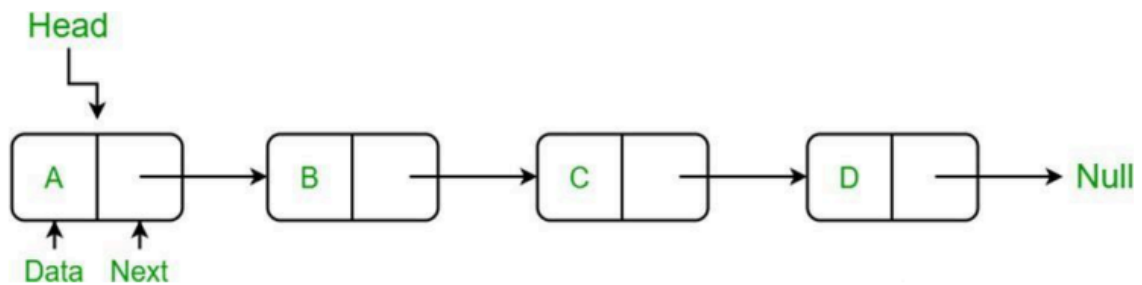
1. Mahasiswa memahami perbedaan konsep Single dan Double Linked List
2. Mahasiswa mampu menerapkan Single dan Double Linked List ke dalam pemrograman

BAB II

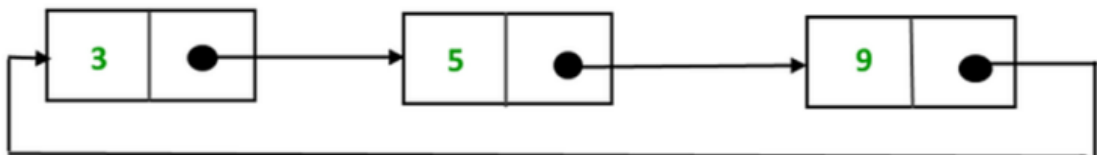
DASAR TEORI

1) Single Linked List

Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Setiap elemen dalam linked list dihubungkan ke elemen lain melalui pointer. Masing-masing komponen sering disebut dengan simpul atau node atau verteks. Pointer adalah alamat elemen. Setiap simpul pada dasarnya dibagi atas dua bagian pertama disebut bagian isi atau informasi atau data yang berisi nilai yang disimpan oleh simpul. Bagian kedua disebut bagian pointer yang berisi alamat dari node berikutnya atau sebelumnya. Dengan menggunakan struktur seperti ini, linked list dibentuk dengan cara menunjuk pointer next suatu elemen ke elemen yang mengikutinya. Pointer next pada elemen terakhir merupakan NULL, yang menunjukkan akhir dari suatu list. Elemen pada awal suatu list disebut head dan elemen terakhir dari suatu list disebut tail.



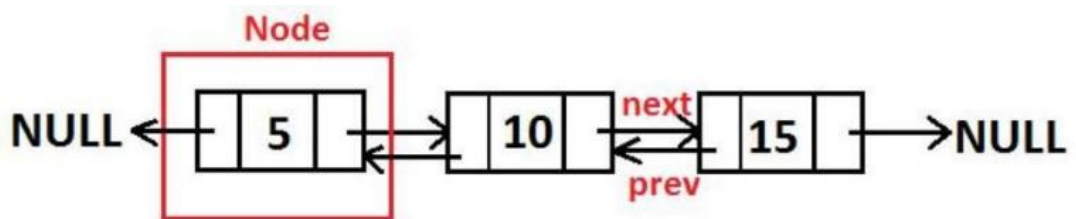
Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List umumnya lebih efisien dalam penggunaan memori dibandingkan dengan jenis Linked List lainnya, seperti Double Linked List dan Circular Linked List. Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.



