LAPORAN PRAKTIKUM

MODUL III
SINGLE AMD DOUBLE LINKED LIST

Disusun oleh:
Rizal Dwi Anggoro
2311102034

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

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023

BAB I
TUJUAN PRAKTIKUM

A. Tujuan Praktikum

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

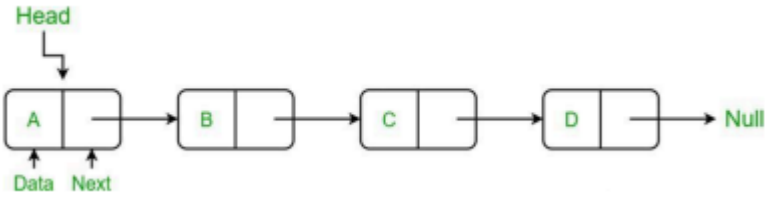
BAB II

DASAR TEORI

a) Single Linked List

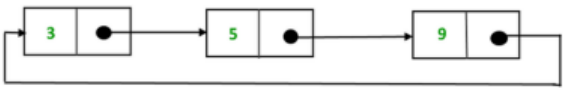
Linked List merupakan suatu bentuk struktur data yang berisi kumpulan data yang disebut sebagai node yang tersusun secara sekuensial, saling sambung menyambung, dinamis, dan terbatas. Setiap elemen dalam linked list dihubungkan ke elemen lain melalui pointer. Masing-masing komponen sering disebut dengan 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.

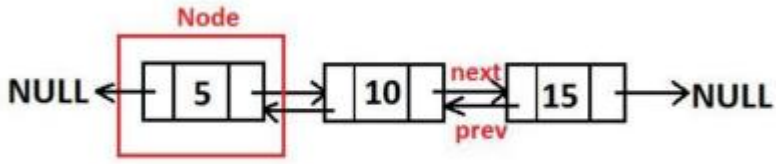


b) Double Linked List

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 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;
```

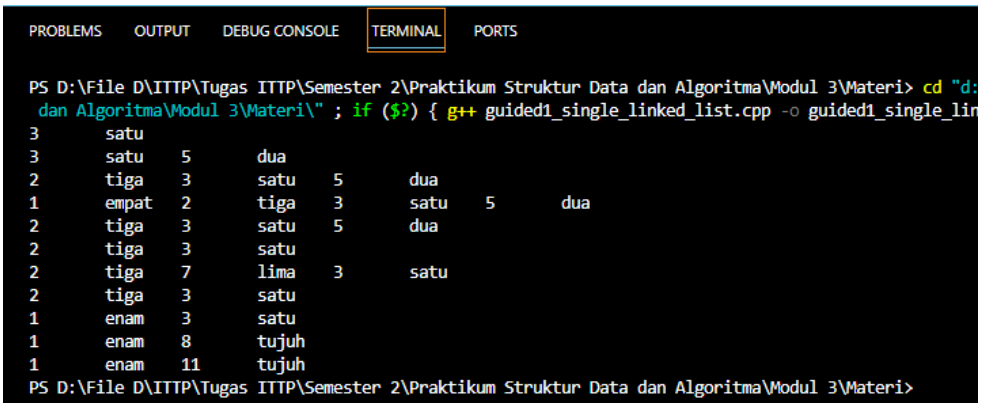
```
        head->kata = kata;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Tengah
void ubahTengah(int data, string kata, int posisi)
{
    Node *bantu;
    if (isEmpty() == false)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "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;
        }
    }
```

```
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Belakang
void ubahBelakang(int data, string kata)
{
    if (isEmpty() == false)
    {
        tail->data = data;
        tail->kata = kata;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Hapus List
void clearList()
{
    Node *bantu, *hapus;
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}
// Tampilkan List
void tampil()
```

```
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << "\t";
            cout << bantu->kata<<"\t";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
int main()
{
    init();
    insertDepan(3, "satu");
    tampil();
    insertBelakang(5, "dua");
    tampil();
    insertDepan(2, "tiga");
    tampil();
    insertDepan(1, "empat");
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7,"lima", 2);
```

