

**LAPORAN PRAKTIKUM
STRUKTUR DATA PEMOGRAMAN**

**MODUL III
“SINGLE AND DOUBLE LINKED LIST”**



Disusun Oleh:

NAMA : Trie Nabilla Farhah

NIM : 2311102071

Dosen Pengampu:

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

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

2024

A. DASAR TEORI

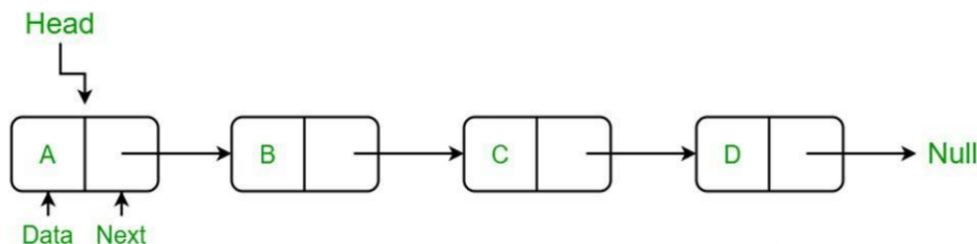
Linked List adalah suatu cara untuk menyimpan data dengan struktur sehingga programmer dapat secara otomatis menciptakan suatu tempat baru untuk menyimpan data kapan saja diperlukan. Linked list dikenal juga dengan sebutan senarai berantai adalah stuktur data yang terdiri dari urutan record data dimana setiap record memiliki field yang menyimpan alamat/referensi dari record selanjutnya (dalam urutan). Elemen data yang dihubungkan dengan link pada linked list disebut Node. Biasanya dalam suatu linked list, terdapat istilah head dan tail .

- Head adalah elemen yang berada pada posisi pertama dalam suatu linked list
- Tail adalah elemen yang berada pada posisi terakhir dalam suatu linked list.

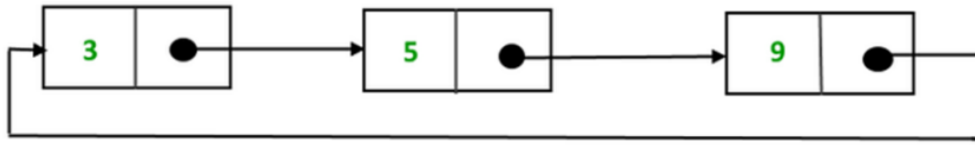
A. Single Linked List

Single Linked List adalah sebuah field pointer-nya hanya satu buah saja dan satu arah serta pada akhir node yang nodenya saling terhubung satu sama lain. Jadi Setiap node pada linked list mempunyai field yang berisi pointer ke node berikutnya, dan juga memiliki field yang berisi data. Node terakhir akan menunjuk ke NULL yang akan digunakan sebagai kondisi berhenti pada saat pembacaan isi linked list.

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.



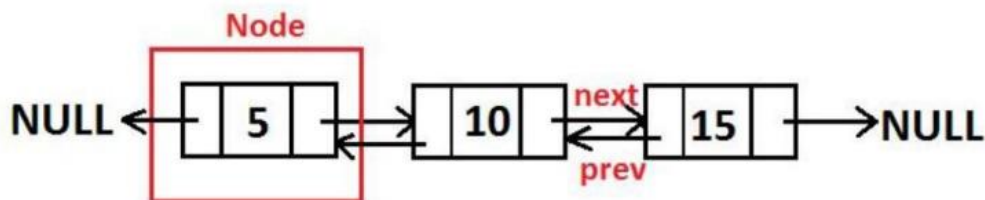
Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Single linked list yang kedua adalah circular linked list. Perbedaan circular linked list dan non circular linked adalah penunjuk next pada node terakhir pada circular linked list akan selalu merujuk ke node pertama.



B. Double Linked List

Tipe data koleksi (Collection Data Type) adalah tipe data yang digunakan untuk mengelompokkan dDouble 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. 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 menyimpan beberapa nilai atau objek secara bersamaan.

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.



