

LAPORAN PRAKTIKUM
MODUL 3
SINGLE AND DOUBLE LINKED LIST



Disusun oleh:

Reli Gita Nurhidayati

NIM: 2311102025

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

Praktikum ini memiliki tujuan, yaitu:

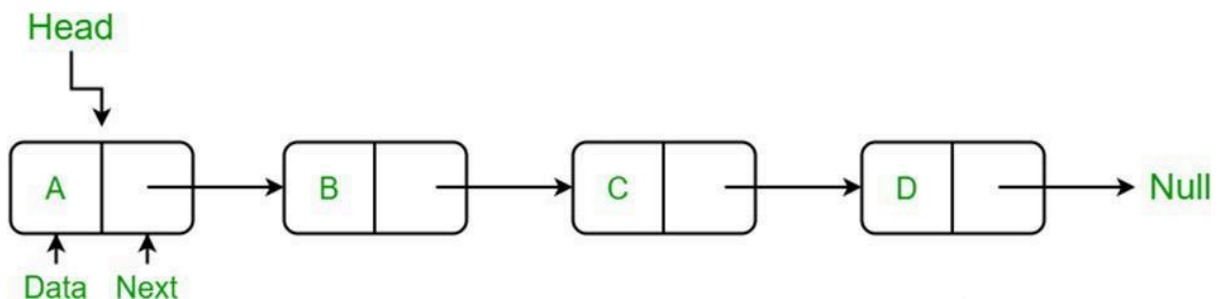
1. Mahasiswa memahami perbeddaan konsep Single and Double Linked List
2. Mahasiswa dapat menerapkan Single and Double Linked List ke dalam pemrograman

BAB II

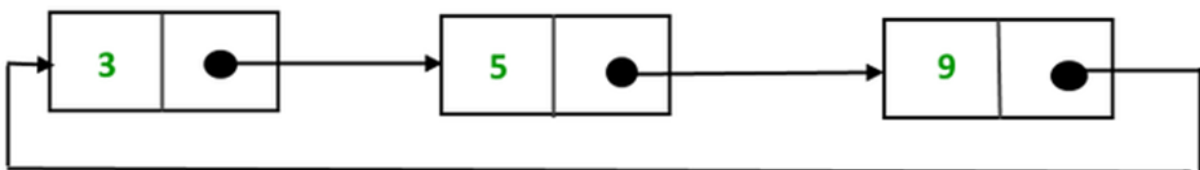
DASAR TEORI

a) Single Linked List

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



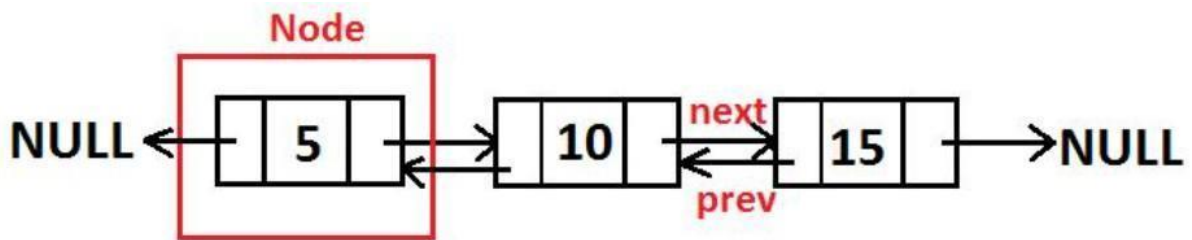
Dalam operasi Single Linked List, umumnya dilakukan operasi penambahan dan penghapusan simpul pada awal atau akhir daftar, serta pencarian dan pengambilan nilai pada simpul tertentu dalam daftar. Karena struktur data ini hanya memerlukan satu pointer untuk setiap simpul, maka Single Linked List umumnya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



