

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL III
SINGLE AND DOUBLE
LINKED LIST**



**Disusun Oleh :
WIDARI DWI HAYATI
2311102060**

**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

Linked List adalah jenis struktur data yang terdiri dari kumpulan data, yang disebut dengan simpul, yang disusun secara berurutan, terhubung, dinamis, dan terbatas. Setiap simpul pada linked list terhubung dengan simpul lainnya menggunakan pointer. Setiap komponen sering disebut dengan simpul, node, atau vertex, sedangkan pointer adalah alamat dari elemen. Secara rinci, setiap simpul pada linked list dibagi menjadi dua bagian: bagian pertama disebut bagian isi, informasi, atau data, yang menyimpan nilai yang disimpan oleh simpul. Bagian kedua disebut bagian pointer, yang berisi alamat dari simpul berikutnya atau simpul sebelumnya. Linked list dibentuk dengan cara menunjuk pointer "next" dari satu simpul ke simpul berikutnya dalam urutan. Pointer "next" dari simpul terakhir dalam daftar diset ke NULL, yang menunjukkan akhir dari daftar. Simpul pertama dalam daftar disebut "head" dan simpul terakhir disebut "tail".

Di dalam operasi Single Linked List, yang umum dilakukan adalah operasi penambahan dan penghapusan simpul pada awal atau akhir dari daftar. Selain itu, dilakukan pencarian dan pengambilan nilai pada simpul tertentu. Hal ini dikarenakan Single Linked List hanya membutuhkan satu pointer untuk setiap simpul, sehingga lebih efisien dalam penggunaan memori. Ada dua jenis linked list, yaitu Single Linked List dan Circular Linked List. Perbedaan Circular Linked List dan Single Linked List adalah bahwa pada Circular Linked List, penunjuk "next" pada node terakhir akan selalu merujuk ke node pertama. Hal ini berbeda dengan Single Linked List, yang penunjuk "next" pada node terakhir sering kali bernilai NULL.

b. Double Linked List

Double Linked List adalah jenis Linked List yang mirip dengan Single Linked List, tetapi dengan tambahan pointer yang menunjuk ke simpul sebelumnya. Hal ini memungkinkan untuk melakukan operasi penambahan dan penghapusan dengan efisiensi. Setiap simpul pada Double Linked List terdiri dari tiga bagian, yaitu elemen data, pointer next yang menunjuk ke simpul berikutnya, dan pointer prev yang menunjuk ke simpul sebelumnya.

Keuntungan dari Double Linked List adalah kemampuan melakukan operasi penghapusan dan penambahan dengan efisiensi, yang sangat berguna untuk beberapa algoritma yang membutuhkan operasi tersebut. Selain itu, Double Linked List juga memungkinkan traversal dari depan dan belakang dengan gampang. Namun, kekurangannya adalah membutuhkan memori yang lebih banyak dan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan.

Dalam sebuah linked list, terdapat dua pointer utama yang digunakan untuk mengindikasikan awal dan akhir dari linked list tersebut, yaitu pointer HEAD dan TAIL. Pointer HEAD menunjuk pada simpul pertama, sedangkan pointer TAIL menunjuk pada simpul terakhir. Sebuah linked list dianggap kosong jika isi pointer HEAD bernilai NULL. Selain itu, pointer prev dari HEAD selalu bernilai NULL, sementara pointer next dari TAIL juga selalu bernilai NULL sebagai penanda bahwa simpul tersebut adalah simpul terakhir.

B. Guided

Guided 1

Latihan Single Linked List

```
#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

```

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 C:\Users\ADVAN>

```

Deskripsi:

Program ini menggunakan Single Linked List. Di dalamnya terdapat beberapa fungsi, seperti `init()` untuk menginisialisasi head dan tail, `isEmpty()` untuk mengecek apakah Linked List kosong, `insertDepan()` dan `insertBelakang()` untuk menambahkan node di depan dan di belakang, `hitungList()` untuk menghitung jumlah node, `insertTengah()` untuk menambahkan node di posisi tertentu, `hapusDepan()` dan `hapusBelakang()` untuk menghapus node di depan dan di belakang, `hapusTengah()` untuk menghapus node di posisi tertentu, serta `ubahDepan()`, `ubahTengah()`, dan `ubahBelakang()` untuk mengubah nilai data node di depan, tengah, dan belakang. Selain itu, terdapat juga fungsi `clearList()` untuk menghapus semua node, dan `tampil()` untuk menampilkan semua node yang ada.

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 = temp;
        }
        delete temp;
    }
};
```

```

        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;

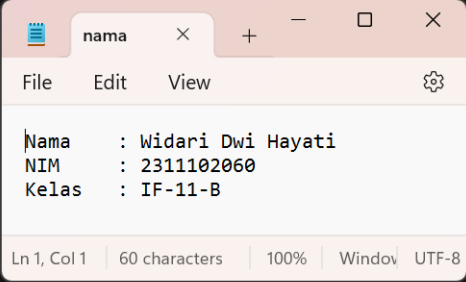
switch (choice) {
    case 1: {
        int data;
        cout << "Enter data to add: ";
        cin >> data;
        list.push(data);
        break;
    }
    case 2: {
        list.pop();
        break;
    }
    case 3: {
        int oldData, newData;
        cout << "Enter old data: ";
        cin >> 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;
    }
}
}