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

B. GUIDED

GUIDED 1 : Latihan Single Linked List

Source Code

```
#include <iostream>
using namespace std;
///PROGRAM SINGLE LINKED LIST NON-CIRCULAR
//Deklarasi Struct Node
struct Node{
    int data;
    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){
    //Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true){
        head = tail = baru;
        tail->next = NULL;
    }
    else{
        baru->next = head;
        head = baru;
    }
}
```

```

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

//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, int posisi){
    if( posisi < 1 || posisi > hitungList() ){
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if( posisi == 1){
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else{
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;

```

```

        // tranversing
        bantu = head;
        int nomor = 1;
        while( nomor < posisi - 1 ){
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
//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){
if (isEmpty() == false){
head->data = data;

```

```

    }
    else{
        cout << "List masih kosong!" << endl;
    }
}

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

//Ubah Belakang
void ubahBelakang(int data){
    if (isEmpty() == false){
        tail->data = data;
    }
    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 << ends;
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();

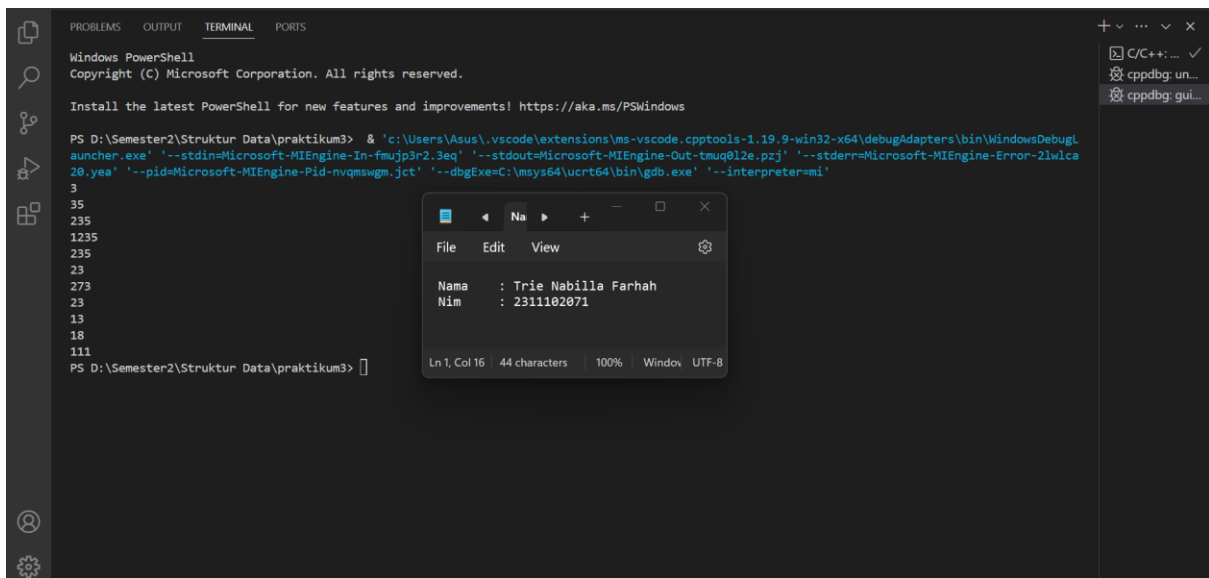
```

```

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

```

Output



```

Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\Semester2\Struktur Data\praktikum3> & 'c:\Users\Asus\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugL
auncher.exe' '--stdin=Microsoft-MIEngine-In-fmu3p3r2.3eq' '--stdout=Microsoft-MIEngine-Out-tmuq012e.pzj' '--stderr=Microsoft-MIEngine-Error-2lw1ca
20.yea' '--pid=Microsoft-MIEngine-Pid-nvqmswgm.jct' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
3
35
235
1235
235
23
273
23
13
18
111
PS D:\Semester2\Struktur Data\praktikum3>

```

Deskripsi Program

Program Single Linked List Non-Circular merupakan implementasi dari Single Linked List, yaitu representasi data dengan struktur node yang hanya memiliki pointer ke node berikutnya. Struktur dari node terdiri atribut data dan next. Fungsi yang tersedia pada program meliputi init untuk inisialisasi head dan tail menjadi NULL, isEmpty untuk mengecek apakah linked list kosong atau tidak, insertDepan untuk menambahkan node baru di awal linked list, insertBelakang untuk menambahkan node baru di akhir linked list, hitungList untuk menghitung jumlah node pada linked list, insertTengah untuk menambahkan node baru pada posisi tertentu pada linked list, hapusDepan untuk menghapus node pertama pada linked list, hapusBelakang untuk menghapus node terakhir pada linked list, hapusTengah untuk menghapus node pada posisi tertentu pada linked list, ubahDepan untuk mengubah data pada node pertama pada linked list, ubahTengah untuk mengubah data pada node pada posisi tertentu pada linked list, ubahBelakang untuk mengubah data pada node terakhir pada linked list, clearList untuk menghapus semua node pada linked list, dan tampil untuk menampilkan data pada semua node pada linked list.