```
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1,"enam") ;
    tampil();
    ubahBelakang(8,"tujuh");
    tampil();
    ubahTengah(11,"delapan", 2);
    tampil();
    return 0;
}
```

Screenshoot program



Deskripsi program

Program C++ ini terdapat fungsi-fungsi untuk mengelola linked list seperti menambah simpul di depan, di belakang, di tengah, menghapus simpul di depan, di belakang, di tengah, serta mengubah nilai data pada simpul. Fungsi-fungsi ini mengatur bagaimana simpul-simpul ditambahkan, dihapus, dan diubah dalam linked list. Pada bagian int main program, dilakukan inisialisasi linked list, kemudian dilakukan berbagai operasi seperti menambah, menghapus, dan mengubah simpul-simpul, serta menampilkan isi dari linked list setelah setiap operasi dilakukan.

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 newKata)
    {
        Node *current = head;
        while (current != nullptr)
        {
            if (current->data == oldData)
```

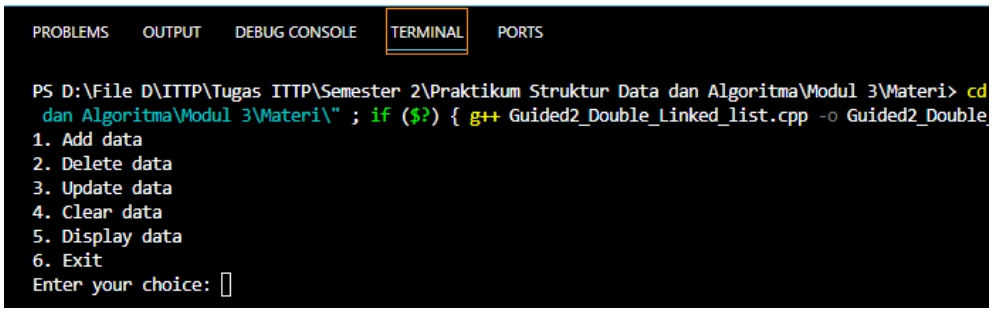
```
            {
                current->data = newData;
                current->kata = newKata;
                return true;
            }
            current = current->next;
        }
        return false;
    }

    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



Deskripsi program

Program C++ diatas memiliki dua pointer yaitu pointer ke node sebelumnya (prev) dan pointer ke node berikutnya (next). Kelas DoublyLinkedList memiliki berbagai metode untuk memanipulasi linked list, seperti menambahkan data ke depan, menghapus data dari depan, mengupdate data, menghapus seluruh data, serta menampilkan isi dari linked list.

Di int main menggunakan loop untuk memungkinkan pengguna untuk memilih operasi yang ingin dilakukan seperti menambah, menghapus, mengupdate, menghapus seluruh data, atau menampilkan isi linked list, dengan memberikan pilihan menu dan menangani input dari pengguna dengan switch-case.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

```
#include <iostream>

// 2311102034_Rizal Dwi Anggoro
using namespace std;

class Node
{
public:
    string nama;
    int usia;
    Node *next;
};

class LinkedList
{

public:
    Node *head;
    LinkedList()
    {
        head = NULL;
    }

    void AddData(string nama, int usia)
    {
        Node *newNode = new Node();
        newNode->nama = nama;
        newNode->usia = usia;
        newNode->next = head;
```

```
        head = newNode;
    }

    void insertAtEnd(string nama, int usia)
    {
        Node *newNode = new Node();
        newNode->nama = nama;
        newNode->usia = usia;
        newNode->next = NULL;
        if (head == NULL)
        {
            head = newNode;
            return;
        }
        Node *temp = head;
        while (temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }

    void insertAfter(string nama, int usia, string keynama)
    {
        Node *temp = head;
        while (temp != NULL)
        {
            if (temp->nama == keynama)
            {
                Node *newNode = new Node();
                newNode->nama = nama;
                newNode->usia = usia;
                newNode->next = temp->next;
                temp->next = newNode;
```