```

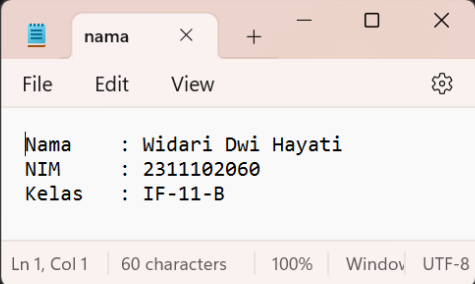
```
    return 0;
}
```

Screenshots Output

```
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: 23
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: 43
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: 63
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
63 43 23
```



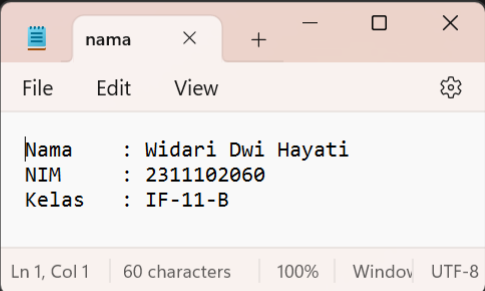
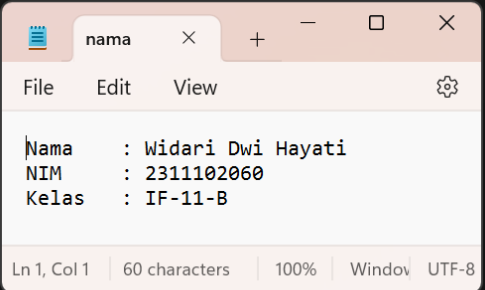
```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 2
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
43 23
```



```
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 3
Enter old data: 43
Enter new data: 73
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
73 23

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

1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 6
PS C:\Users\ADVAN>
```



Deskripsi:

Program ini menggunakan Double Linked List. Di dalamnya terdapat beberapa fungsi, seperti `push()` untuk menambahkan node di depan, `pop()` untuk menghapus node di depan, `update()` untuk mengubah nilai data di node tertentu, `deleteAll()` untuk menghapus semua node, dan `display()` untuk menampilkan seluruh isi node.

Dalam `main()`, terdapat sebuah perulangan `while` yang akan terus berjalan hingga pengguna memilih untuk keluar. Pengguna akan diminta untuk memilih opsi yang tersedia, yaitu menambah data, menghapus data, mengubah data, menghapus semua data, atau menampilkan data.

Jika pengguna memilih opsi 1 (menambah data), maka program akan meminta inputan data yang akan dipush ke dalam Double Linked List.

Jika pengguna memilih opsi 2 (menghapus data), maka program akan menghapus node pertama.

Jika pengguna memilih opsi 3 (mengubah data), maka program akan meminta inputan data yang akan diganti dan data yang akan dicari untuk diganti.

Jika pengguna memilih opsi 4 (menghapus semua data), maka program akan menghapus semua node dalam Double Linked List.

Jika pengguna memilih opsi 5 (menampilkan data), maka program akan menampilkan semua node yang ada di dalam Double Linked List.