Dalam main program, dilakukan beberapa operasi pada linked list, yaitu menambahkan data, menghapus data, dan menampilkan data pada linked list. Selain itu, program juga menambahkan fungsi hitungList untuk menghitung jumlah node pada linked list, dan insertTengah untuk menambahkan data pada posisi tertentu pada linked list.

GUIDED 2 : Latihan Double Link List

Source Code

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

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

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

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

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

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

```

        head = newNode;
    }

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

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

    delete temp;
}

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

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

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

```

```

    }

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

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

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

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

```

Output

```

PROBLEMS OUTPUT TERMINAL PORTS
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: 12 10
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: Invalid choice
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: 3
Enter old data: 12
Enter new data: 14
Data not found
1. Add data

```

Na

File Edit View

Nama : Trie Nabilla Farhah
Nim : 2311102071

Ln 1, Col 16 44 characters 100% Window UTF-8

C/C++: ... X

cppdbg: un...
cppdbg: gui...
cppdbg: gui...

Deskripsi Program

Program ini adalah representasi Double Linked List (DLL) yang merupakan struktur data yang memiliki dua node, yaitu node sebelumnya dan node selanjutnya. Setiap node terdiri dari tiga atribut, yaitu data, prev, dan next. DLL memiliki dua pointer, yaitu head dan tail. Pointer head menunjuk ke elemen pertama pada DLL, sedangkan tail menunjuk ke elemen terakhir pada DLL. Di dalam program ini terdapat beberapa fungsi, yaitu push, pop, update, deleteAll, dan display. Fungsi push digunakan untuk menambahkan data baru pada depan DLL, fungsi pop digunakan untuk menghapus data pada depan DLL, fungsi update digunakan untuk mengganti data pada elemen tertentu, fungsi deleteAll digunakan untuk menghapus semua data pada DLL, dan fungsi display digunakan untuk menampilkan data yang ada pada DLL.

Di dalam main program, terdapat perulangan while yang akan melakukan operasi sesuai dengan pilihan yang dipilih oleh pengguna. Jika pengguna memilih "1", maka akan diminta untuk memasukkan data baru. Jika pengguna memilih "2", maka akan dihapus data pada depan DLL. Jika pengguna memilih "3", maka akan diganti data pada elemen tertentu sesuai dengan data yang dimasukkan pengguna. Jika pengguna memilih "4", maka akan dihapus semua data pada DLL. Jika pengguna memilih "5", maka akan ditampilkan data pada DLL. Jika pengguna memilih "6", maka program akan berhenti. Selain itu, terdapat beberapa validasi pada program ini, yaitu jika pengguna memilih "2", maka hanya akan dihapus data pada depan DLL jika DLL tidak kosong. Jika pengguna memilih "3", maka akan diupdate data pada elemen tertentu jika elemen tersebut ditemukan dalam DLL. Jika pengguna memilih "4", maka akan dihapus semua data pada DLL. Hal ini bertujuan untuk menghindari error pada program.

C. UNGUIDED

Unguided 1.

Buatlah program menu Single Linked List Non-Circular untuk menyimpan Nama dan usia mahasiswa, dengan menggunakan inputan dari user. Lakukan operasi berikut.

- a. Masukkan data sesuai urutan berikut. (Gunakan insert depan, belakang atau tengah).
Data pertama yang dimasukkan adalah nama dan usia anda.
- b. Hapus data Akechi
- c. Tambahkan data berikut diantara John dan Jane
- d. Tambahkan data berikut diawal
- e. Ubah data Michael menjadi
- f. Tampilkan seluruh data

Source Code

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

struct node
{
    string name;
    int age;
    node* next;
};

class linked_list
{
private:
    node* head, * tail;

public:
    linked_list()
    {
        head = NULL;
        tail = NULL;
    }

    // tambah awal
    void add_first(string name, int age)
    {
```



```

    node* temp = new node;
    temp->name = name;
    temp->age = age;
    temp->next = head;
    head = temp;
    if (tail == NULL)
    {
        tail = temp;
    }
}