```
                return;
            }
            temp = temp->next;
        }
        cout << keynama << " Tidak Ada" << endl;
    }

    void updateNode(string nama, int usia)
    {
        Node *temp = head;
        while (temp != NULL)
        {
            if (temp->nama == nama)
            {
                temp->usia = usia;
                return;
            }
            temp = temp->next;
        }
        cout << nama << " Tidak Ada" << endl;
    }

    void deleteNode(string nama)
    {
        Node *temp = head;
        Node *prev = NULL;
        while (temp != NULL)
        {
            if (temp->nama == nama)
            {
                if (prev == NULL)
                {
                    head = temp->next;
                }
```

```
                else
                {
                    prev->next = temp->next;
                }
                delete temp;
                return;
            }
            prev = temp;
            temp = temp->next;
        }
        cout << nama << " Tidak Ada" << endl;
    }

    void clearAll()
    {
        Node *temp = head;
        while (temp != NULL)
        {
            Node *next = temp->next;
            delete temp;
            temp = next;
        }
        head = NULL;
    }

    // Display all nodes
    void display()
    {
        Node *temp = head;
        while (temp != NULL)
        {
            cout << "Nama Anda : " << temp->  nama << ", usia: "
<< temp->  usia << endl;
            temp = temp->next;
```

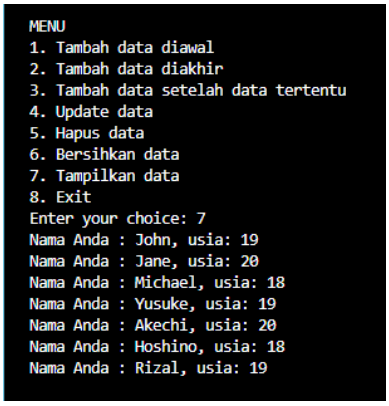
```
        }
    }
};
// Main function
int main()
{
    LinkedList list;
    int pilihan, position, usia;
    string nama, keynama;

    do
    {
        cout << endl;
        cout << "MENU" << endl;
        cout << "1. Tambah data diawal" << endl;
        cout << "2. Tambah data diakhir" << endl;
        cout << "3. Tambah data setelah data tertentu" << endl;
        cout << "4. Update data" << endl;
        cout << "5. Hapus data" << endl;
        cout << "6. Bersihkan data" << endl;
        cout << "7. Tampilkan data" << endl;
        cout << "8. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> pilihan;
        switch (pilihan){
        case 1:
            cout << "Masukan Nama: ";
            cin >> nama;
            cout << "Masukan Usia: ";
            cin >> usia;
            list.AddData(nama, usia);
            break;
        case 2:
            cout << "Masukan Nama: ";
```