2) Array Dua Dimensi

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

Guided 1

```
#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;
}

```

Screenshots Output

```

3      satu
3      satu      5      dua
2      tiga      3      satu      5      dua
1      empat      2      tiga      3      satu      5      dua
2      tiga      3      satu      5      dua
2      tiga      3      satu
2      tiga      7      lima      3      satu
2      tiga      3      satu
1      enam      3      satu
1      enam      8      tujuh
1      enam      11     tujuh
PS C:\Semester 2\Praktikum Strutur Data\Modul_3>

```

Deskripsi:

Program diatas merupakan program implementasi dari sigle linked list. Program ini dapat menambahkan node baru didepan dan belakang linked list, menghitung jumlah dalam linked list, menambahkan node baru di tengah, menghapus node pertama,tengah dan terakhir, menampilkan semua isi linked list. Program ini menggunakan pointer head dan tail sebagai penanda awal dan akhir dari linked list. Pointer ini digunakan untuk menghubungkan setiap node.

Guided 2

```
#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;
}
```


Screenshots Output

Add data:

```
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: 1
Enter kata to add: radit
```

Display data:

```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
3 wisnu
2 putra
1 radit
```

Delete data:

```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 2
```

Update data:

```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 1
Enter new data: 1
Enter new kata: raditya
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
2 putra
1 raditya
```

Exit:

```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 6
PS C:\Semester 2\Praktikum Strutur Data\Modul_3>
```

Deskripsi

Program diatas merupakan program double linked list. Program ini memiliki 2 class utama yaitu Node dan DoublyLinkedList. Program ini memiliki 2 pointer pada class node yaitu prev dan next yang menunjuk ke node sebelumnya dan setelahnya. Program ini juga memiliki 2 pointer pada class DoublyLinkedList yaitu head dan tail yang menunjuk ke node pertama dan terakhir. Program memungkinkan user untuk menambah, menghapus, memperbarui, menghapus semua data, dan menampilkan data dalam linked list. Pada saat user menginputkan data, program akan memproses pilihan yang dipilih dan akan memanggil fungsi yang sesuai didalam class DoublyLinkedList. Setelah itu, program akan menampilkan output yang diinginkan user.

BAB IV

UNGUIDED

Unguided 1

```
#include <iostream>
#include <string>
using namespace std;
struct Node
{
    string nama;
    int usia;
    Node *next;
};
class LinkedList
{
private:
    Node *head;

public:
    LinkedList()
    {
        head = nullptr;
    }
    void insertDepan(string nama, int usia)
    {
        Node *baru = new Node;
        baru->nama = nama;
        baru->usia = usia;
        baru->next = head;
        head = baru;
    }
    void insertBelakang(string nama, int usia)
    {
        Node *baru = new Node;
        baru->nama = nama;
        baru->usia = usia;
        baru->next = nullptr;
        if (head == nullptr)
        {
            head = baru;
        }
        else
        {
            Node *temp = head;
            while (temp->next != nullptr)
            {
                temp = temp->next;
            }
            temp->next = baru;
        }
    }
}
```

```

    }
    void insertTengah(string nama, int usia, int posisi)
    {
        if (posisi < 1)
        {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        if (posisi == 1)
        {
            insertDepan(nama, usia);
            return;
        }
        Node *baru = new Node;
        baru->nama = nama;
        baru->usia = usia;
        Node *temp = head;
        for (int i = 1; i < posisi - 1 && temp != nullptr;
i++)
        {
            temp = temp->next;
        }
        if (temp == nullptr)
        {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        baru->next = temp->next;
        temp->next = baru;
    }
    void hapus(string nama)
    {
        if (head == nullptr)
        {
            cout << "List kosong!" << endl;
            return;
        }
        if (head->nama == nama)
        {
            Node *temp = head;
            head = head->next;
            delete temp;
            return;
        }
        Node *temp = head;
        while (temp->next != nullptr && temp->next->nama
!= nama)
        {
            temp = temp->next;
        }
        if (temp->next == nullptr)

```

```

        {
            cout << "Data tidak ditemukan" << endl;
            return;
        }
        Node *hapus = temp->next;
        temp->next = temp->next->next;
        delete hapus;
    }
    void ubah(string namaLama, string namaBaru, int
usiaBaru)
    {
        Node *temp = head;
        while (temp != nullptr)
        {
            if (temp->nama == namaLama)
            {
                temp->nama = namaBaru; // Mengubah nama
                temp->usia = usiaBaru; // Mengubah usia
                return;
            }
            temp = temp->next;
        }
        cout << "Data tidak ditemukan" << endl;
    }
    void tampil()
    {
        if (head == nullptr)
        {
            cout << "List kosong!" << endl;
            return;
        }
        Node *temp = head;
        while (temp != nullptr)
        {
            cout << temp->nama << " " << temp->usia <<
endl;
            temp = temp->next;
        }
    }
};
int main()
{
    LinkedList list;
    string namaAnda;
    int usiaAnda;
    cout << "Masukkan nama anda: ";
    getline(cin, namaAnda);
    cout << "Masukkan usia anda: ";
    cin >> usiaAnda;
    cin.ignore();
    list.insertDepan(namaAnda, usiaAnda);
}