// tambah akhir
void add_last(string name, int age)
{
    node* temp = new node;
    temp->name = name;
    temp->age = age;
    temp->next = NULL;
    if (tail == NULL)
    {
        head = temp;
        tail = temp;
    }
    else
    {
        tail->next = temp;
        tail = temp;
    }
}

// tambah tengah
void add_middle(int pos, string name, int age)
{
    node* prev = new node;
    node* cur = new node;
    node* temp = new node;
    cur = head;
    for (int i = 1; i < pos; i++)
    {
        prev = cur;
        cur = cur->next;
    }
}

```

```

        temp->name = name;
        temp->age = age;
        prev->next = temp;
        temp->next = cur;
    }

// ubah tengah
void edit_middle(int pos, string name, int age)
{
    node* temp = new node;
    temp = head;
    for (int i = 1; i < pos; i++)
    {
        temp = temp->next;
    }
    temp->name = name;
    temp->age = age;
}

// tampilkan jumlah data
void count_data()
{
    int count = 0;
    node* temp = new node;
    temp = head;
    while (temp != NULL)
    {
        count++;
        temp = temp->next;
    }
    cout << "Jumlah Data: " << count << endl;
}

// hapus data
void delete_data(int pos)
{
    node* prev = new node;
    node* cur = new node;
    cur = head;
    for (int i = 1; i < pos; i++)
    {

```

```

        prev = cur;
        cur = cur->next;
    }
    prev->next = cur->next;
}

// tampilkan data
void display()
{
    node* temp = new node;
    temp = head;
    while (temp != NULL)
    {
        cout << "Nama: " << temp->name << ", Usia: "
<< temp->age
        << endl;
        temp = temp->next;
    }
}

};

int main()
{
    linked_list l;

    // input data
    int choice;
    string name;
    int age;
    int pos;

    do {
        cout << endl;
        cout << "Menu Utama: " << endl;
        cout << "1. Tambahkan Awal" << endl;
        cout << "2. Tambahkan Akhir" << endl;
        cout << "3. Tambahkan Tengah" << endl;
        cout << "4. Ubah Tengah" << endl;
        cout << "5. Hapus Data" << endl;
        cout << "6. Tampilkan Jumlah Data" << endl;
    }

```

```
cout << "7. Tampilkan Data" << endl;
cout << "8. Keluar" << endl;
cout << "Pilih Menu: ";
cin >> choice;

switch (choice)
{
case 1:
    cout << "Masukkan Nama: ";
    cin >> name;
    cout << "Masukkan Usia: ";
    cin >> age;
    l.add_first(name, age);
    break;
case 2:
    cout << "Masukkan Nama: ";
    cin >> name;
    cout << "Masukkan Usia: ";
    cin >> age;
    l.add_last(name, age);
    break;
case 3:
    cout << "Masukkan Posisi: ";
    cin >> pos;
    cout << "Masukkan Nama: ";
    cin >> name;
    cout << "Masukkan Usia: ";
    cin >> age;
    l.add_middle(pos, name, age);
    break;
case 4:
    cout << "Masukkan Posisi: ";
    cin >> pos;
    cout << "Masukkan Nama: ";
    cin >> name;
    cout << "Masukkan Usia: ";
    cin >> age;
    l.edit_middle(pos, name, age);
break;
case 5:
    cout << "Masukkan Posisi: ";
```

```

        cin >> pos;
        l.delete_data(pos);
        break;
    case 6:
        l.count_data();
        break;
    case 7:
        l.display();
        break;
    case 8:
        cout << "Program Selesai" << endl;
        break;
    default:
        cout << "Pilihan tidak tersedia" << endl;
        break;
    }
} while (choice != 8);
return 0;
}

```

Output

The screenshot shows a C++ IDE with a dark theme. On the left, the 'TERMINAL' tab is active, displaying the program's output. The output shows a menu with 8 options. The user has selected option 1, entered 'nabilla' for the name and '18' for the age. Then, they selected option 2, entered 'karina' for the name and '20' for the age. The program then displays the menu again, showing only the first 5 options. In the center, a small window titled 'Na' is open, showing the entered data: 'Nama : Trie Nabilla Farhah' and 'Nim : 2311102071'. The status bar at the bottom of the window indicates 'Ln 2, Col 17', '44 characters', '100%', 'Window', and 'UTF-8'.

