

LAPORAN PRAKTIKUM

MODUL III SINGLE AND DOUBLE LINKED LIST



Disusun oleh:
Muhammad Agha Zulfadhli
NIM: 2311102015

Dosen Pengampu:
Wahyu Andi Saputra, S .Pd, M .Eng.

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

BAB I

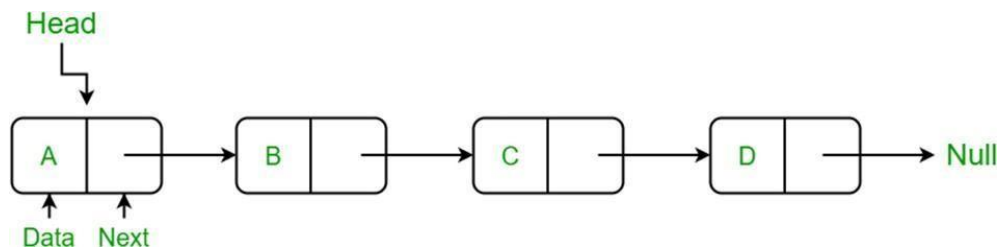
TUJUAN PRAKTIKUM

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

BAB II

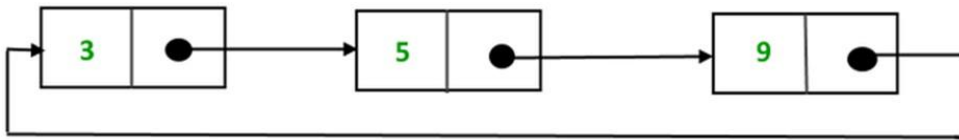
DASAR TEORI

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



Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List umumnya lebih efisien dalam penggunaan memori dibandingkan dengan jenis Linked List lainnya, seperti Double Linked List dan Circular Linked List.

Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.

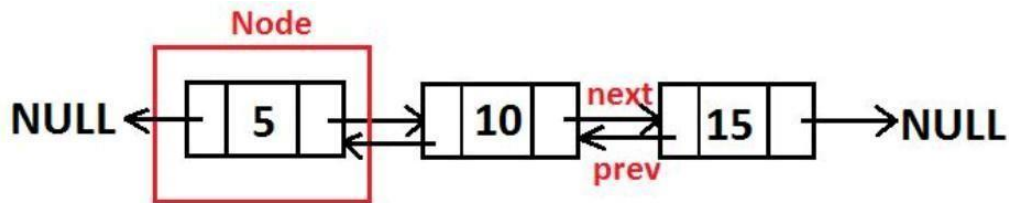


b) Double Linked List

Double Linked List adalah struktur data Linked List yang mirip dengan Single Linked List, namun dengan tambahan satu pointer tambahan pada setiap simpul yaitu pointer prev yang menunjuk ke simpul sebelumnya. Dengan adanya pointer prev, Double Linked List memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul mana saja secara efisien. Setiap simpul pada Double Linked List memiliki tiga elemen penting, yaitu elemen data (biasanya berupa nilai), pointer next yang menunjuk ke simpul berikutnya, dan pointer prev yang menunjuk ke simpul sebelumnya.

Keuntungan dari Double Linked List adalah memungkinkan untuk melakukan operasi penghapusan dan penambahan pada simpul dimana saja dengan efisien, sehingga sangat berguna dalam implementasi beberapa algoritma yang membutuhkan operasi tersebut. Selain itu, Double Linked List juga memungkinkan kita untuk melakukan traversal pada list baik dari depan (head) maupun dari belakang (tail) dengan mudah. Namun, kekurangan dari Double Linked List adalah penggunaan memori yang lebih besar dibandingkan dengan Single Linked List, karena setiap simpul membutuhkan satu pointer tambahan. Selain itu, Double Linked List juga membutuhkan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan jika dibandingkan dengan Single Linked List.

Representasi sebuah double linked list dapat dilihat pada gambar berikut ini:



Di dalam sebuah linked list, ada 2 pointer yang menjadi penunjuk utama, yakni pointer HEAD yang menunjuk pada node pertama di dalam linked list itu sendiri dan pointer TAIL yang menunjuk pada node paling akhir di dalam linked list. Sebuah linked list dikatakan kosong apabila isi pointer head adalah NULL. Selain itu, nilai pointer prev dari HEAD selalu NULL, karena merupakan data pertama. Begitu pula dengan pointer next dari TAIL yang selalu bernilai NULL sebagai penanda data terakhir.

BAB III

GUIDED

1. Guided 1

Source code

```
#include <iostream>
using namespace std;
///PROGRAM SINGLE LINKED LIST NON-CIRCULAR
//Deklarasi Struct Node
struct Node{
    //komponen/member
    int data;
    string kata;
    Node *next;
};

Node *head;
Node *tail;

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

// Pengecekan
bool isEmpty(){
    if (head == NULL)
        return true;
    else
        return false;
}

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

```

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

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

//Tambah Tengah
void insertTengah(int data, string kata, int posisi){
    if( posisi < 1 || posisi > hitungList() ){
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if( posisi == 1){
        cout << "Posisi bukan posisi tengah" <<
        endl;
    } else{
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;
        baru->kata = kata;
        // tranversing
        bantu = head;
        int nomor = 1;
        while( nomor < posisi - 1 ){
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

```

```

}

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

//Hapus Belakang
void hapusBelakang() {
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false){
        if (head != tail){
            hapus = tail;
            bantu = head;
            while (bantu->next != tail){
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
            delete hapus;
        } else{
            head = tail = NULL;
        }
    } else{
        cout << "List kosong!" << endl;
    }
}

//Hapus Tengah
void hapusTengah(int posisi){
    Node *hapus, *bantu, *bantu2;
    if( posisi < 1 || posisi > hitungList() ){
        cout << "Posisi di luar jangkauan" << endl;
    } else if( posisi == 1){
        cout << "Posisi bukan posisi tengah" << endl;
    }else{
        int nomor = 1;
        bantu = head;
        while( nomor <= posisi ){
            if( nomor == posisi-1 ){

```