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 besar dibandingkan dengan Single Linked List, karena setiap simpul membutuhkan satu pointer tambahan. Selain itu, Double Linked List juga membutuhkan waktu eksekusi yang lebih lama dalam operasi penambahan dan penghapusan jika dibandingkan dengan Single Linked List. Representasi sebuah double linked list dapat dilihat pada gambar berikut ini:



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

BAB III

GUIDED

GUIDED 1

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;
    }
}

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

```



```

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

int main()
{
    init();
    insertDepan(3, "satu");
    tampil();
    insertBelakang(5, "dua");
    tampil();
    insertDepan(2, "tiga");
    tampil();
    insertDepan(1, "empat");
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, "lima", 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1, "enam");
    tampil();
    ubahBelakang(8, "tujuh");
    tampil();
    ubahTengah(11, "delapan", 2);
    tampil();
    return 0;
}

```

Screenshoot Program

```

PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunne
rFile } ; if ($?) { .\tempCodeRunnerFile }
3
satu
3
satu5 dua
2
tiga3 satu5 dua
1
empat2 tiga3 satu5 dua
2
tiga3 satu5 dua
2
tiga3 satu lima3 satu
2
tiga3 lima3 satu
2
tiga3 satu
1
enam3 satu
1
enam8 tujuh
1
enam11 delapan
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT\" ; if ($?) { g++ GUIDED 2 MOSUL 3.cpp -o GUIDED 2 MOSUL

```

Deskripsi Program

Program ini adalah implementasi dari single linked list non-circular menggunakan bahasa pemrograman C++. Berikut adalah fungsionalitas utama yang disediakan oleh program ini:

1. **Deklarasi Struct Node** berguna untuk Mendefinisikan struktur data Node yang berisi data, kata, dan pointer ke Node selanjutnya.
2. **Inisialisasi Node** berguna untuk Menginisialisasi pointer head dan tail ke NULL.
3. **Pengecekan List Kosong** berguna untuk Memeriksa apakah linked list kosong atau tidak.
4. **Tambah Node di Depan List** berguna untuk Menambahkan node baru di depan linked list.
5. **Tambah Node di Belakang List** berguna untuk Menambahkan node baru di belakang linked list.
6. **Hitung Jumlah Node di List** berguna untuk Menghitung jumlah node dalam linked list.
7. **Tambah Node di Posisi Tertentu** berguna untuk Menambahkan node baru pada posisi tertentu dalam linked list.
8. **Hapus Node di Depan List** berguna untuk Menghapus node pertama dari linked list.
9. **Hapus Node di Belakang List** berguna untuk Menghapus node terakhir dari linked list.
10. **Hapus Node di Posisi Tertentu** berguna untuk Menghapus node pada posisi tertentu dalam linked list.
11. **Ubah Node di Depan List** berguna untuk Mengubah data node pertama dalam linked list.
12. **Ubah Node di Belakang List** berguna untuk Mengubah data node terakhir dalam linked list.
13. **Ubah Node di Posisi Tertentu** berguna untuk Mengubah data node pada posisi tertentu dalam linked list.
14. **Hapus Seluruh Node di List** berguna untuk Menghapus semua node dalam linked list.
15. **Menampilkan Isi List** berguna untuk Menampilkan isi dari linked list.

Program ini kemudian menunjukkan contoh penggunaan fungsi-fungsi tersebut dengan melakukan penambahan, penghapusan, dan perubahan data pada linked list serta menampilkan isi dari linked list setiap kali operasi akan dilakukan.

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

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

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

};

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

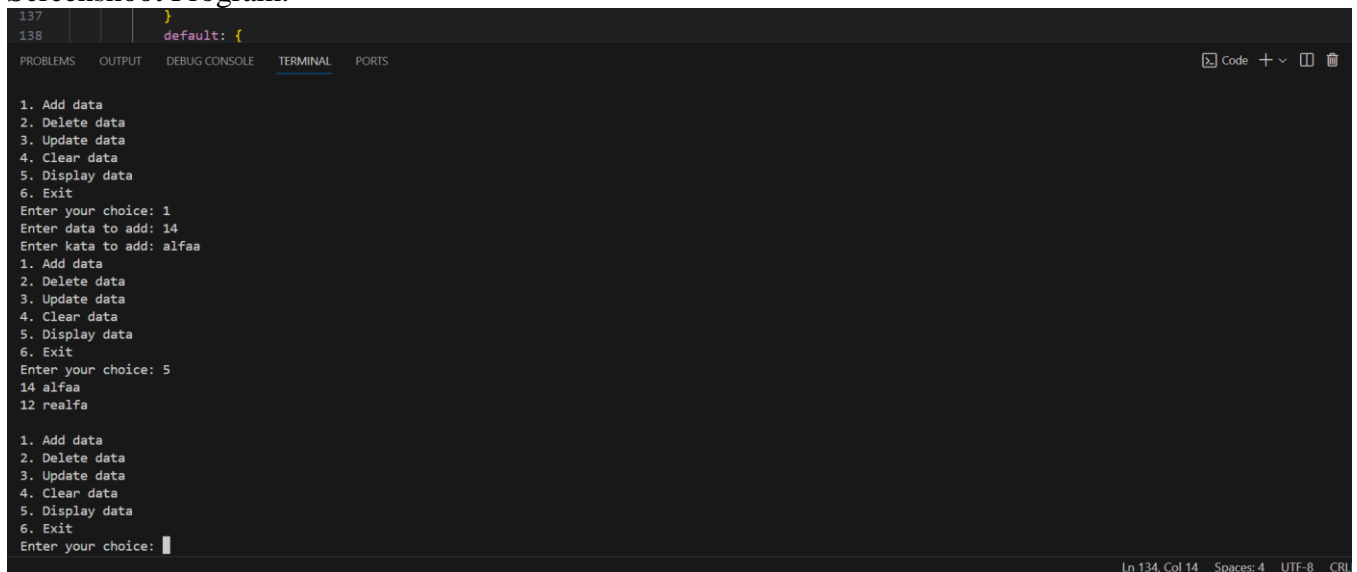
```

```

        case 1: {
            int data;
            string kata;
            cout << "Enter data to add: ";
            cin >> data;
            cout << "Enter kata to add: ";
            cin >> kata;
            list.push(data, kata);
            break;
        }
        case 2: {
            list.pop();
            break;
        }
        case 3: {
            int oldData, newData;
            string newKata;
            cout << "Enter old data: ";
            cin >> oldData;
            cout << "Enter new data: ";
            cin >> newData;
            cout << "Enter new kata: ";
            cin >> newKata;
            bool updated = list.update(oldData,
                                      newData, newKata);
            if (!updated) {
                cout << "Data not found" << endl;
            }
            break;
        }
        case 4: {
            list.deleteAll();
            break;
        }
        case 5: {
            list.display();
            break;
        }
        case 6: {
            return 0;
        }
        default: {
            cout << "Invalid choice" << endl;
            break;
        }
    }
}
return 0;
}