```

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 1
Masukkan Nama: nabilla
Masukkan Usia: 18

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data

```

Na

Nama : Trie Nabilla Farhah
Nim : 2311102071

Ln 2, Col 17 44 characters 100% Window UTF-8

PROBLEMS OUTPUT **TERMINAL** PORTS

6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 2
Masukkan Nama: tanaka
Masukkan Usia: 19

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 2
Masukkan Nama: yui
Masukkan Usia: 5

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 2
Masukkan Nama: lima
Masukkan Usia: 26

File Edit View

Nama : Trie Nabilla Farhah
Nim : 2311102071

Ln 2, Col 17 44 characters 100% Window UTF-8

PROBLEMS OUTPUT **TERMINAL** PORTS

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 2
Masukkan Nama: leah
Masukkan Usia: 27

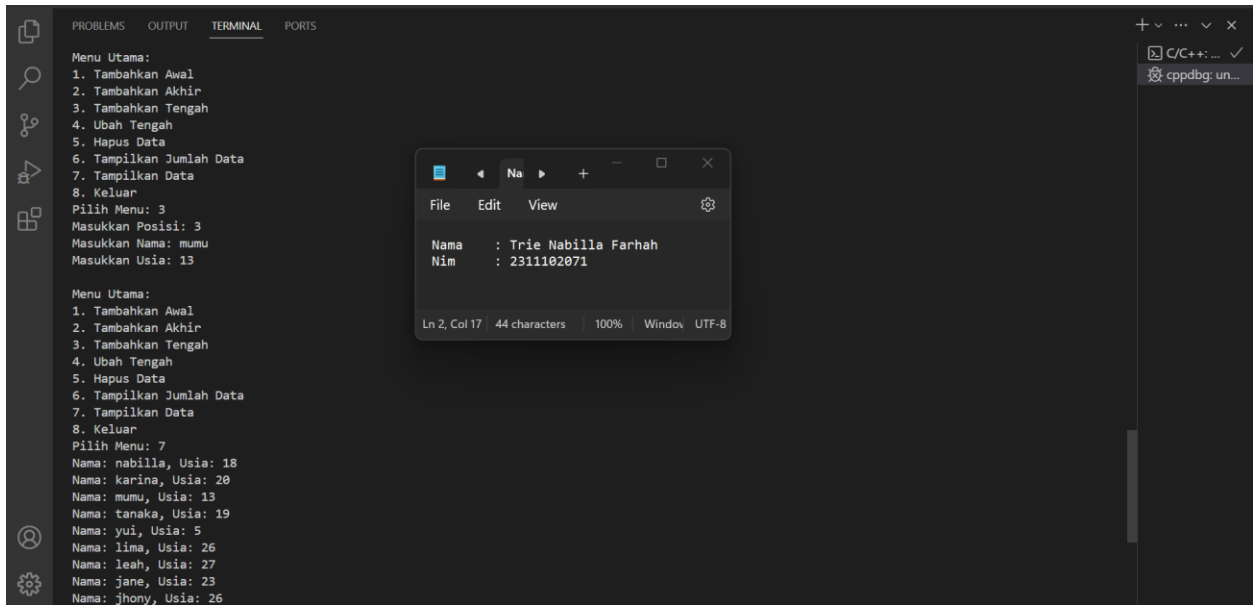
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 2
Masukkan Nama: jane
Masukkan Usia: 23

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data

File Edit View

Nama : Trie Nabilla Farhah
Nim : 2311102071

Ln 2, Col 17 44 characters 100% Window UTF-8



```
Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 3
Masukkan Posisi: 3
Masukkan Nama: mumu
Masukkan Usia: 13