```

```

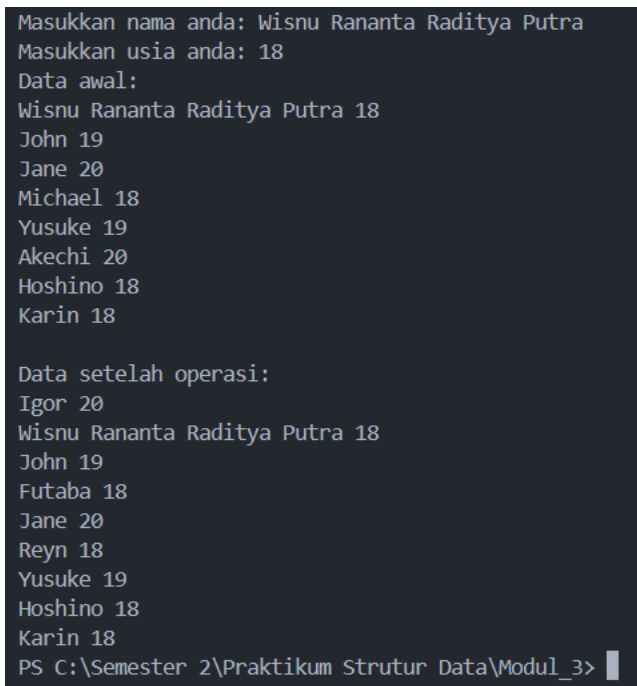
list.insertBelakang("John", 19);
list.insertBelakang("Jane", 20);
list.insertBelakang("Michael", 18);
list.insertBelakang("Yusuke", 19);
list.insertBelakang("Akechi", 20);
list.insertBelakang("Hoshino", 18);
list.insertBelakang("Karin", 18);

cout << "Data awal:" << endl;
list.tampil();

list.hapus("Akechi");
list.insertTengah("Futaba", 18, 3);
list.insertDepan("Igor", 20);
list.ubah("Michael", "Reyn", 18);
cout << "\nData setelah operasi:" << endl;
list.tampil();
return 0;
}

```

Screenshots Output:



```

Masukkan nama anda: Wisnu Rananta Raditya Putra
Masukkan usia anda: 18
Data awal:
Wisnu Rananta Raditya Putra 18
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18

Data setelah operasi:
Igor 20
Wisnu Rananta Raditya Putra 18
John 19
Futaba 18
Jane 20
Reyn 18
Yusuke 19
Hoshino 18
Karin 18
PS C:\Semester 2\Praktikum Strutur Data\Modul_3>

```

Deskripsi:

Program di atas merupakan implementasi dari Single Linked List. Dalam program ini, terdapat sebuah struktur data **Node** yang memiliki dua atribut yaitu **nama** dan **usia**. Kemudian, terdapat class **LinkedList** yang memiliki beberapa metode untuk mengelola linked list, seperti menyisipkan node di depan, di belakang, atau di tengah, menghapus

node, mengubah data pada node, dan menampilkan isi linked list. Lalu program akan meminta user untuk menginputkan nama dan usia mahasiswa yang akan ditambahkan. Kemudian memasukkan data tersebut ke linked list. Selanjutnya, data lainnya dimasukkan ke dalam linked list menggunakan metode **insertBelakang**. Setelah itu, program menampilkan data awal, melakukan beberapa operasi seperti menghapus data Akechi, menyisipkan data Futaba di tengah antara data John dan Jane, menyisipkan di depan data Igor, dan mengubah data Michael menjadi Reyn, lalu menampilkan seluruh data setelah operasi dilakukan.