```
            cin >> nama;
            cout << "Masukan Usia: ";
            cin >> usia;
            list.AddData(nama, usia);
            break;
        case 3:
            cout << "Masukan Nama: ";
            cin >> nama;
            cout << "Masukan Usia: ";
            cin >> usia;
            cout << "Masukan Nama Setelah: ";
            cin >> keynama;
            list.insertAfter(nama, usia, keynama);
            break;
        case 4:
            cout << "Masukan Update Nama: ";
            cin >> nama;
            cout << "Masukan Update Usia: ";
            cin >> usia;
            list.updateNode(nama, usia);
            break;
        case 5:
            cout << "Masukan Nama yang Dihapus: ";
            cin >> nama;
            list.deleteNode(nama);
            break;
        case 6:
            list.clearAll();
            break;
        case 7:
            list.display();
            break;
        case 8:
            cout << "Keluar program" << endl;
```

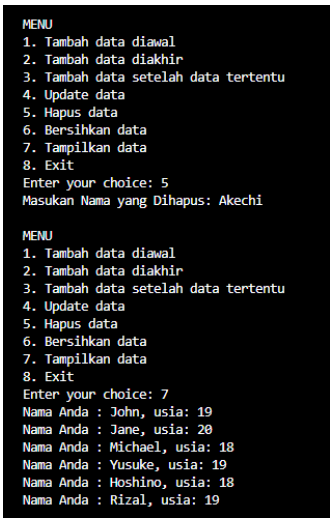
```
            break;
        default:
            cout << "Salah Pilihan" << endl;
        }
    } while (pilihan != 8);
    return 0;
}
```

Screenshoot program

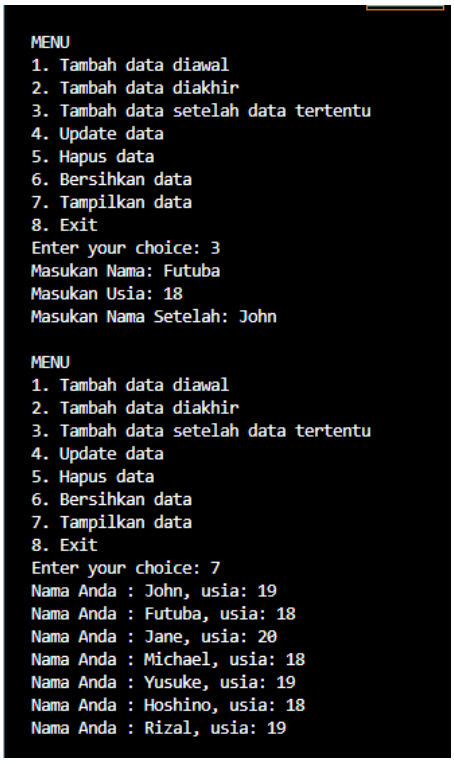
a) Data pertama yang dimasukkan adalah nama dan usia anda.



b) Hapus data Akechi

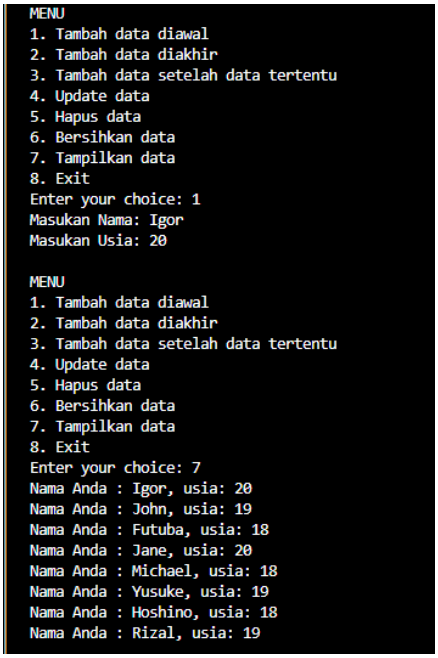


c) Tambahkan data berikut diantara John dan Jane : Futaba 18



```
MENU
1. Tambah data diawal
2. Tambah data diakhir
3. Tambah data setelah data tertentu
4. Update data
5. Hapus data
6. Bersihkan data
7. Tampilkan data
8. Exit
Enter your choice: 3
Masukan Nama: Futuba
Masukan Usia: 18
Masukan Nama Setelah: John

MENU
1. Tambah data diawal
2. Tambah data diakhir
3. Tambah data setelah data tertentu
4. Update data
5. Hapus data
6. Bersihkan data
7. Tampilkan data
8. Exit
Enter your choice: 7
Nama Anda : John, usia: 19
Nama Anda : Futuba, usia: 18
Nama Anda : Jane, usia: 20
Nama Anda : Michael, usia: 18
Nama Anda : Yusuke, usia: 19
Nama Anda : Hoshino, usia: 18
Nama Anda : Rizal, usia: 19
```

d) Tambahkan data berikut diawal : Igor 20



```
MENU
1. Tambah data diawal
2. Tambah data diakhir
3. Tambah data setelah data tertentu
4. Update data
5. Hapus data
6. Bersihkan data
7. Tampilkan data
8. Exit
Enter your choice: 1
Masukan Nama: Igor
Masukan Usia: 20