```

Screenshoot Program:



```
137 }
138 default: {
PROBLEMS OUTPUT DEBUG CONSOLE 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: 14
Enter kata to add: alfaa
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
14 alfaa
12 realfa
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 
```

Deskripsi Program

Program ini merupakan implementasi dari double linked list yang menggunakan bahasa pemrograman c++. Berikut adalah deskripsi singkat dari fungsionalitas yang disediakan oleh program ini:

1. **Kelas Node**: Mendefinisikan struktur data Node yang memiliki data, kata, serta pointer prev dan next.
2. **Kelas DoublyLinkedList**: Menyediakan operasi-operasi seperti push (menambahkan node di awal), pop (menghapus node di awal), update (memperbarui data), deleteAll (menghapus semua data), dan display (menampilkan semua data).
3. **Metode push**: Menambahkan node baru di awal linked list.
4. **Metode pop**: Menghapus node pertama dari linked list.
5. **Metode update**: Memperbarui data node berdasarkan nilai data lama.
6. **Metode deleteAll**: Menghapus semua node dalam linked list.
7. **Metode display**: Menampilkan semua data yang ada dalam linked list.
8. **Fungsi main**: Memberikan menu interaktif kepada pengguna untuk menambahkan, menghapus, memperbarui, menghapus semua data, dan menampilkan data dalam linked list.
9. **Loop Utama**: Program akan terus berjalan hingga pengguna memilih untuk keluar (pilihan 6).