Unguided 2

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

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

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

```

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

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

void insertAt(string namaProduk, int harga, int
position)
{
    if (position < 1)
    {
        cout << "Invalid position" << endl;
        return;
    }

    Node *newNode = new Node;

```



```

newNode->namaProduk = namaProduk;
newNode->harga = harga;
newNode->prev = nullptr;
newNode->next = nullptr;

if (position == 1)
{
    newNode->next = head;
    head->prev = newNode;
    head = newNode;
}
else
{
    Node *current = head;
    int count = 1;
    while (current != nullptr && count < position
- 1)
    {
        current = current->next;
        count++;
    }

    if (current == nullptr)
    {
        cout << "Invalid position" << endl;
        return;
    }

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

```

```

        {
            current->next->prev = newNode;
        }
        else
        {
            tail = newNode;
        }
        current->next = newNode;
    }
}

```

```

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

```

```

        {
            current->next->prev = current->prev;
        }
        delete current;
        return;
    }
    current = current->next;
}

cout << "Data not found" << endl;
}

bool    update(string    oldNamaProduk,    string
newNamaProduk, int newHarga)
{
    Node *current = head;
    while (current != nullptr)
    {
        if (current->namaProduk == oldNamaProduk)
        {
            current->namaProduk = newNamaProduk;
            current->harga = newHarga;
            return true;
        }
        current = current->next;
    }
    return false;
}

void deleteAll()
{
    Node *current = head;

```

```

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

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

int main()
{
    DoublyLinkedList list;
    list.push("Originote", 60000);
    list.push("Somethinc", 150000);
    list.push("Skintific", 100000);
    list.push("Wardah", 50000);

```

```
list.push("Hanasui", 30000);

while (true)
{
    cout << "Toko Skincare Purwokerto" << endl;
    cout << "1. Tambah Data" << endl;
    cout << "2. Hapus Data" << endl;
    cout << "3. Update Data" << endl;
    cout << "4. Tambah Data Urutan Tertentu" << endl;
    cout << "5. Hapus Data Urutan Tertentu" << endl;
    cout << "6. Hapus Seluruh Data" << endl;
    cout << "7. Tampilkan Data" << endl;
    cout << "8. Exit" << endl;

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

    switch (choice)
    {
        case 1:
        {
            string namaProduk;
            int harga;
            cout << "Enter nama produk: ";
            cin.ignore();
            getline(cin, namaProduk);
            cout << "Enter harga: ";
            cin >> harga;
            list.push(namaProduk, harga);
            break;
        }
    }
}
```

```

    }
    case 2:
    {
        string namaProduk;
        cout << "Enter nama produk to delete: ";
        cin.ignore();
        getline(cin, namaProduk);
        list.deleteNode(namaProduk);
        break;
    }
    case 3:
    {
        string oldNamaProduk, newNamaProduk;
        int newHarga;
        cout << "Enter nama produk to update: ";
        cin.ignore();
        getline(cin, oldNamaProduk);
        cout << "Enter new nama produk: ";
        getline(cin, newNamaProduk);
        cout << "Enter new harga: ";
        cin >> newHarga;

        bool updated = list.update(oldNamaProduk,
newNamaProduk, newHarga);

        if (!updated)
        {
            cout << "Data not found" << endl;
        }
        break;
    }
    case 4:
    {

```

```

        string namaProduk;
        int harga, position;
        cout << "Enter nama produk to add: ";
        cin.ignore();
        getline(cin, namaProduk);
        cout << "Enter harga: ";
        cin >> harga;
        cout << "Enter position: ";
        cin >> position;

        list.insertAt(namaProduk, harga,
position);

        break;
    }
    case 5:
    {
        int position;
        cout << "Enter position to delete: ";
        cin >> position;

        cout << "This functionality is not
implemented." << endl;

        break;
    }
    case 6:
    {
        list.deleteAll();

        break;
    }
    case 7:
    {
        list.display();

        break;
    }

```

```

        }

    case 8:
    {
        return 0;
    }

    default:
    {
        cout << "Invalid choice" << endl;
        break;
    }
}

return 0;
}