MENU
1. Tambah data diawal
2. Tambah data diakhir
3. Tambah data setelah data tertentu
4. Update data
5. Hapus data
6. Bersihkan data
7. Tampilkan data
8. Exit
Enter your choice: 7
Nama Anda : Igor, usia: 20
Nama Anda : John, usia: 19
Nama Anda : Futuba, usia: 18
Nama Anda : Jane, usia: 20
Nama Anda : Michael, usia: 18
Nama Anda : Yusuke, usia: 19
Nama Anda : Hoshino, usia: 18
Nama Anda : Rizal, usia: 19
```

e) Ubah data Michael menjadi : Reyn 18

```
MENU
1. Tambah data diawal
2. Tambah data diakhir
3. Tambah data setelah data tertentu
4. Update data
5. Hapus data
6. Bersihkan data
7. Tampilkan data
8. Exit
Enter your choice: 7
Nama Anda : Igor, usia: 20
Nama Anda : John, usia: 19
Nama Anda : Futuba, usia: 18
Nama Anda : Jane, usia: 20
Nama Anda : Reyn, usia: 18
Nama Anda : Yusuke, usia: 19
Nama Anda : Hoshino, usia: 18
Nama Anda : Rizal, usia: 19
```

f) Tampilkan seluruh data

```
MENU
1. Tambah data diawal
2. Tambah data diakhir
3. Tambah data setelah data tertentu
4. Update data
5. Hapus data
6. Bersihkan data
7. Tampilkan data
8. Exit
Enter your choice: 7
Nama Anda : Igor, usia: 20
Nama Anda : John, usia: 19
Nama Anda : Futuba, usia: 18
Nama Anda : Jane, usia: 20
Nama Anda : Reyn, usia: 18
Nama Anda : Yusuke, usia: 19
Nama Anda : Hoshino, usia: 18
Nama Anda : Rizal, usia: 19
```

Deskripsi program

Code C++ diatas merupakan implementasi dari single linked list. Dalam kode ini, terdapat definisi kelas Node yang merepresentasikan simpul dalam linked list, dengan atribut nama dan usia serta pointer next yang menunjukkan ke simpul berikutnya. Kelas LinkedList digunakan untuk mengelola linked list, dengan metode tambah data di awal, tambah data di akhir, tambah data setelah data tertentu, update data, hapus data, bersihkan data, dan tampilkan data.

2. Unguided 2

Source code

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

//2311102034_Rizal Dwi Anggoro

struct Node
{
    string nama;
    int harga;
    Node *prev;
    Node *next;
};

class DoubleLinkedList
{
private:
    Node *head;
    Node *tail;
    int size;
```

```
public:
    DoubleLinkedList()
    {
        head = NULL;
        tail = NULL;
        size = 0;
    }

    void addData(string nama, int harga)
    {
        Node *node = new Node;
        node->nama = nama;
        node->harga = harga;
        node->prev = tail;
        node->next = NULL;
        if (head == NULL)
        {
            head = node;
            tail = node;
        }
        else
        {
            tail->next = node;
            tail = node;
        }
        size++;
    }

    void addDataAt(int index, string nama, int harga)
    {
        if (index < 0 || index > size)
        {
            cout << "Index out of bounds" << endl;
```

```
            return;
        }
        Node *node = new Node;
        node->nama = nama;
        node->harga = harga;
        if (index == 0)
        {
            node->prev = NULL;
            node->next = head;
            head->prev = node;
            head = node;
        }
        else if (index == size)
        {
            node->prev = tail;
            node->next = NULL;
            tail->next = node;
            tail = node;
        }
        else
        {
            Node *current = head;
            for (int i = 0; i < index - 1; i++)
            {
                current = current->next;
            }
            node->prev = current;
            node->next = current->next;
            current->next->prev = node;
            current->next = node;
        }
        size++;
    }
```