Menu Utama:
1. Tambahkan Awal
2. Tambahkan Akhir
3. Tambahkan Tengah
4. Ubah Tengah
5. Hapus Data
6. Tampilkan Jumlah Data
7. Tampilkan Data
8. Keluar
Pilih Menu: 7
Nama: nabilla, Usia: 18
Nama: karina, Usia: 20
Nama: mumu, Usia: 13
Nama: tanaka, Usia: 19
Nama: yui, Usia: 5
Nama: lina, Usia: 26
Nama: leah, Usia: 27
Nama: jane, Usia: 23
Nama: jhony, Usia: 26
```

Deskripsi Program

Program ini adalah representasi dari linked list non-circular yang digunakan untuk menyimpan data nama dan usia mahasiswa. Dalam program ini, terdapat class `linked_list` yang memiliki beberapa method, yaitu `add_first`, `add_last`, `add_middle`, `edit_middle`, `count_data`, `delete_data`, dan `display`. Method `add_first` digunakan untuk menambah data di awal linked list, sedangkan `add_last` digunakan untuk menambah data di akhir linked list. Untuk menambah data pada posisi tertentu, digunakan method `add_middle`. Untuk mengubah data pada posisi tertentu, digunakan method `edit_middle`. Method `count_data` digunakan untuk menghitung jumlah data yang ada pada linked list, sedangkan method `delete_data` digunakan untuk menghapus data pada posisi tertentu. Untuk menampilkan seluruh data pada linked list, digunakan method `display`.

Dalam main program, terdapat looping `do-while` yang akan mengulangi menu utama hingga pengguna memilih untuk keluar. Pada setiap pilihan, akan dipanggil method yang sesuai dari class `linked_list`. Di dalam pengisian data, pengguna diminta untuk memasukkan nama dan usia mahasiswa. Selain itu, di dalam program ini juga terdapat validasi untuk memastikan bahwa posisi yang dimasukkan oleh pengguna valid, yaitu antara 1 sampai jumlah data yang ada pada linked list. Jika posisi yang dimasukkan pengguna tidak valid, maka akan muncul pesan error "Posisi tidak valid!".

Unguided 2

Modifikasi Guided Double Linked list dilakukan dengan penambahan operasi untuk menambah data, menghapus, dan update di tengah atau di urutan yang diminta. Selain itu, buatlah agar tampilannya menampilkan nama produk dan harga.

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 :

- a) Tambah Data
- b) Hapus Data
- c) Update Data
- d) Tambah Data Urutan Tertentu
- e) Hapus Data Urutan Tertentu
- f) Hapus Seluruh Data
- g) Tampilkan Data
- h) Exi

Source Code

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

class Node {
public:

    string product_name;
    float price;
    Node* prev;
    Node* next;
};

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

    DoublyLinkedList() {
        head = nullptr;
```



```

        tail = nullptr;
    }
    void push(string product_name, float price) {
        Node* newNode = new Node;
        newNode->product_name = product_name;
        newNode->price = price;
        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(string oldProduct, string newProduct,
float newPrice)
    {
        Node* current = head;
        while (current != nullptr) {
            if (current->product_name == oldProduct)
            {
                current->product_name = newProduct;
                current->price = newPrice;
                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->product_name << " (Rp"
<< current->price
        << ")" << endl;
        current = current->next;
    }
}

void insert(string product_name, float price, int
position) {
    if (position <= 0) {
        push(product_name, price);
        return;
    }
    Node* current = head;
    for (int i = 1; i < position && current !=
nullptr; i++) {
        current = current->next;
    }

    if (current == nullptr) {
        Node* newNode = new Node;
        newNode->product_name = product_name;

```

```

        newNode->price = price;
        newNode->prev = tail;
        newNode->next = nullptr;
        if (tail != nullptr) {
            tail->next = newNode;
        } else {
            head = newNode;
        }
        tail = newNode;
    } else {
        Node* newNode = new Node;
        newNode->product_name = product_name;
        newNode->price = price;
        newNode->prev = current->prev;
        newNode->next = current;
        if (current->prev != nullptr) {
            current->prev->next = newNode;
        } else {
            head = newNode;
        }
        current->prev = newNode;
    }
}

void remove(int position) {
    if (position <= 0) {
        pop();
        return;
    }
    Node* current = head;
    for (int i = 1; i < position && current !=
nullptr; i++) {
        current = current->next;
    }
    if (current == nullptr) {
        return;
    }
    if (current == head) {
        head = current->next;
    } else {
        current->prev->next = current->next;
    }
}