```

        bantu2 = bantu;
    }
    if( nomor == posisi ){
        hapus = bantu;
    }
    bantu = bantu->next;
    nomor++;
}
bantu2->next = bantu;
delete hapus;
}

//Ubah Depan
void ubahDepan(int data, string kata){
    if (isEmpty() == false){
        head->data = data;
        head->kata = kata;
    } else{
        cout << "List masih kosong!" << endl;
    }
}

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

int main(){
    init();
    insertDepan(3, "Ms. Puff");
    tampil();
    insertBelakang(5, "Plankton");
    tampil();
    insertDepan(2, "Squidward");
    tampil();
    insertDepan(1, "Gary");
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, "Spongebob", 2);
    tampil();
    hapusTengah(2);
    tampil();
}

```

```

    ubahDepan(1,"Mr. Krabs");
    tampil();
    ubahBelakang(8,"Sandy");
    tampil();
    ubahTengah(11,"Patrick", 2);
    tampil();
    return 0;
}

```

Screenshoot program

```

Github\Modul 3> cd "d:\SelfSyncting\Personal\1. WORKER\KULIAH\STRUKDAT\Github\Modul 3\" ; if ($?) { g++ GUIDED1.cpp -o GUIDED1 } ; if ($?) { .\GUIDED1 }
3. Ms. Puff

3. Ms. Puff
5. Plankton

2. Squidward
3. Ms. Puff
5. Plankton

1. Gary
2. Squidward
3. Ms. Puff
5. Plankton

2. Squidward
3. Ms. Puff
5. Plankton

2. Squidward
3. Ms. Puff

2. Squidward
7. Spongebob
3. Ms. Puff

2. Squidward
3. Ms. Puff

1. Mr. Krabs
3. Ms. Puff

1. Mr. Krabs
8. Sandy

1. Mr. Krabs
11. Patrick

PS D:\SelfSyncting\Personal\1. WORKER\KULIAH\STRUKDAT\Github\Modul 3>

```

Deskripsi program

Pada awalnya, terdapat inisialisasi untuk head dan tail yang dideklarasikan sebagai pointer ke Node yang awalnya diset NULL. Fungsi-fungsi utama termasuk penambahan data ke depan, belakang, dan tengah, penghapusan data dari depan, belakang, dan tengah, serta fungsi untuk mengubah data di depan, belakang, dan tengah. Selain itu, ada juga fungsi untuk membersihkan seluruh isi linked list dan untuk menampilkan isi linked list.

Di dalam fungsi main, program melakukan serangkaian operasi pada linked list yang telah diinisialisasi. Ini termasuk menambah dan menghapus elemen, serta mengubah nilai dari beberapa elemen. Setiap operasi tersebut diikuti dengan pemanggilan fungsi tampil() yang bertujuan untuk menampilkan isi linked list setelah operasi dilakukan.

Misalnya, program pertama-tama menambahkan beberapa data ke linked list, kemudian menampilkan isi linked list setiap kali sebuah operasi ditambahkan atau dihapus. Misalnya, setelah beberapa operasi penambahan dan penghapusan dilakukan, fungsi tampil() dipanggil untuk menunjukkan perubahan yang terjadi pada linked list.

2. Guided 2

Source code

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

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

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

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

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

```

    bool update(int oldData, int newData, string oldKata, string
newKata) {
        Node* current = head;
        while (current != nullptr) {
            if (current->data == oldData && current->kata ==
oldKata) {
                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;

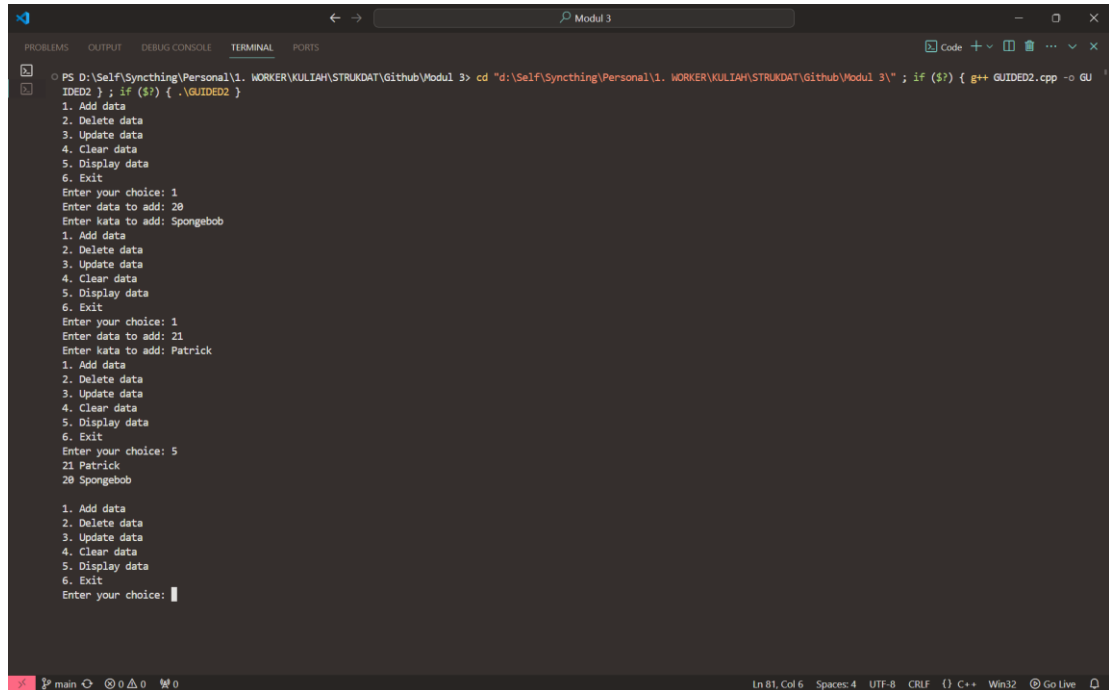
```

```

        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 oldKata, newKata;
        cout << "Enter old data: ";
        cin >> oldData;
        cout << "Enter new data: ";
        cin >> newData;
        cout << "Enter old kata: ";
        cin >> oldKata;
        cout << "Enter new kata: ";
        cin >> newKata;
        bool updated = list.update(oldData, newData,
oldKata ,newKata);
        if (!updated) {
            cout << "Data not found" << endl;
        }
        break;
    }
    case 4: {
        list.deleteAll();
        break;
    }
    case 5: {
        list.display();
        break;
    }
    case 6: {
        return 0;
    }
    default: {
        cout << "Invalid choice" << endl;
        break;
    }
}
}
return 0;
}

```

Screenshoot program