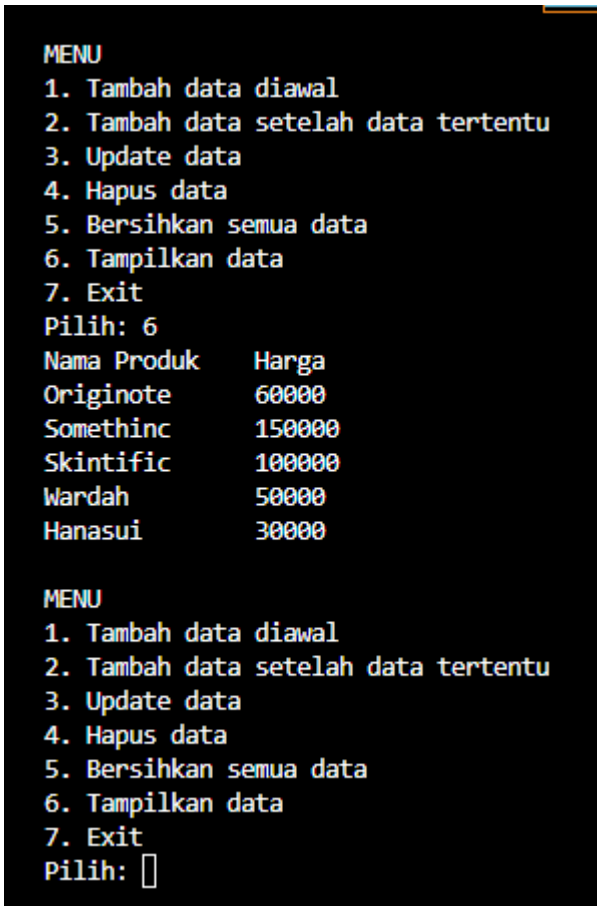
```
void deleteDataAt(int index)
{
    if (index < 0 || index >= size)
    {
        cout << "Index out of bounds" << endl;
        return;
    }
    if (index == 0)
    {
        Node *temp = head;
        head = head->next;
        head->prev = NULL;
        delete temp;
    }
    else if (index == size - 1)
    {
        Node *temp = tail;
        tail = tail->prev;
        tail->next = NULL;
        delete temp;
    }
    else
    {
        Node *current = head;
        for (int i = 0; i < index; i++)
        {
            current = current->next;
        }
        current->prev->next = current->next;
        current->next->prev = current->prev;
        delete current;
    }
    size--;
}
```

```
    void clearData()
    {
        while (head != NULL)
        {
            Node *temp = head;
            head = head->next;
            delete temp;
        }
        tail = NULL;
        size = 0;
    }

    void displayData()
    {
        cout << setw(15) << left << "Nama Produk" << setw(10) <<
left << "Harga" << endl;
        Node *current = head;
        while (current != NULL)
        {
           cout << setw(15) << left << current->nama << setw(10)
<< left << current->harga << endl;
            current = current->next;
        }
    }

    void updateDataAt(int index, string nama, int harga)
    {
        if (index < 0 || index >= size)
        {
            cout << "Index out of bounds" << endl;
            return;
        }
        Node *current = head;
```

```
        for (int i = 0; i < index; i++)
        {
            current = current->next;
        }
        current->nama = nama;
        current->harga = harga;
    }
};

int main()
{
    DoubleLinkedList dll;
    int pilihan;
    string nama;
    int harga;
    int index;
    do
    {
        cout << "MENU" << endl;
        cout << "1. Tambah data diawal" << endl;
        cout << "2. Tambah data setelah data tertentu" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Hapus data" << endl;
        cout << "5. Bersihkan semua data" << endl;
        cout << "6. Tampilkan data" << endl;
        cout << "7. Exit" << endl;
        cout << "Pilih: ";
        cin >> pilihan;
        switch (pilihan)
        {
        case 1:
            cout << "Nama Produk: ";
            cin >> nama;
            cout << "Harga: ";
```

```
            cin >> harga;
            dll.addData(nama, harga);
            break;
        case 2:
            cout << "Index: ";
            cin >> index;
            cout << "Nama Produk: ";
            cin >> nama;
            cout << "Harga: ";
            cin >> harga;
            dll.addDataAt(index, nama, harga);
            break;
        case 3:
            cout << "Index: ";
            cin >> index;
            cout << "Nama Produk: ";
            cin >> nama;
            cout << "Harga: ";
            cin >> harga;
            dll.updateDataAt(index, nama, harga);
            break;
        case 4:
            cout << "Index: ";
            cin >> index;
            dll.deleteDataAt(index);
            break;
        case 5:
            dll.clearData();
            break;
        case 6:
            dll.displayData();
            break;
        case 7:
            break;
```

```
        default:
            cout << "Pilihan tidak valid" << endl;
            break;
        }
        cout << endl;
    } while (pilihan != 7);
    return 0;
}
```



