

# **LAPORAN PRAKTIKUM**

## **MODUL III SINGLE AND DOUBLE LINKED LIST**



**Disusun oleh:**  
**Imelda Fajar Awalina Crisyanti**  
**NIM: 2311102004**

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

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

# **BAB I**

## **TUJUAN PRAKTIKUM**

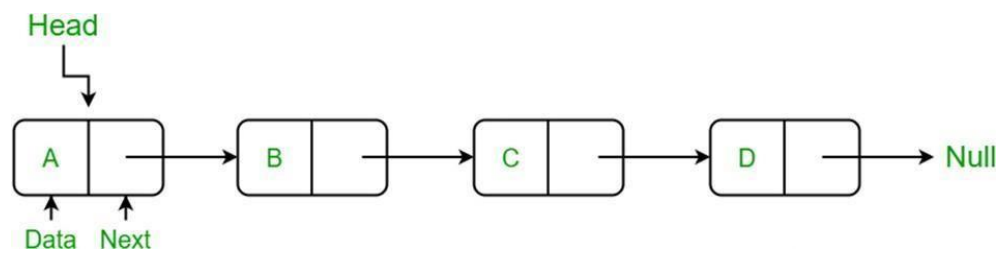
- 1.** Mahasiswa memahami perbedaan konsep Single dan Double Linked List
- 2.** Mahasiswa mampu menerapkan Single dan Double Linked List ke dalam pemrograman

## BAB II

### DASAR TEORI

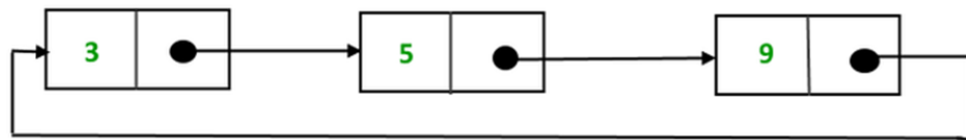
#### a) Single Linked List

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.

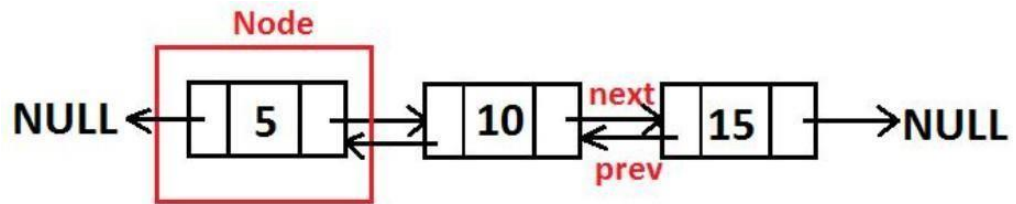


#### **b) Double Linked List**

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;

struct Node
{
    int data;
    string kata;
    Node *next;
};
Node *head;
Node *tail;

void init()
{
    head = NULL;
    tail = NULL;
}

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

void insertDepan(int nilai, string kata)
{
    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;
    }
}

void insertBelakang(int nilai, string kata )
{
    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;
    }
}

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

```

```

}

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;
    }
}

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;
}
}

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;
    }
}

```

```

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;
    }
}

void ubahDepan(int data, string kata)
{
    if (isEmpty() == false)
    {
        head->data = data;
    }
    else
    {

```

```

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

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

void ubahBelakang(int data, string kata)
{
    if (isEmpty() == false)
    {
        tail->data = data;
    }
}

```