```
PS D:\Self\Synching\Personal\1. WORKER\KULIAH\STRUKDAT\Github\Modul 3> cd "d:\Self\Synching\Personal\1. WORKER\KULIAH\STRUKDAT\Github\Modul 3\" ; if ($?) { g++ GUIDED2.cpp -o GU
GUIDED2 } ; if ($?) { .\GUIDED2 }
1. Add data
2. Delete data
3. Update data
4. Clean data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 20
Enter kata to add: Spongebob
1. Add data
2. Delete data
3. Update data
4. Clean data
5. Display data
6. Exit
Enter your choice: 1
Enter data to add: 21
Enter kata to add: Patrick
1. Add data
2. Delete data
3. Update data
4. Clean data
5. Display data
6. Exit
Enter your choice: 5
21 Patrick
20 Spongebob
1. Add data
2. Delete data
3. Update data
4. Clean data
5. Display data
6. Exit
Enter your choice: 
```

Deskripsi program

Dalam program ini, terdapat dua kelas, yaitu kelas Node yang merepresentasikan setiap node dalam Doubly Linked List, dan kelas DoublyLinkedList yang berisi operasi-operasi untuk memanipulasi linked list tersebut.

Kelas Node memiliki empat anggota data, yaitu data (integer) yang menyimpan nilai, kata (string) yang menyimpan informasi tambahan, serta dua pointer, prev dan next, yang menunjuk ke node sebelumnya dan setelahnya.

Kelas DoublyLinkedList memiliki dua pointer, head dan tail, yang masing-masing menunjuk ke node pertama (head) dan terakhir (tail) dalam linked list. Konstruktor kelas ini menginisialisasi head dan tail menjadi nullptr yang menandakan linked list kosong.

Beberapa operasi yang didefinisikan dalam kelas DoublyLinkedList antara lain:

1. `push(int data, string kata)`: Menambahkan node baru ke depan linked list.
2. `pop()`: Menghapus node dari depan linked list.
3. `update(int oldData, int newData, string oldKata, string newKata)`: Mengubah nilai data dan kata pada node tertentu.
4. `deleteAll()`: Menghapus seluruh node dalam linked list.
5. `display()`: Menampilkan isi linked list.

Dalam fungsi `main`, program memberikan menu kepada pengguna untuk melakukan operasi-operasi tersebut. Pengguna dapat menambahkan data, menghapus data, mengubah data, membersihkan seluruh data, menampilkan data, dan keluar dari program. Setiap operasi yang dipilih pengguna akan dieksekusi sesuai dengan pilihan yang dimasukkan.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

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

struct Node {
    string nama_anda;
    int usia_anda;
    Node *next;
};

void addNode(Node *&list, string name = "", int age = -1, int
placement = 0) {
    Node *newNode = new Node;
    newNode->next = NULL;
    int iterator = 0;

    if (placement < 0) {
        // Jika placement ngawur
        cout << "Urutan invalid !";
        return;
    }

    if (name != "" && age != -1) {
        // jika variabel di fungsi ada
        newNode->nama_anda = name;
        newNode->usia_anda = age;
    } else {
        // jika variabel kosong
        cout << "Input nama : ";
        cin >> newNode->nama_anda;
        cout << "Input usia : ";
        cin >> newNode->usia_anda;
    }

    // Jika list kosong
    if (list == NULL || list->nama_anda == "") {
        list = newNode;
        return;
    }

    // Perulangan hingga terakhir
    Node *current = list;
    while (current->next != nullptr) {
        if (placement != 0) {
```

```

        if (/*wkwk*/ iterator+2 == placement) {break;}
        if (/*kocak*/ 1 == placement) {
            newNode->next = current->next;
            current->nama_anda = newNode->nama_anda;
            current->usia_anda = newNode->usia_anda;
            return;
        }
        iterator++;
    }
    current = current->next;
}
newNode->next = current->next;
current->next = newNode;
}

void initList(Node *&list) {
    cout << "----- INIT\n";
    addNode(list, "John", 19);
    addNode(list, "Jane", 20);
    addNode(list, "Michael", 18);
    addNode(list, "Yusuke", 19);
    addNode(list, "Akechi", 20);
    addNode(list, "Hoshino", 18);
    addNode(list, "Karin", 18);
}

void printNodes(Node *&list) {

    cout << "----- PRINT OUT\n";
    // Jika list kosong
    if (list->nama_anda == "") {
        cout << "kosong :[\n";
        return;
    }

    Node *counter = list;
    cout << left << setw(18) << "Nama";
    cout << left << setw(4) << "Usia" << endl;
    while (counter != NULL) {
        cout << left << setw(18) << counter->nama_anda;
        cout << left << setw(4) << counter->usia_anda << endl;
        counter = counter->next;
    }
}

void deleteNode(Node *&list) {

    string deleteThis;
    Node *current = list;
    Node *previous = NULL;

    printNodes(list);

```

```

        // jika list kosong
        if (list == NULL) {
            cout << "LIST KOSONG CURUT";
            return;
        }
        cout << "---- DELETE\n";
        // deleteThis->nama_anda = "Akechi"; (buat tugas 1b)
        cout << "Hapus nama : ";
        cin >> deleteThis;

        while (deleteThis != current->nama_anda) {
            previous = current;
            current = current->next;
        }

        // If the note to delete is the first node
        if (previous == NULL) {
            list = list->next;
        } else {
            previous->next = current->next;
        }
        delete current;
    }

void changeNode(Node *&list) {

    string oldName, newName;

    printNodes(list);
    cout << "----- CHANGE\n";
    cout << "Nama siapa yang ingin diganti : ";
    cin >> oldName;
    cout << "Nama baru : ";
    cin >> newName;

    Node *current = list;
    while (oldName != current->nama_anda) {current = current->next;}
    current->nama_anda = newName;
}

void addNodeSpecified(Node *&list) {

    string newName;
    int age,placement;

    printNodes(list);
    cout << "----- ADD\n";
    cout << "Nama : ";
    cin >> newName;