Dengan menggunakan program ini, akan sangat mempermudah pengguna untuk mengelola data dalam double linked list melalui antarmuka yang disediakan.

BAB IV

UNGUIDED

1. Soal mengenai Single Linked List

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

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

[Nama_anda]	[Usia_anda]
John	19
Jane	20
Michael	18
Yusuke	19
Akechi	20
Hoshino	18
Karin	18

b. Hapus data Akechi

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

d. Tambahkan data berikut diawal : Igor 20

e. Ubah data Michael menjadi : Reyn 18

f. Tampilkan seluruh data

Source Code

```
#include <iostream>
#include <string>

using namespace std;

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

Node* head = nullptr;

// Fungsi untuk menambahkan data di depan linked list
void insertDepan(string nama, int usia) {
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    newNode->next = head;
    head = newNode;
}

// Fungsi untuk menambahkan data di belakang linked list
void insertBelakang(string nama, int usia) {
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;
    newNode->next = nullptr;
```

```

if (head == nullptr) {
    head = newNode;
    return;
}

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

temp->next = newNode;
}

// Fungsi untuk menambahkan data di tengah linked list
void insertTengah(string nama, int usia, string nama_sebelumnya) {
    Node* newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;

    Node* temp = head;
    while (temp != nullptr) {
        if (temp->nama == nama_sebelumnya) {
            newNode->next = temp->next;
            temp->next = newNode;
            return;
        }
        temp = temp->next;
    }
}

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

int main() {
    string namaAnda;
    int usiaAnda;

    cout << "Masukkan nama anda: ";
    getline(cin, namaAnda);
    cout << "Masukkan usia anda: ";
    cin >> usiaAnda;
    cin.ignore();

    insertDepan(namaAnda, usiaAnda);
    insertBelakang("John", 19);
    insertBelakang("Jane", 20);
    insertBelakang("Michael", 18);
    insertBelakang("Yusuke", 19);
    insertBelakang("Akechi", 20);
    insertBelakang("Hoshino", 18);
    insertBelakang("Karin", 18);

    cout << "Data Mahasiswa:" << endl;
    tampilkanData();

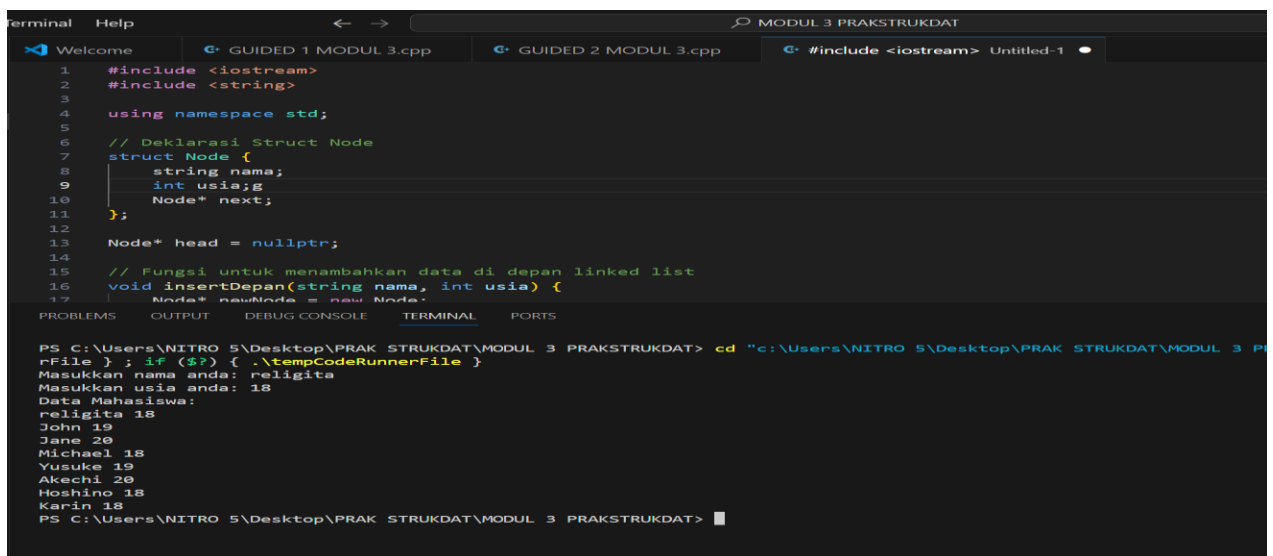
    return 0;
}