C. Unguided/Tugas

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

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

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

Node* head;
Node* tail;
```

```

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

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

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

// Tambah Node di belakang
void insertBelakang(string nama, int usia) {
    Node* baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    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(string nama, int usia, 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->nama = nama;
        baru->usia = usia;
        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 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(string nama, int usia) {
    if (!isEmpty()) {
        head->nama = nama;
        head->usia = usia;
    } else {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah data Node di posisi tengah
void ubahTengah(string nama, int usia, 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->nama = nama;
            bantu->usia = usia;
        }
    } 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() {
    int nomor = 1;
    cout << "No." << "\tNama" << "\t\tUsia" << endl;
    if (!isEmpty()) {
        Node* bantu = head;
        while (bantu != NULL) {
            cout << nomor << "." << "\t" << bantu->nama;
            cout << "\t\t" << bantu->usia << endl;
            bantu = bantu->next;
            nomor++;
        }
    } else {
        cout << "List masih kosong!" << endl;
    }
    cout << endl;
}

int main() {
    init();
    string nama;
    int usia;
    cout << "Masukkan nama: ";
    cin >> nama;
    cout << "Masukkan usia: ";
    cin >> usia;
    insertDepan(nama, usia);
    insertBelakang("John", 19);
    insertBelakang("Jane", 20);
    insertBelakang("Michael", 18);
    insertBelakang("Yusuke", 19);
    insertBelakang("Akechi", 20);
    insertBelakang("Hoshino", 18);
    insertBelakang("Karin", 18); tampil();
    hapusTengah(6); tampil();
    insertTengah("Futaba", 18, 3); tampil();
    insertDepan("Igor", 20); tampil();
    ubahTengah("Reyn", 18, 6); tampil();
}

```

Screenshots Output

Masukkan nama: Widari
Masukkan usia: 18

| No. | Nama | Usia |
|-----|---------|------|
| 1. | Widari | 18 |
| 2. | John | 19 |
| 3. | Jane | 20 |
| 4. | Michael | 18 |
| 5. | Yusuke | 19 |
| 6. | Akechi | 20 |
| 7. | Hoshino | 18 |
| 8. | Karin | 18 |

nama

File Edit View

Nama : Widari Dwi Hayati
NIM : 2311102060
Kelas : IF-11-B

Ln 3, Col 17 | 60 characters | 100% | Window UTF-8

| No. | Nama | Usia |
|-----|---------|------|
| 1. | Widari | 18 |
| 2. | John | 19 |
| 3. | Jane | 20 |
| 4. | Michael | 18 |
| 5. | Yusuke | 19 |
| 6. | Hoshino | 18 |
| 7. | Karin | 18 |

nama

File Edit View

Nama : Widari Dwi Hayati
NIM : 2311102060
Kelas : IF-11-B

Ln 3, Col 17 | 60 characters | 100% | Window UTF-8

| No. | Nama | Usia |
|-----|---------|------|
| 1. | Igor | 20 |
| 2. | Widari | 18 |
| 3. | John | 19 |
| 4. | Futaba | 18 |
| 5. | Jane | 20 |
| 6. | Michael | 18 |
| 7. | Yusuke | 19 |
| 8. | Hoshino | 18 |
| 9. | Karin | 18 |

nama

File Edit View

Nama : Widari Dwi Hayati
NIM : 2311102060
Kelas : IF-11-B

Ln 3, Col 17 | 60 characters | 100% | Window UTF-8

Deskripsi:

Program ini menggunakan Single Linked List, yang digunakan untuk menyimpan data mengenai nama dan usia. Di dalamnya terdapat beberapa fungsi, yaitu:

init(): Fungsi untuk menginisialisasi Single Linked List. isEmpty(): Fungsi untuk mengecek apakah Single Linked List kosong atau tidak. insertDepan(string nama, int usia): Fungsi untuk menambahkan data di depan Single Linked List. insertBelakang(string nama, int usia): Fungsi untuk menambahkan data di belakang Single Linked List. hitungList(): Fungsi untuk menghitung jumlah simpul di Single Linked List. insertTengah(string nama, int usia, int posisi): Fungsi untuk menambahkan data di posisi tengah Single Linked List. hapusDepan(): Fungsi untuk menghapus data di depan Single Linked List. hapusTengah(int posisi): Fungsi untuk menghapus data di posisi tengah Single Linked List. ubahDepan(string nama, int usia): Fungsi untuk mengubah data di depan Single Linked List. ubahTengah(string nama, int usia, int posisi): Fungsi untuk mengubah data di posisi tengah Single Linked List. clearList(): Fungsi untuk menghapus semua data di Single Linked List. tampil(): Fungsi untuk menampilkan semua data di Single Linked List.

Dalam main(), dilakukan beberapa pemanggilan fungsi, yaitu:

Memanggil init() untuk menginisialisasi Single Linked List. Memasukkan nama dan usia pertama, dan memanggil insertDepan(nama, usia) untuk menambahkan data di depan Single Linked List. Memasukkan nama dan usia kedua, dan memanggil insertBelakang(nama, usia) untuk menambahkan data di belakang Single Linked List. Memasukkan beberapa nama dan usia lainnya, dan memanggil insertBelakang(nama, usia) untuk menambahkan data di belakang Single Linked List. Menampilkan data Single Linked List dengan memanggil tampil(). Menghapus data di posisi tengah Single Linked List dengan memanggil hapusTengah(6). Menambahkan data di posisi tengah Single Linked List dengan memanggil insertTengah(nama, usia, 3). Menambahkan data di depan Single Linked List dengan memanggil insertDepan(nama, usia). Mengubah data di posisi tengah Single Linked List dengan memanggil ubahTengah(nama, usia, 6). Menampilkan data Single Linked List dengan memanggil tampil().

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 |

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

class Node {
public:
    string nama, harga;
    Node* prev;
    Node* next;
};

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

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

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

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

        head = newNode;
    }
}
```

```

void push_back(string nama, string harga) {
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->harga = harga;
    newNode->prev = nullptr;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
        tail = newNode;
    } else {
        newNode->prev = tail;
        tail->next = newNode;
        tail = 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 index, string newName, string newPrice) {
    if (index < 0 || index >= getLength()) {
        cout << "Invalid index" << endl;
        return false;
    }

    Node* current = head;
    for (int i = 0; i < index; i++) {
        current = current->next;
    }

    current->nama = newName;
    current->harga = newPrice;

    return true;
}