Screenshoot program

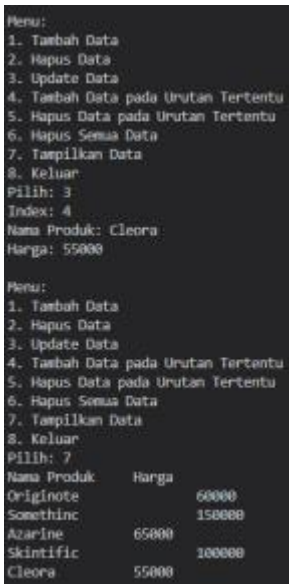
a. Tambahkan produk Azarine dengan harga 65000 diantara Somethinc dan Skintific

```
MENU
1. Tambah data diawal
2. Tambah data setelah data tertentu
3. Update data
4. Hapus data
5. Bersihkan semua data
6. Tampilkan data
7. Exit
Pilih: 2
Index: 2
Nama Produk: Azarine
Harga: 65000

MENU
1. Tambah data diawal
2. Tambah data setelah data tertentu
3. Update data
4. Hapus data
5. Bersihkan semua data
6. Tampilkan data
7. Exit
Pilih: 6
Nama Produk    Harga
Originote      60000
Somethinc      150000
Azarine        65000
Skintific      100000
Wardah         50000
Hanasui        30000
```

b. Hapus produk wardah

```
MENU
1. Tambah data diawal
2. Tambah data setelah data tertentu
3. Update data
4. Hapus data
5. Bersihkan semua data
6. Tampilkan data
7. Exit
Pilih: 4
Index: 4

MENU
1. Tambah data diawal
2. Tambah data setelah data tertentu
3. Update data
4. Hapus data
5. Bersihkan semua data
6. Tampilkan data
7. Exit
Pilih: 6
Nama Produk    Harga
Originote      60000
Somethinc      150000
Azarine        65000
Skintific      100000
Hasanui        30000
```

c. Update produk Hanasui menjadi Cleora dengan harga 55.000



d. 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

```
MENU
1. Tambah data diawal
2. Tambah data setelah data tertentu
3. Update data
4. Hapus data
5. Bersihkan semua data
6. Tampilkan data
7. Exit
Pilih: 6
Nama Produk    Harga
Originote      60000
Somethinc      150000
Azarine        65000
Skintific      100000
Cleora         55000
```

DESKRIPSI

Program C++ di atas merupakan implementasi Doubly Linked List yang memungkinkan pengguna untuk menambah, menghapus, memperbarui, membersihkan, dan menampilkan data. Setiap simpul menyimpan informasi nama produk dan harganya. Program dilengkapi dengan antarmuka pengguna sederhana yang memungkinkan pengguna memilih operasi yang ingin dilakukan melalui menu. Program akan berjalan sampai pengguna memilih untuk keluar.

BAB IV

KESIMPULAN

Dari praktikum ini, kami menyimpulkan bahwa pemahaman tentang struktur data seperti Linked List sangat penting dalam pengembangan perangkat lunak. Single Linked List cocok digunakan untuk aplikasi yang memerlukan penambahan atau penghapusan data terutama di ujung depan struktur, sedangkan Double Linked List lebih cocok digunakan jika aplikasi memerlukan navigasi maju dan mundur dalam struktur data