```

a. Source Code

```
insertDepan(namaAnda, usiaAnda);
insertBelakang("John", 19);
insertBelakang("Jane", 20);
insertBelakang("Michael", 18);
insertBelakang("Yusuke", 19);
insertBelakang("Akechi", 20);
insertBelakang("Hoshino", 18);
insertBelakang("Karin", 18);
```

Screenshoot Program



The screenshot shows a VS Code editor with a C++ file named 'MODUL 3 PRAKSTRUKDAT'. The code defines a linked list structure and a function to insert data at the front. The terminal output shows the program running and displaying the data entered by the user.

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 // Deklarasi Struct Node
7 struct Node {
8     string nama;
9     int usia;
10    Node* next;
11 };
12
13 Node* head = nullptr;
14
15 // Fungsi untuk menambahkan data di depan linked list
16 void insertDepan(string nama, int usia) {
17     Node* newNode = new Node;
```

```
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT"
nFile } ; if ($?) { .\tempCodeRunnerFile }
Masukkan nama anda: religita
Masukkan usia anda: 18
Data Mahasiswa:
religita 18
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT>
```

Deskripsi Program

Setelah melakukan run pada program dengan media vscode tersebut, pengguna akan diminta memberi input berupa nama pengguna serta usia pengguna, output yang dihasilkan maka akan seperti screenshoot pada program tersebut.

b. Source Code

```
insertDepan(namaAnda, usiaAnda);
insertBelakang("John", 19);
insertBelakang("Jane", 20);
insertBelakang("Michael", 18);
insertBelakang("Yusuke", 19);
insertBelakang("Hoshino", 18);
insertBelakang("Karin", 18);
```


Screenshoot Program

```
65     cout << temp->nama << " " << temp->usia << endl;
66     temp = temp->next;

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT\" ; if
rFile } ; if ($?) { .\tempCodeRunnerFile }
Masukkan nama anda: religita
Masukkan usia anda: 18
Data Mahasiswa:
religita 18
John 19
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> |
```

Deskripsi Program

Untuk perintah hapus data akechi, pengguna hanya perlu menghapus data bagian akechi seperti pada source code diatas, maka data akechi tidak akan keluar pada output setelah program di run.

c. Source Code

```
insertDepan(namaAnda, usiaAnda);
insertBelakang("John", 19);
insertBelakang("Futaba", 18);
insertBelakang("Jane", 20);
insertBelakang("Michael", 18);
insertBelakang("Yusuke", 19);
insertBelakang("Hoshino", 18);
insertBelakang("Karin", 18);
```

Screenshoot Program