```



```

}

void insert(int index, string nama, string harga) {
    if (index < 0 || index > getLength()) {
        return;
    }

    if (index == 0) {
        push(nama, harga);
        return;
    }

    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->harga = harga;
    newNode->prev = nullptr;
    newNode->next = head;

    Node* current = head;
    for (int i = 0; i < index - 1; i++) {
        current = current->next;
    }

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

void remove(int index) {
    if (index < 0 || index >= getLength()) {
        return;
    }

    if (index == 0) {
        pop();
        return;
    }

    Node* current = head;
    for (int i = 0; i < index; i++) {
        current = current->next;
    }

```

```

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

int getLength() {
    Node* current = head;
    int length = 0;
    while (current != nullptr) {
        length++;
        current = current->next;
    }
    return length;
}

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 << " _____ \n";
    cout << "| Nama Produk " << "\t\t| Harga " << "\t\t|\n";
    cout << " _____ \n";
    while (current != nullptr) {
        cout << "| " << current->nama << "\t\t| " << current->harga << "\t\t|" << endl;
        current = current->next;
        cout << " _____ \n";
    }
    cout << endl;
}

```

```

};

int main() {
    DoublyLinkedList list;
    list.push("Originote", "60.000");
    list.push_back("Somethinc", "150.000");
    list.push_back("Skintific", "100.000");
    list.push_back("Wardah", "50.000");
    list.push_back("Hanasui", "30.000");
    list.display();
    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;

        string nama, harga;

        switch (choice) {
            case 1: {
                cout << "Enter nama: ";
                cin >> nama;
                cout << "Enter harga: ";
                cin >> harga;
                list.push(nama, harga);
                break;
            }
            case 2: {
                list.pop();
                break;
            }
            case 3: {
                int index;
                string newName, newPrice;
                cout << "Enter index: ";
                cin >> index;
                cout << "Enter new nama: ";
                cin >> newName;
            }
        }
    }
}

```

```

        cout << "Enter new harga: ";
        cin >> newPrice;
        bool updated = list.update(index, newName, newPrice);
        if (!updated) {
            cout << "Index not found" << endl;
        }
        break;
    }
    case 4: {
        string nama, harga;
        cout << "Enter nama: ";
        cin >> nama;
        cout << "Enter harga: ";
        cin >> harga;
        cout << "Enter index: ";
        int index;
        cin >> index;
        list.insert(index, nama, harga);
        break;
    }
    case 5: {
        cout << "Enter index: ";
        int index;
        cin >> index;
        list.remove(index);
        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

| 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

Enter your choice: 4
Enter nama: Azarine
Enter harga: 65.000
Enter index: 2

nama

File Edit View

Nama : Widari Dwi Hayati
NIM : 2311102060
Kelas : IF-11-B

Ln 3, Col 17 60 characters 100% Window UTF-8

Enter your choice: 4
Enter nama: Azarine
Enter harga: 65.000
Enter index: 2

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 | 60.000 |
| Somethinc | 150.000 |
| Azarine | 65.000 |
| Skintific | 100.000 |
| Wardah | 50.000 |
| Hanasui | 30.000 |

nama

File Edit View

Nama : Widari Dwi Hayati
NIM : 2311102060
Kelas : IF-11-BSS

Ln 3, Col 19 62 characters 100% Window UTF-8

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: 5

Enter index: 4

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 | 60.000 |
| Somethinc | 150.000 |
| Azarine | 65.000 |
| Skintific | 100.000 |
| Hanasui | 30.000 |

nama

File Edit View

Nama : Widari Dwi Hayati

NIM : 2311102060

Kelas : IF-11-BSS

Ln 3, Col 19 | 62 characters | 100% | Window UTF-8

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 index: 4

Enter new nama: Cleora

Enter new 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: 7

nama

File Edit View

Nama : Widari Dwi Hayati

NIM : 2311102060

Kelas : IF-11-BSS

Ln 3, Col 19 | 62 characters | 100% | Window UTF-8