```

```

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

};

int main()
{
    DoublyLinkedList list;
    int choice;
    string productName, newProductName;
    float price, newPrice;
    int position;
    do {
        cout << "1. tambah data\n";
        cout << "2. hapus data\n";
        cout << "3. Update data \n";
        cout << "4. tambah data urutan tertentu
insert\n";
        cout << "5. hapus data urutan tertentu
remove\n";
        cout << "6. hapus seluruh data\n";
        cout << "7. tampilkan data\n";
        cout << "8. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "masukan nama produk: ";
                cin >> productName;
                cout << "masukan harga: ";
                cin >> price;
                list.push(productName, price);
                break;
            case 2:
                list.pop();

```

```

        break;
    case 3:
        cout << "masukan nama produk yang lama:
";
        cin >> productName;
        cout << "masukan nama produk yang baru:
";
        cin >> newProductName;
        cout << "masukan harga baru: ";
        cin >> newPrice;
        if (list.update(productName,
newProductName, newPrice))
        {
            cout << "Produk berhasil ditambahkan\n";
        } else {
            cout << "produk tidak ada\n";
        }
        break;
    case 4:
        cout << "masukan nama produk: ";
        cin >> productName;
        cout << "masukan harga: ";
        cin >> price;
        cout << "masukan posisi: ";
        cin >> position;
        list.insert(productName, price,
position);
        break;
    case 5:
        cout << "masukan urutan ke-: ";
        cin >> position;
        list.remove(position);
        break;
    case 6:
        list.deleteAll();
        cout << "semua produk berhasil di
hapus\n";
        break;
    case 7:
        list.display();
        break;

```

```

        case 8:
            cout << "keluar dari progam...\n";
            break;
        default:
            cout << "pilihan salah\n";
            break;
    }
} while (choice != 8);

return 0;
}

```

Output

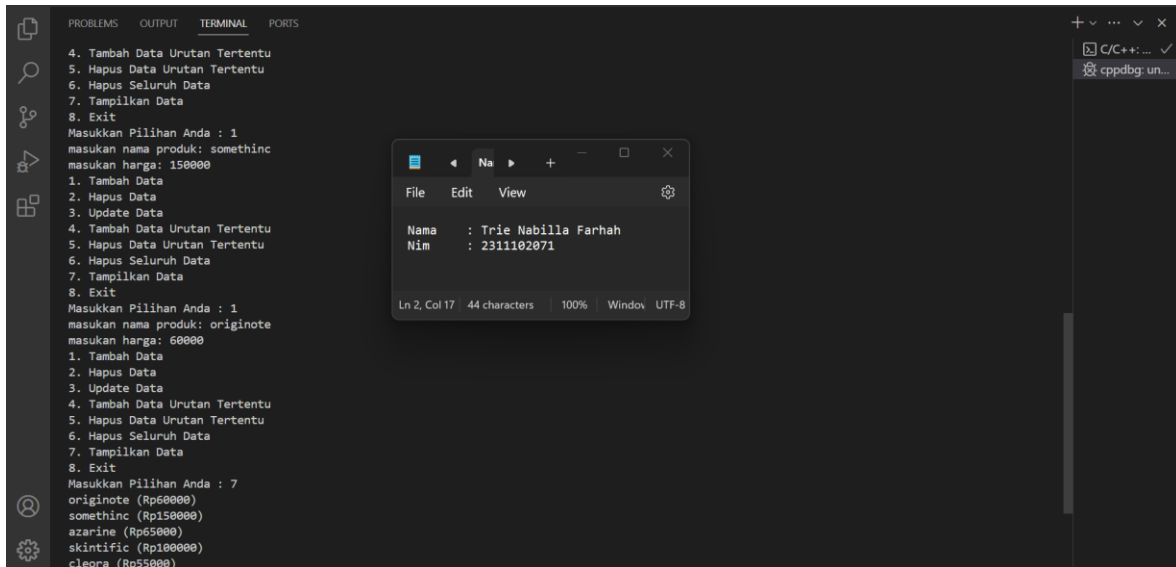
The image displays two screenshots of a C++ IDE, likely Visual Studio Code, showing the output of a program and a data entry window.

Top Screenshot: The terminal window shows the program's output, which is a menu of options for a database system. The user has selected option 1, "Tambah Data". The program prompts the user to enter a product name and price. The user enters "skintific" and "100000". The program then displays the menu again.

Bottom Screenshot: The terminal window shows the program's output, which is the same menu of options. The user has selected option 1, "Tambah Data". The program prompts the user to enter a product name and price. The user enters "something" and "150000". The program then displays the menu again.