```
Terminal Help
Welcome
60 // Fungsi untuk menampilkan seluruh data
61 void tampilkanData() {
62     node* temp = head;
63     while (temp != nullptr) {
64         cout << temp->nama << " " << temp->usia << endl;
65         temp = temp->next;
66     }
67 }
68
69
70 int main() {
71     string namaAnda;
72     int usiaAnda;
73
74     cout << "Masukkan nama anda: ";
75     getline(cin, namaAnda);
76
77     PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunne
rFile } ; if ($?) { .\tempCodeRunnerFile }
Masukkan nama anda: religita
Masukkan usia anda: 18
Data Mahasiswa:
religita 18
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> |
```

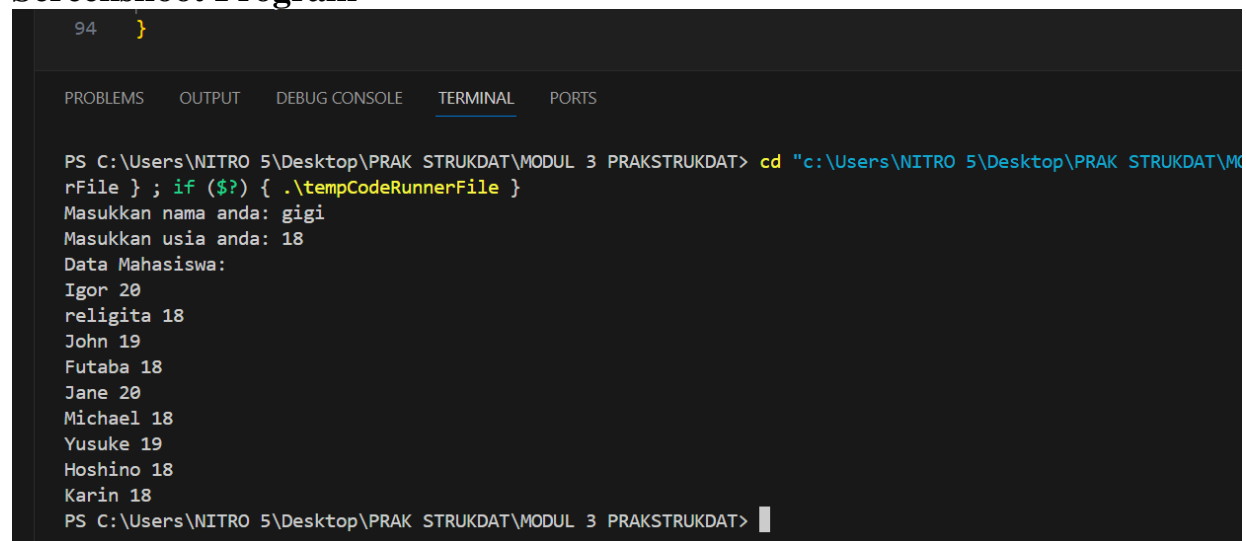
Deskripsi Program

Untuk perintah tambahkan data Futaba 18 di antara data John dan Jane, pengguna hanya perlu menambah (insertBelakang("Futaba", 18);) di antara data John dan Jane, setelah itu apabila program di run, maka output yang dihasilkan akan seperti screenshoot program di atas.

d. Source Code

```
insertDepan(namaAnda, usiaAnda);
insertBelakang("Igor",20);
insertBelakang("John", 19);
insertBelakang("Futaba", 18);
insertBelakang("Jane", 20);
insertBelakang("Michael", 18);
insertBelakang("Yusuke", 19);
insertBelakang("Hoshino", 18);
insertBelakang("Karin", 18);
```

Screenshot Program