```

```

        cout << "Usia : ";
        cin >> age;
        cout << "Placement : ";
        cin >> placement;

        addNode(list, newName, age, placement);
    }

int main() {
    Node* list = new Node;
    //compeler
    list->nama_anda = "";
    list->usia_anda = -1;
    list->next = NULL;

    int choice = 0;

    do {
        cout << "----- MENU\nMenu orang program yeah\n\n1.
Tampilkan list\n2. Tambah orang\n3. Tambah orang di urutan \n4.
Ganti orang\n5. Inisialisasi list orang (template)\n6. Hapus
orang\n7. Exit";
        cout << "\n\nPilih [1-7] : ";
        cin >> choice;

        switch (choice) {
            case 1:
                printNodes(list);
                break;

            case 2:
                cout << "----- ADD\n";
                addNode(list);
                break;

            case 3:
                addNodeSpecified(list);
                break;

            case 4:
                changeNode(list);
                break;

            case 5:
                initList(list);
                cout << "DONE\n";
                break;

            case 6:
                deleteNode(list);
                break;
        }
    } while (choice != 7);
}

```

```

        default:
            break;
    }
} while (choice != 7);
}

```

Screenshot program

A. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah). Data pertama yang dimasukkan adalah nama dan usia anda.

```

PS D:\Self\Something\Personal\1. MONK\KULIAH\STRUKTUR\Modul 3> cd "D:\Self\Something\Personal\1. MONK\KULIAH\STRUKTUR\Modul 3\" ; if ($?) { g++ UNMAJEDDED.cpp -o UNMAJEDDED } ; if ($?) { .\UNMAJEDDED }

----- MENU
Menu orang program yuah

1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] : 5
----- INIT
DONE
----- MENU
Menu orang program yuah

1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

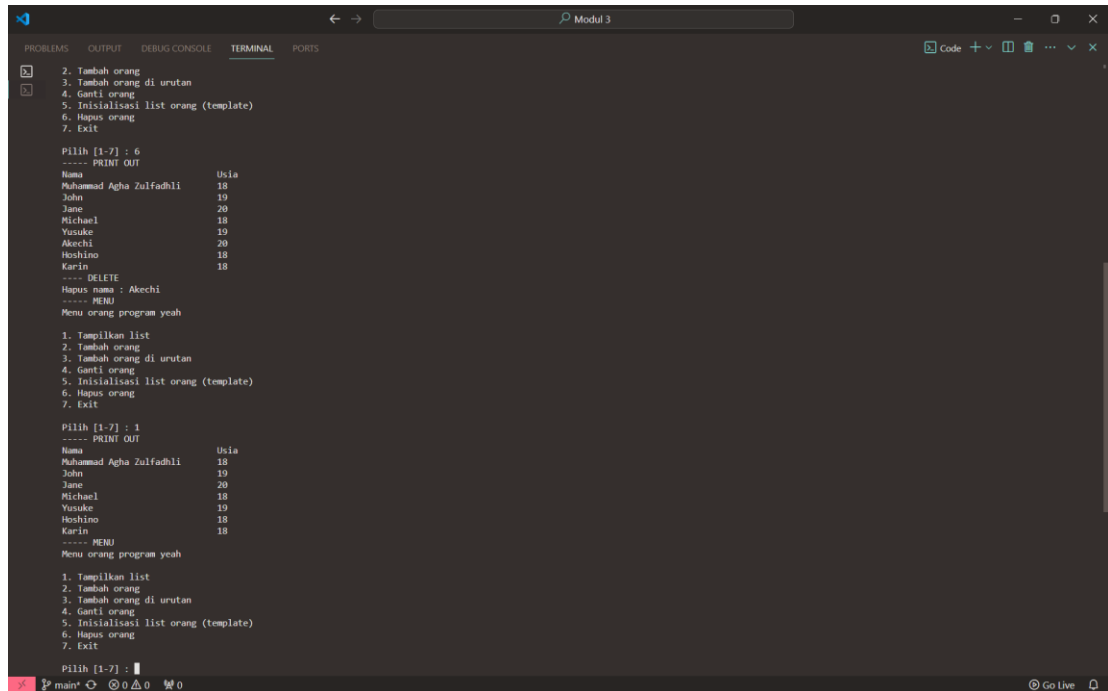
Pilih [1-7] : 1
----- PRINT OUT
Nama      Usia
Muhammad Agha Zulfaidhi 18
Jaka      19
Jawa      20
Michael   18
Yusuf     19
Akochi    20
Rochino   18
Karlin    18
----- MENU
Menu orang program yuah

1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] :