Data Entry Window: A small window titled "Na" is shown in the center of the IDE. It contains a table with two columns: "Nama" and "Nim". The data entered by the user is displayed in the table:

Nama	Nim
skintific	2311102071



Deskripsi Program

Program ini adalah implementasi dari Doubly Linked List (DLL) yang digunakan untuk menyimpan data nama produk dan harga. Di dalam program ini terdapat class DoublyLinkedList yang memiliki head dan tail sebagai titik awal dan akhir dari DLL. Class ini juga memiliki beberapa method, yaitu push, pop, update, insert, remove, deleteAll, dan display.

Method push digunakan untuk menambah data baru di awal DLL, sedangkan pop digunakan untuk menghapus data pertama di DLL. Method update digunakan untuk mengubah data pada DLL dengan mencocokkan nama produk lama dan menginputkan nama produk baru beserta harga barunya. Method insert digunakan untuk menambah data baru pada posisi tertentu di DLL, sedangkan method remove digunakan untuk menghapus data pada posisi tertentu di DLL. Method deleteAll digunakan untuk menghapus semua data pada DLL. Selain itu, method display digunakan untuk menampilkan seluruh data yang ada pada DLL.

Di dalam main program, terdapat looping do-while yang akan mengulangi menu utama hingga pengguna memilih untuk keluar. Pada setiap pilihan, akan dipanggil method yang sesuai dari class DoublyLinkedList.

Dalam pengisian data, user diminta untuk memasukkan nama produk dan harga. Selain itu, di dalam program ini juga terdapat validasi untuk memastikan bahwa posisi yang dimasukkan oleh user valid, yaitu antara 1 sampai jumlah data yang ada pada DLL.

D. KESIMPULAN

Setelah mempelajari dan mencoba Single Linked List dan Double Linked List, dapat disimpulkan bahwa kedua struktur data ini merupakan representasi dari data yang terhubung dengan node lainnya. Namun, Single Linked List hanya memiliki reference ke node berikutnya, sedangkan Double Linked List memiliki reference ke node sebelumnya dan node berikutnya. Dengan demikian, Double Linked List memiliki beberapa kelebihan dibanding Single Linked List, seperti lebih mudah dalam melakukan operasi pada node tertentu dan lebih efisien dalam melakukan operasi insert dan delete. Namun, Double Linked List membutuhkan memori yang lebih besar karena harus menyimpan reference ke node sebelumnya.

Dalam pengimplementasian kedua struktur data ini, dapat dilihat bahwa Double Linked List memiliki beberapa method tambahan dibanding Single Linked List, seperti method update, insert, dan remove. Hal ini dapat mempermudah dalam mengelola data yang ada pada linked list. Selain itu, dalam pengimplementasian kedua struktur data ini, dapat dilihat bahwa ada beberapa validasi yang harus dilakukan, seperti validasi posisi yang dimasukkan oleh user dan validasi apakah linked list kosong atau tidak. Dengan adanya validasi tersebut, program dapat menghindari error dan mencapai kehandalan yang lebih baik. Program ini juga memiliki beberapa fitur lain, seperti menghapus semua data pada linked list dan menampilkan seluruh data yang ada pada linked list. Hal ini dapat membantu user dalam mengelola data yang ada pada linked list.

Dalam keseluruhan, Single Linked List dan Double Linked List merupakan struktur data yang sangat berguna dalam pengembangan program, terutama dalam pengelolaan data yang terhubung dengan node lainnya. Selain itu, dengan mengimplementasikan beberapa validasi dan fitur tambahan, program dapat menjadi lebih handal dan mudah digunakan.

E. REFERENSI

Asisten praktikum, “Modul 3 Single Linked List and Double Linked List”, learning Management System, 2024

Bahrul Ulum, M. (2018). *Linked List*. Diakses pada 30 Maret 2024, dari https://lms-paralel.esaunggul.ac.id/pluginfile.php?file=%2F86227%2Fmod_resource%2Fcontent%2F1%2FModul%20Struktur%20Data-Linked%20List.pdf

Daisma Bali. (28 Desember 2019). *Mengenal Single Linked List dalam Struktur Data*. Diakses pada 30 Maret 2024, dari https://daismabali.com/artikel_detail/54/1/Mengenal-Single-Linked-List-dalam-Struktur-Data.html