```
| Nama Produk | Harga |
|-----|-----|
| Originote  | 60.000 |
| Somethinc  | 150.000 |
| Azarine    | 65.000 |
| Skintific  | 100.000 |
| Cleora     | 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: 8
PS C:\Users\ADVAN>
```

nama

File Edit View

Nama : Widari Dwi Hayati
NIM : 2311102060
Kelas : IF-11-BSS

Ln 3, Col 19 | 62 characters | 100% | Window UTF-8

Deskripsi:

Program ini menggunakan Double Linked List, yang digunakan untuk menyimpan data mengenai nama produk dan harga, yang digunakan oleh sebuah toko skincare. Di dalamnya terdapat beberapa fungsi, yaitu:

push(string nama, string harga): Fungsi untuk menambahkan data di depan Double Linked List. push_back(string nama, string harga): Fungsi untuk menambahkan data di belakang Double Linked List. pop(): Fungsi untuk menghapus data di depan Double Linked List. update(int index, string newName, string newPrice): Fungsi untuk mengubah data di Double Linked List dengan menggunakan index. insert(int index, string nama, string harga): Fungsi untuk menambahkan data di posisi tertentu Double Linked List. remove(int index): Fungsi untuk menghapus data di posisi tertentu Double Linked List. getLength(): Fungsi untuk menghitung jumlah simpul di Double Linked List. deleteAll(): Fungsi untuk menghapus semua data di Double Linked List. display(): Fungsi untuk menampilkan semua data di Double Linked List.

Dalam main(), dilakukan beberapa pemanggilan fungsi, yaitu:

Memanggil fungsi push dan push_back untuk menambahkan data di depan dan belakang Double Linked List. Memanggil fungsi pop untuk menghapus data di depan Double Linked List. Memanggil fungsi update untuk mengubah data di Double Linked List. Memanggil fungsi insert untuk menambahkan data di posisi tertentu Double Linked List. Memanggil fungsi remove untuk menghapus data di posisi tertentu Double Linked List. Memanggil fungsi getLength untuk mengetahui jumlah simpul yang ada di Double Linked List. Memanggil fungsi deleteAll untuk menghapus semua data di Double Linked List. Memanggil fungsi display untuk menampilkan semua data di Double Linked List.

D. Kesimpulan

Single Linked List adalah struktur data yang memiliki pointer yang hanya menunjuk ke node berikutnya. Single Linked List memiliki keuntungan dalam melakukan operasi penambahan dan penghapusan pada node yang diinginkan, serta membutuhkan memori yang lebih sedikit daripada Array. Namun, Single Linked List memiliki kekurangan dalam melakukan traversal pada list dari belakang.

Double Linked List adalah struktur data yang memiliki dua pointer, yaitu pointer yang menunjuk ke node sebelumnya dan node berikutnya. Double Linked List memiliki keuntungan dalam melakukan operasi penambahan dan penghapusan pada node mana saja, serta kelebihan dalam melakukan traversal pada list baik dari depan maupun dari belakang. Namun, Double Linked List membutuhkan memori yang lebih banyak dan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan.

E. Referensi (APA)

Stroustrup, B. (2021). A Tour of C++ (2nd ed.). Addison-Wesley Professional.

Lippman, S. L., Lajoie, J., & Moo, B. (2020). C++ Primer (6th ed.). Addison-Wesley Professional.

Malik, D. S. (2014). C++ Programming: From Problem Analysis to Program Design (7th ed.). Cengage Learning.