```

B. Hapus data Akechi.



```
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] : 6
----- PRINT OUT
Nama          Usia
Muhammad Agha Zulfadhli 18
John          19
Jane          20
Michael       18
Yusuke        19
Akechi        20
Hoshino       18
Karin         18
----- DELETE
Hapus nama : Akechi
----- MENU
Menu orang program yeah

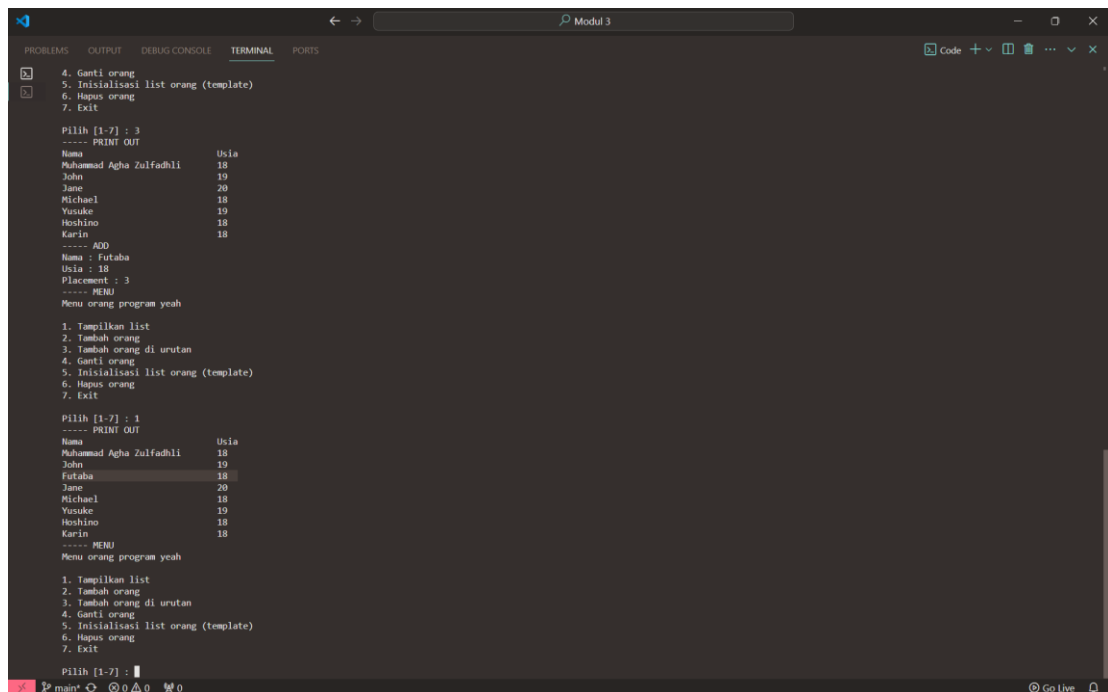
1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] : 1
----- PRINT OUT
Nama          Usia
Muhammad Agha Zulfadhli 18
John          19
Jane          20
Michael       18
Yusuke        19
Akechi        20
Hoshino       18
Karin         18
----- MENU
Menu orang program yeah

1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] :
```

C. Tambahkan data berikut diantara John dan Jane : Futaba 18.



```
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] : 3
----- PRINT OUT
Nama          Usia
Muhammad Agha Zulfadhli 18
John          19
Jane          20
Michael       18
Yusuke        19
Hoshino       18
Karin         18
----- ADD
Nama : Futaba
Usia : 18
Placement : 3
----- MENU
Menu orang program yeah

1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] : 1
----- PRINT OUT
Nama          Usia
Muhammad Agha Zulfadhli 18
John          19
Futaba        18
Jane          20
Michael       18
Yusuke        19
Hoshino       18
Karin         18
----- MENU
Menu orang program yeah

1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] :
```

D. Tambahkan data berikut diawal : Igor 20.

```
6. Hapus orang
7. Exit

Pilih [1-7] : 3
----- PRINT OUT
Nama                Usia
Muhammad Agha Zulfadhli 18
John                 19
Futaba               18
Jane                 20
Michael              18
Yusuke               19
Hoshino              18
Karin                18
----- ADD
Nama : Igor
Usia : 20
Placement : 1
----- MENU
Menu orang program yeah

1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] : 1
----- PRINT OUT
Nama                Usia
Igor                 20
Muhammad Agha Zulfadhli 18
John                 19
Futaba               18
Jane                 20
Michael              18
Yusuke               19
Hoshino              18
Karin                18
----- MENU
Menu orang program yeah

1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] :
```

E. Ubah data Michael menjadi : Reyn 18.

```
7. Exit

Pilih [1-7] : 4
----- PRINT OUT
Nama                Usia
Igor                 20
Muhammad Agha Zulfadhli 18
John                 19
Futaba               18
Jane                 20
Michael              18
Yusuke               19
Hoshino              18
Karin                18
----- CHANGE
Nama siapa yang ingin diganti : Michael
Nama baru : Reyn
Berapa umur dia : 18
----- MENU
Menu orang program yeah

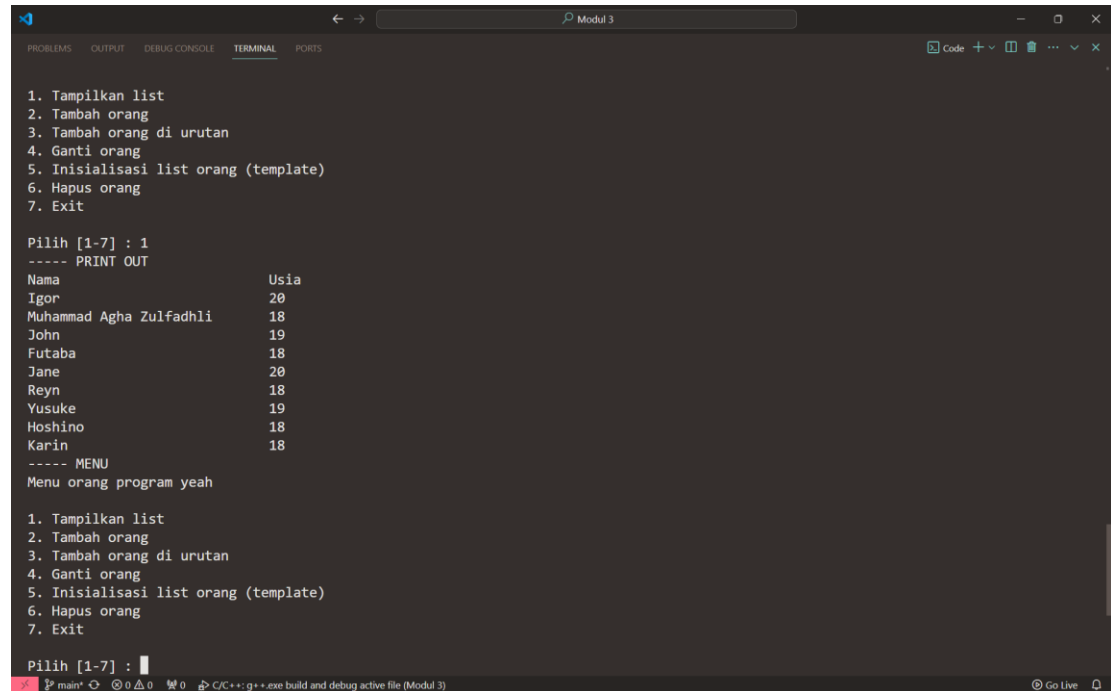
1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] : 1
----- PRINT OUT
Nama                Usia
Igor                 20
Muhammad Agha Zulfadhli 18
John                 19
Futaba               18
Jane                 20
Reyn                 18
Yusuke               19
Hoshino              18
Karin                18
----- MENU
Menu orang program yeah

1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] :
```