```
94 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT"
rFile } ; if ($?) { .\tempCodeRunnerFile }
Masukkan nama anda: gigi
Masukkan usia anda: 18
Data Mahasiswa:
Igor 20
religita 18
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT>
```

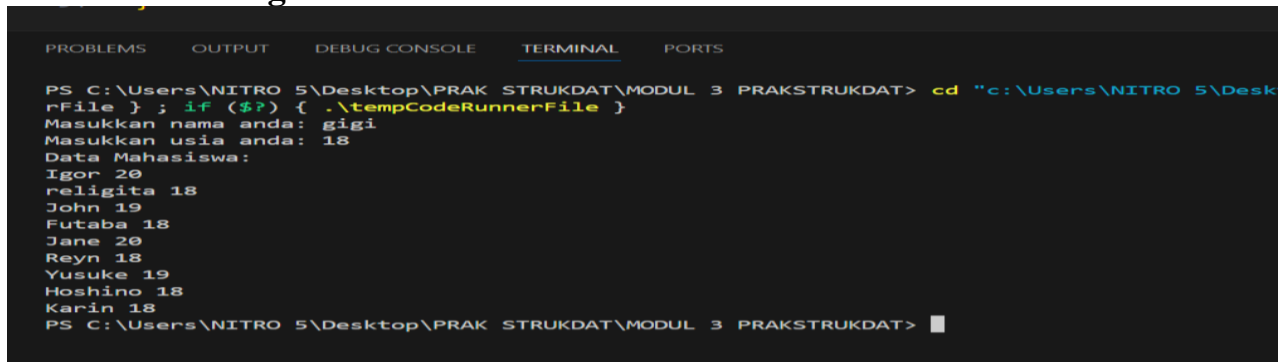
Deskripsi Program

Untuk perintah Tambahkan data berikut diawal : **Igor 20** , pengguna hanya perlu menambah (insertBelakang("Igor", 20);) setelah data nama pengguna, setelah itu apabila program di run, maka output yang dihasilkan akan seperti screenshot program di atas.

e. Source Code

```
insertDepan(namaAnda, usiaAnda);
insertBelakang("Igor",20);
insertBelakang("Religita", 18);
insertBelakang("John", 19);
insertBelakang("Futaba", 18);
insertBelakang("Jane", 20);
insertBelakang("Reyn", 18);
insertBelakang("Yusuke", 19);
insertBelakang("Hoshino", 18);
insertBelakang("Karin", 18);
```

Screenshoot Program



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\Desktop\PRAK STRUKDAT"
rFile } ; if ($?) { .\tempCodeRunnerFile }
Masukkan nama anda: gigi
Masukkan usia anda: 18
Data Mahasiswa:
Igor 20
religita 18
John 19
Futaba 18
Jane 20
Reyn 18
Yusuke 19
Hoshino 18
Karin 18
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT>
```

Deskripsi Program

Untuk perintah Ubah data Michael menjadi : **Reyn 18** , pengguna hanya perlu merubah data yang sebelumnya(`insertBelakang("Michael",18);`) menjadi `(insertBelakang("Reyn",18);` , Maka setelah itu, saat program Kembali di run, akan menghasilkan screenshoot seperti gambar di atas.

2. Soal mengenai Double Linked List

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

Skintific	100.000
Wardah	50.000
Hanasui	30.000

Nama Produk	Harga
Originote	60.000
Somethinc	150.000

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

Source Code

```

// Unguided 2
#include <iostream>
#include <string>
using namespace std;
struct Node
{
    string namaProduk;
    int harga;
    Node *prev;
    Node *next;
};
class DoublyLinkedList
{
private:
    Node *head;
    Node *tail;

public:
    DoublyLinkedList() : head(nullptr), tail(nullptr) {}
    void insertBelakang(string namaProduk, int harga)
    {
        Node *newNode = new Node;
        newNode->namaProduk = namaProduk;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = nullptr;
        if (head == nullptr)
        {
            head = tail = newNode;
        }
        else
        {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
    }
    void display()
    {
        Node *current = head;
        while (current != nullptr)
        {
            cout << current->namaProduk << "\t" << current->harga << endl;
            current = current->next;
        }
    }
};
int main()
{
    DoublyLinkedList list;
    // Menambahkan data
    list.insertBelakang("Originote", 60000);
    list.insertBelakang("Somethinc", 150000);
    list.insertBelakang("Skintific", 100000);
    list.insertBelakang("Wardah", 50000);
    list.insertBelakang("Hanasui", 30000);
    cout << "Nama Produk\tHarga" << endl;
    list.display();
    return 0;
}

```

Case 1

```
:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Nama Produk      Harga
Originote         60000
Somethinc         150000
Azarine 65000
Skintific         100000
Wardah 50000
Hanasui 30000
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT>
```