```

Screenshots Output

Data sebelumnya:

```

PS C:\Semester 2\Praktikum Strudur Data\Modul_3>
dowsDebugLauncher.exe' '--stdin=Microsoft-MIEngin
-Error-3axiekqw.lml' '--pid=Microsoft-MIEngine-Pi
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
Enter your choice: 7
Nama Produk    Harga
Originote, 60000
Somethinc, 150000
Skintific, 100000
Wardah, 50000
Hanasui, 30000

```


Tambah Azarine dengan harga 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
Enter your choice: 4
Enter nama produk to add: Azarine
Enter harga: 65000
Enter position: 3
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
Enter your choice: 7
Nama Produk  Harga
Originote, 60000
Somethinc, 150000
Azarine, 65000
Skintific, 100000
Wardah, 50000
Hanasui, 30000
```

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
Enter your choice: 2
Enter nama produk to delete: 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
Enter your choice: 7
Nama Produk  Harga
Originote, 60000
Somethinc, 150000
Azarine, 65000
Skintific, 100000
Hanasui, 30000
```

Update produk Hanasui menjadi Cleora dengan harga 55.000

```
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
Enter your choice: 3
Enter nama produk to update: Hanasui
Enter new nama produk: Cleora
Enter new harga: 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
Enter your choice: 7
Nama Produk    Harga
Originote, 60000
Somethinc, 150000
Azarine, 65000
Skintific, 100000
Cleora, 55000
```

Deskripsi:

Program di atas adalah implementasi dari Doubly Linked List. Program ini memiliki 4 pointer yaitu prev dan next menunjuk ke node sebelumnya dan setelahnya, head dan tail menunjuk ke node pertama dan terakhir. Dalam program ini, terdapat dua class utama: **Node** dan **DoublyLinkedList**. Class **Node** merepresentasikan elemen tunggal dalam linked list dengan atribut namaProduk dan harga, serta dua pointer prev dan next. Class **DoublyLinkedList** digunakan untuk mengelola linked list itu sendiri. Program ini memungkinkan pengguna untuk menambah, menghapus, mengubah, atau menampilkan elemen dalam linked list. Dalam fungsi main, terdapat sebuah loop yang meminta pengguna untuk memilih aksi yang ingin dilakukan, seperti menambah, menghapus, mengubah, atau menampilkan data. Pengguna dapat memilih aksi tersebut dengan memasukkan nomor yang sesuai dengan pilihan menu yang disediakan. Kemudian, program akan menangani aksi sesuai dengan pilihan pengguna.

BAB V

KESIMPULAN

Kesimpulan yang dapat saya ambil, Single linked list merupakan koleksi linear dari data, yang disebut sebagai nodes, dimana setiap node akan menunjuk pada node lain melalui sebuah pointer. Linked List dapat didefinisikan pula sebagai kumpulan nodes yang merepresentasikan sebuah sequence. Double Linked List tidak jauh berbeda dengan Single Linked List. Jadi Double Linked List adalah Tambahan dari Single Linked List yang menambahkan satu pointer (yang nantinya dapat dikembangkan dengan banyak pointer).

Kedua jenis Linked List memiliki kelebihan dan kekurangan masing-masing. Single Linked List cocok digunakan dalam situasi di mana hanya navigasi maju diperlukan dan efisiensi ruang penyimpanan menjadi prioritas. Sedangkan Double Linked List lebih cocok digunakan ketika navigasi maju dan mundur diperlukan secara fleksibel, meskipun membutuhkan sedikit lebih banyak ruang penyimpanan.

BAB VI

DAFTAR PUSTAKA

- [1] Asisten Praktikum. 2024. Modul 3 “*Single Linked List dan Double Linked List*”. Diakses 02 April 2024, 15:00 WIB. <https://lms.ittelkom-pwt.ac.id/>
- [2] Wongso, Rini. 2017. “*Single Linked List*”. Diakses 02 April 2024, 17:00 WIB. <https://socs.binus.ac.id/2017/03/15/single-linked-list/>
- [3] Daisma Bali. “*Memahami Doubly Linked List dalam Strukt Data dengan mudah*”. Diakses 02 April 2024, 17:00 WIB. https://daismabali.com/artikel_detail/63/1/Memahami-Doubly-Linked-List-dalam-Struktu-Data-dengan-mudah.html