F. Tampilkan seluruh data.



```
1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] : 1
----- PRINT OUT
Nama                               Usia
Igor                               20
Muhammad Agha Zulfadhli           18
John                               19
Futaba                             18
Jane                               20
Reyn                               18
Yusuke                             19
Hoshino                            18
Karin                              18
----- MENU
Menu orang program yeah

1. Tampilkan list
2. Tambah orang
3. Tambah orang di urutan
4. Ganti orang
5. Inisialisasi list orang (template)
6. Hapus orang
7. Exit

Pilih [1-7] :
```

Deskripsi program

Program di atas merupakan sebuah aplikasi sederhana untuk mengelola data individu, yang dapat menampilkan daftar individu, menambahkan individu, menambahkan individu pada posisi tertentu dalam daftar, mengubah data individu, menghapus individu dari daftar, dan menginisialisasi daftar dengan data template. Program ini menggunakan struktur data linked list untuk menyimpan dan mengelola data individu, dengan setiap node dalam linked list menyimpan informasi seperti nama dan usia seseorang.

Dalam struktur linked list, setiap node direpresentasikan oleh sebuah struktur Node yang memiliki dua anggota data, yaitu nama_anda yang menyimpan nama individu dan usia_anda yang menyimpan usia individu,

serta sebuah pointer next yang menunjuk ke node berikutnya dalam linked list.

Fungsi-fungsi yang didefinisikan dalam program ini antara lain:

1. `addNode`: Menambahkan node baru ke dalam linked list, baik di awal, di akhir, atau di posisi tertentu sesuai dengan input pengguna.
2. `initList`: Menginisialisasi linked list dengan data template, yaitu sejumlah nama dan usia yang telah ditentukan sebelumnya.
3. `printNodes`: Menampilkan semua node dalam linked list, yaitu nama dan usia setiap individu.
4. `deleteNode`: Menghapus node dari linked list berdasarkan nama individu yang dipilih pengguna.
5. `changeNode`: Mengubah data (nama dan usia) dari sebuah node dalam linked list berdasarkan nama individu yang dipilih pengguna.
6. `addNodeSpecified`: Menambahkan node baru ke dalam linked list pada posisi tertentu sesuai dengan input pengguna.
7. `main`: Merupakan titik masuk utama program, yang berisi loop utama untuk menampilkan menu dan menerima input dari pengguna untuk menjalankan operasi yang diinginkan.

2. Unguided 2

Source code

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

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

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

    void push(double 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;
        delete temp;
    }

    bool update(int oldData, int newData, string oldKata, string
newKata) {
        Node* current = head;
        while (current != nullptr) {
```

```

        if (current->data == oldData && current->kata ==
oldKata) {
            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 deleteSpecific(string name) {
    Node* current = head;

    /*
    jika current itu ada & kata selanjutnya dan sekarang
    BUKAN kata yang dicari
    maka inkrementasi.
    */
    while (current != nullptr && (current->next->kata != name
&& current->kata != name)) {
        current = current->next;
    }

    if (current->prev != nullptr) {
        current->prev->next = current->next;
    } else {
        // Jika yg dihapus kepala
        head = current->next;
    }

    if (current->next != nullptr) {
        current->next->prev = current->prev;
    } else {
        // Jika yg dihapus buntut
        tail = current->prev;
    }
    delete current;
}

void display() {

```

```

        Node* current = head;
        cout << left << setw(20) << "Nama Produk" << "Harga" <<
endl;
        while (current != nullptr) {
            cout << left << setw(20) << current->kata;
            cout << fixed << setprecision(3) << current->data <<
endl;
            current = current->next;
        }
    }

    void pushSpecific(double data, string kata, string cari) {
        Node *newNode = new Node;
        Node *current = head;
        newNode->data = data;
        newNode->kata = kata;

        while (cari != current->kata) {
            current = current->next;
        }
        newNode->next = current->next;
        current->prev = current->prev;
        current->next = newNode;
    }
};

int main() {
    DoublyLinkedList list;

    list.push(60.000, "Originote");
    list.push(150.000, "Somethinc");
    list.push(100.000, "Skintific");
    list.push(50.000, "Wardah");
    list.push(30.000, "Hanasui");

    while (true) {
        cout << "\n1. 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;
        cout << endl;

        switch (choice) {

```

```

        case 1: {
            double data;
            string kata;
            cout << "Enter data to add (double): ";
            cin >> data;
            cin.ignore();
            cout << "Enter kata to add: ";
            getline(cin, kata);
            list.push(data, kata);
            break;
        }
        case 2: {
            list.pop();
            break;
        }
        case 3: {
            list.display();

            double oldData, newData;
            string oldKata, newKata;
            cout << "\nEnter old data (double): ";
            cin >> oldData;
            cout << "Enter new data (double): ";
            cin >> newData;
            cin.ignore();
            cout << "Enter old kata: ";
            getline(cin, oldKata);
            cout << "Enter new kata: ";
            getline(cin, newKata);
            bool updated = list.update(oldData, newData,
oldKata ,newKata);
            if (!updated) { cout << "Data not found" << endl;
}

            break;
        }
        case 4: {
            double data;
            string kata;
            cout << "Enter data to add (double): ";
            cin >> data;
            cin.ignore();
            cout << "Enter kata to add: ";
            getline(cin, kata);

            list.display();

            string cekThis;
            cout << "\nAdd after what? : ";
            getline(cin, cekThis);
            list.pushSpecific(data, kata, cekThis);
            break;
        }
    }
}