Case 2

```
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Nama Produk      Harga
Originote         60000
Somethinc         150000
Azarine 65000
Skintific         100000
Hanasui 30000
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT>
```

Case 3

```
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> cd "c:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Nama Produk      Harga
Originote         60000
Somethinc         150000
Azarine 65000
Skintific         100000
Cleora 55000
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT>
```

Case 4

```
Hanasui 30000
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> cd "c
:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT\" ; if ($?
) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\
tempCodeRunnerFile }
Nama Produk      Harga
Originote        60000
Somethinc         150000
Azarine 65000
Skintific         100000
Cleora 55000
PS C:\Users\NITRO 5\Desktop\PRAK STRUKDAT\MODUL 3 PRAKSTRUKDAT> |
```

Ln 46, Col 36 Spaces: 4 UTF-8 CRLF C++

Deskripsi Program

Program C++ di atas merupakan program yang menggunakan Doubly Linked List untuk menyimpan data produk berupa nama dan harga. Program ini memiliki struktur data Node yang memiliki informasi tentang nama produk, harga, serta pointer ke node sebelumnya (prev) dan node selanjutnya (next). Kelas DoublyLinkedList memiliki metode untuk memasukkan data ke belakang list (insertBelakang) dan menampilkan data (display). Di dalam fungsi main, program membuat objek DoublyLinkedList, memasukkan beberapa data produk, dan menampilkan daftar nama produk beserta harganya.

BAB V

KESIMPULAN

Linked list adalah struktur data linier yang terdiri dari sejumlah simpul (node) yang saling terhubung. Terdapat dua jenis utama linked list: Single Linked List dan Double Linked List.

Single Linked List:

1. Single Linked List terdiri dari simpul-simpul yang terhubung satu arah, memungkinkan akses dari simpul pertama ke terakhir.
2. Setiap simpul memiliki dua bagian, yaitu data dan pointer ke simpul berikutnya.
3. Operasi umum untuk Single Linked List meliputi penambahan di awal atau akhir, penghapusan, pencarian, dan traversal.
4. Kelebihan Single Linked List termasuk penggunaan memori yang efisien untuk penambahan atau penghapusan elemen di tengah list, serta fleksibilitas dalam ukuran list.

Double Linked List:

1. Double Linked List mirip dengan Single Linked List, namun setiap simpul memiliki dua pointer: satu ke simpul sebelumnya dan satu ke simpul berikutnya.
2. Struktur Double Linked List memungkinkan traversal maju dan mundur, meningkatkan kecepatan akses terhadap elemen tertentu dalam list.
3. Operasi pada Double Linked List meliputi penambahan di awal atau akhir, penghapusan, pencarian, dan traversal maju dan mundur.
4. Kelebihan Double Linked List antara lain kemampuan untuk menghapus simpul dengan lebih efisien karena memiliki akses ke simpul sebelumnya, serta kemampuan traversal maju dan mundur.

Perbedaan utama antara Single Linked List dan Double Linked List terletak pada jumlah pointer yang dimiliki setiap simpul. Single Linked List hanya memiliki satu pointer untuk mengakses simpul berikutnya, sedangkan Double Linked List memiliki dua pointer untuk mengakses simpul sebelumnya dan berikutnya. Pemilihan antara kedua jenis linked list tergantung pada kebutuhan spesifik dalam implementasi struktur data, di mana Double Linked List dapat lebih efisien untuk operasi-operasi tertentu seperti penghapusan dan traversal mundur.

DAFTAR PUSTAKA

- [1] Komang Setia, B. (2018). *Struktur Data*. BukuKita.com diakses dari https://books.google.co.id/books?hl=id&lr=&id=EH2DDwAAQBAJ&oi=fnd&pg=PA1&dq=materi+double+linked+list&ots=gL3b8nxjb8&sig=fpytk4O4Rx7hQkYdwiqce16KUhE&redir_esc=y#v=onepage&q&f=false