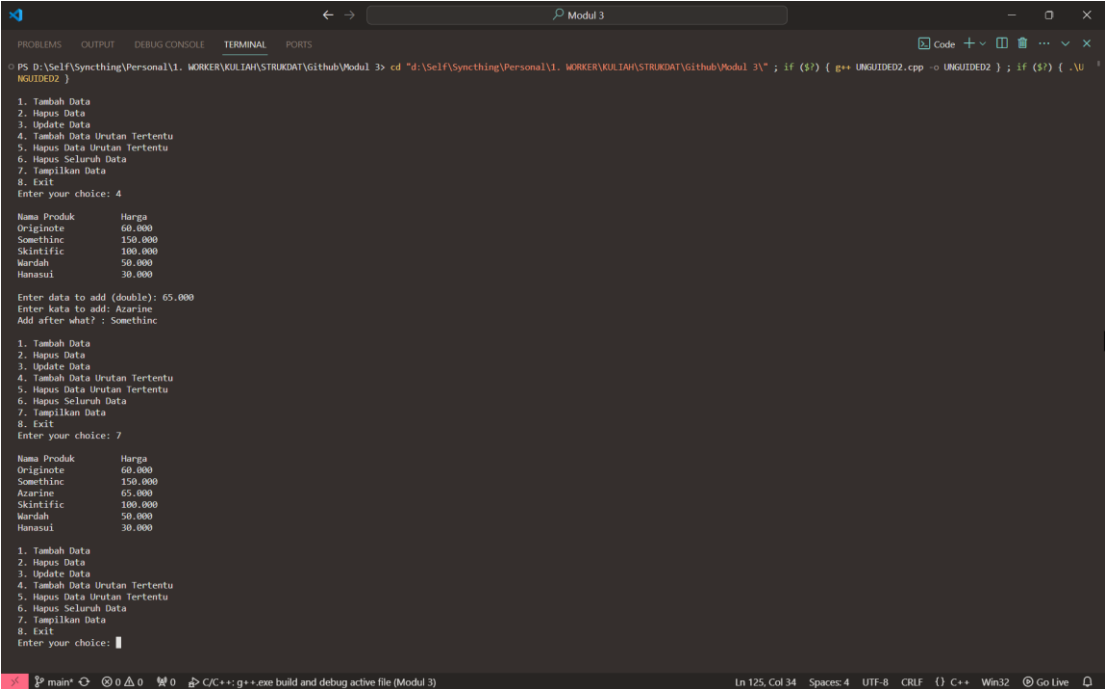
```

```
        case 5: {
            list.display();

            string delThis;
            cin.ignore();
            cout << "\nDelete what? : ";
            getline(cin, delThis);
            list.deleteSpecific(delThis);
            break;
        }
        case 6: {
            list.deleteAll();
            break;
        }
        case 7: {
            list.display();
            break;
        }
        case 8: {
            return 0;
        }
        default: {
            cout << "Invalid choice" << endl;
            break;
        }
    }
}
return 0;
}
```

Screenshoot program

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



```
PS D:\Self\Syncthing\Personal\1. WORKER\KULIAH\STRUKDAT\Github\Modul 3> cd "d:\Self\Syncthing\Personal\1. WORKER\KULIAH\STRUKDAT\Github\Modul 3\" ; if ($?) { g++ UNGUIDED2.cpp -o UNGUIDED2 } ; if ($?) { .\UNGUIDED2 }

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

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

Enter data to add (double): 65.000
Enter kata to add: Azarine
Add after what? : Somethinc

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

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. Hapus produk wardah.

```
Modul 3
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Originote 60.000
Something 150.000
Azarine 65.000
Skintific 100.000
Wardah 50.000
Hanasul 30.000

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

Nama Produk Harga
Originote 60.000
Something 150.000
Azarine 65.000
Skintific 100.000
Wardah 50.000
Hanasul 30.000

Delete what? : Wardah

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
Something 150.000
Azarine 65.000
Skintific 100.000
Hanasul 30.000

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: 1

main* C/C++ g++ build and debug active file (Modul 3) Ln 125, Col 34 Spaces: 4 UTF-8 CRLF C++ Win32 Go Live
```

3. Update produk Hanasui menjadi Cleora dengan harga 55.000

```
Modul 3

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Something 150.000
Azarine 65.000
Skintific 100.000
Hanasui 30.000

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

Nama Produk Harga
Originote 60.000
Something 150.000
Azarine 65.000
Skintific 100.000
Hanasui 30.000

Enter old data (double): 30.000
Enter new data (double): 55.000
Enter old kata: Hanasui
Enter new kata: Cleora

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
Something 150.000
Azarine 65.000
Skintific 100.000
Cleora 55.000

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: 1
```

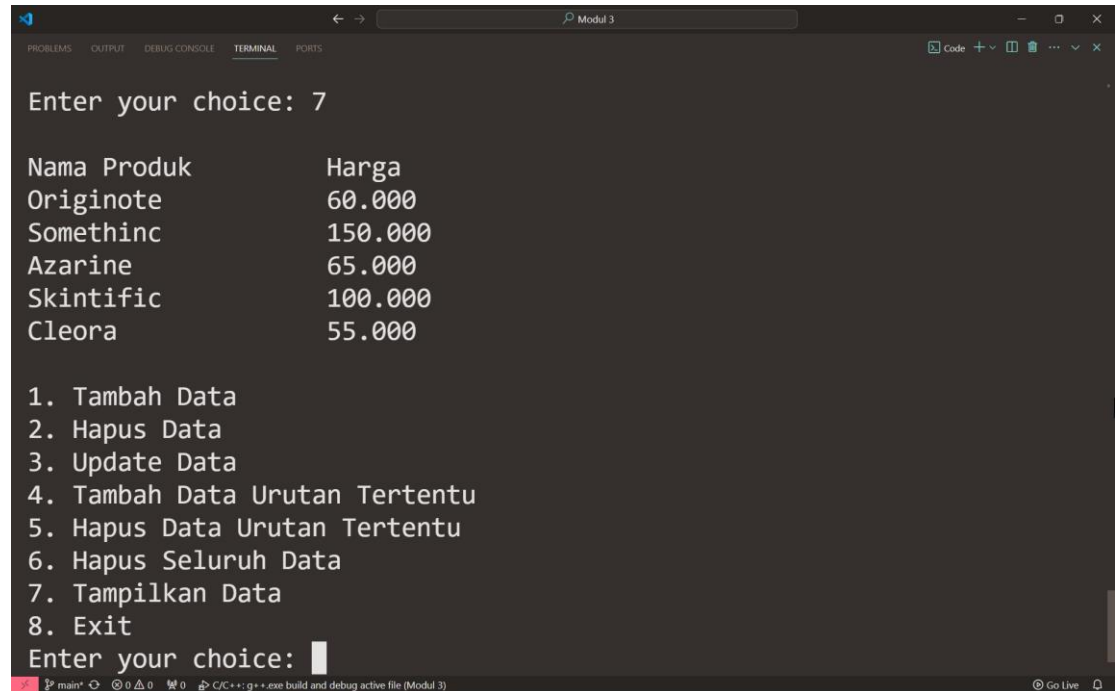
4. Tampilkan menu seperti dibawah ini.

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



```
Enter your choice: 7

Nama Produk      Harga
Originote        60.000
Somethinc         150.000
Azarine           65.000
Skintific         100.000
Cleora            55.000

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: █
```

Deskripsi program

Program yang diberikan adalah sebuah aplikasi yang mengimplementasikan Doubly Linked List menggunakan bahasa pemrograman C++. Doubly Linked List adalah struktur data berurutan di mana setiap elemen (node) memiliki dua pointer, yaitu satu yang menunjuk ke node sebelumnya (prev) dan satu lagi yang menunjuk ke node berikutnya (next).

Berikut adalah penjelasan mengenai fungsi-fungsi utama dalam program beserta menu yang disediakan:

1. Konstruktor DoublyLinkedList()

Konstruktor ini digunakan untuk menginisialisasi objek DoublyLinkedList dengan menetapkan pointer head dan tail menjadi nullptr, menandakan bahwa linked list masih kosong.

2. Fungsi push(int data, string kata)

Fungsi ini digunakan untuk menambahkan node baru ke depan linked list. Parameter data dan kata digunakan untuk menentukan nilai data dan kata yang akan disimpan dalam node baru.

Node baru akan ditambahkan di depan linked list dan menjadi elemen pertama dengan cara mengubah pointer head sehingga menunjuk ke node baru.

3. Fungsi pop()

Fungsi ini digunakan untuk menghapus node pertama dari linked list.

Jika linked list tidak kosong, node pertama (disebut head) akan dihapus, dan head akan diubah untuk menunjuk ke node berikutnya. Jika linked list hanya memiliki satu node, tail akan diatur menjadi nullptr.

4. Fungsi update(int oldData, int newData, string oldKata, string newKata)

Fungsi ini digunakan untuk mengubah nilai data dan kata pada node tertentu dalam linked list.

Pengguna harus memberikan nilai oldData dan oldKata yang akan diubah, serta nilai newData dan newKata yang akan menggantikannya.

Fungsi akan mencari node yang memiliki nilai data dan kata sesuai dengan nilai oldData dan oldKata, dan kemudian menggantinya dengan nilai newData dan newKata.

5. Fungsi deleteAll()

Fungsi ini digunakan untuk menghapus seluruh node dalam linked list.

Fungsi akan menghapus setiap node satu per satu, dimulai dari head hingga tail, dan kemudian mengatur head dan tail menjadi nullptr.

6. Fungsi display()

Fungsi ini digunakan untuk menampilkan isi linked list.

Fungsi akan menampilkan nilai data dan kata dari setiap node dalam linked list secara berurutan, dimulai dari head hingga tail.

7. Fungsi main()

Fungsi main() merupakan titik masuk utama program.

Program akan menampilkan menu untuk pengguna yang terdiri dari beberapa pilihan operasi: menambah data, menghapus data, mengubah data, membersihkan data, menampilkan data, dan keluar dari program.

Setiap pilihan menu akan memanggil fungsi yang sesuai dari objek DoublyLinkedList untuk menjalankan operasi tersebut.

BAB IV

KESIMPULAN

Dari uraian tersebut, dapat disimpulkan beberapa poin terkait Linked List, khususnya Single Linked List dan Double Linked List:

1. Linked List adalah struktur data dinamis yang terdiri dari kumpulan elemen yang disebut node, di mana setiap node memiliki dua bagian: bagian data yang menyimpan nilai, dan bagian pointer yang menunjukkan ke node berikutnya.
2. Single Linked List Merupakan jenis Linked List di mana setiap node hanya memiliki satu pointer yang menunjuk ke node berikutnya.
3. Circular Linked List Mirip dengan Single Linked List, namun penunjuk next pada node terakhir selalu merujuk ke node pertama. Ini memungkinkan akses ke seluruh elemen dalam daftar melalui loop.
4. Double Linked List adalah jenis Linked List di mana setiap node memiliki dua pointer, yaitu pointer next yang menunjuk ke node berikutnya, dan pointer prev yang menunjuk ke node sebelumnya. Ini memungkinkan operasi penghapusan dan penambahan pada simpul mana saja secara efisien, serta traversal dari depan (head) maupun belakang (tail).
5. Keuntungan Double Linked List Memungkinkan operasi penghapusan dan penambahan pada simpul mana saja dengan efisien, serta traversal dari depan dan belakang dengan mudah.
6. Kekurangan Double Linked List Menggunakan memori lebih besar dan membutuhkan waktu eksekusi lebih lama dibandingkan dengan Single Linked List.

Dengan demikian, Linked List merupakan struktur data yang sangat berguna dalam menyimpan dan mengelola data secara dinamis, dengan masing-masing jenis memiliki karakteristik dan kegunaan yang berbeda

DAFTAR PUSTAKA

- [1] Learning Management System ,MODUL I PENGENALAN GERBANG LOGIKA DASAR.
- [2] muh—agha—zul, *「PRAK」 Single and Double Linked List*, 2023.
<https://www.youtube.com/watch?v=-LHppqxjqRw>
- [3] muh—agha—zul, *「PRAK」 Single and Double Linked List 2*, 2023.
<https://www.youtube.com/watch?v=hKGsduhyImc>