```

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

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;
}

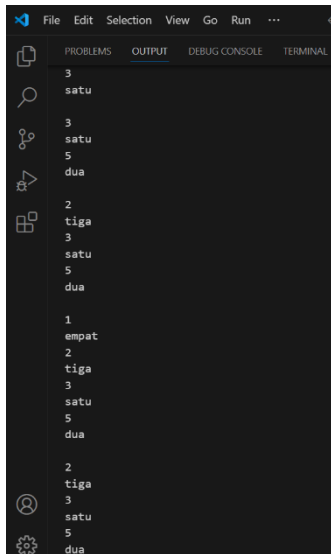
void tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << "\n";
            cout << bantu->kata << "\n";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        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;
}
```

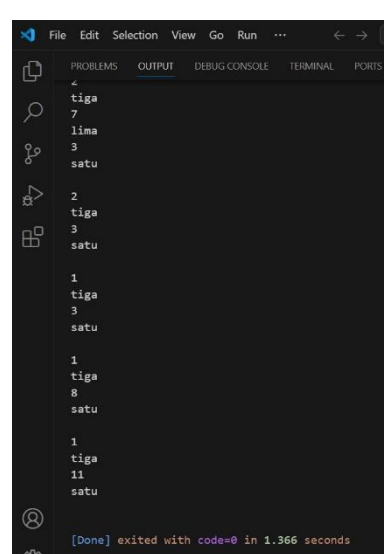
**Screenshoot program**



```
3
satu
3
satu
5
dua
2
tiga
3
satu
5
dua
1
empat
2
tiga
3
satu
5
dua
2
tiga
3
satu
5
dua
```



```
2
tiga
3
satu
5
dua
2
tiga
3
satu
2
tiga
7
lima
3
satu
2
tiga
3
satu
1
tiga
3
satu
```



```
4
tiga
7
lima
3
satu
2
tiga
3
satu
1
tiga
3
satu
1
tiga
8
satu
1
tiga
11
satu
[Done] exited with code=0 in 1.366 seconds
```

## Deskripsi program

Program ini adalah implementasi sederhana dari sebuah linked list dalam bahasa C++. Linked list ini dapat melakukan berbagai operasi dasar seperti menyisipkan data di depan, belakang, atau di tengah linked list, menghapus data dari linked list, mengubah data, serta menampilkan isi linked list. Program ini menggunakan struct **Node** untuk merepresentasikan setiap elemen dalam linked list dan memiliki fungsi-fungsi untuk melakukan operasi-operasi tersebut. Pada fungsi **main()**, terdapat contoh penggunaan dari setiap operasi yang telah didefinisikan.

## 2. Guided 2

### Source code

```
#include <iostream>
using namespace std;
class Node
{
public:
    int data;
    string kata;
    Node *prev;
    Node *next;
};
class DoublyLinkedList
{
public:
    Node *head;
    Node *tail;
    DoublyLinkedList()
    {
        head = nullptr;
        tail = nullptr;
    }
    void push(int data, string kata)
    {
        Node *newNode = new Node;
        newNode->data = data;
        newNode->kata = kata;
        newNode->prev = nullptr;
        newNode->next = head;
        if (head != nullptr)
        {
            head->prev = newNode;
        }
        else
        {
            tail = newNode;
        }
        head = newNode;
    }
    void pop()
```

```

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

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

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

```



```

    }
    head = nullptr;
    tail = nullptr;
}
void display()
{
    Node *current = head;
    while (current != nullptr)
    {
        cout << current->data << " ";
        cout << current -> kata<<endl;
        current = current->next;
    }
    cout << endl;
}
};
int main()
{
    DoublyLinkedList list;
    while (true)
    {
        cout << "1. Add data" << endl;
        cout << "2. Delete data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Clear data" << endl;
        cout << "5. Display data" << endl;
        cout << "6. Exit" << endl;
        int choice;
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice)
        {
            case 1:
            {
                int data;
                string kata;
                cout << "Enter data to add: ";
                cin >> data;
                cout << "Enter kata to add: ";
                cin >> kata;
                list.push(data,kata);
            }
        }
    }
}

```

```

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

```

```

        default:
        {
            cout << "Invalid choice" << endl;
            break;
        }
    }
    return 0;
}

```

## Screenshoot program

```

1  #include <iostream>
2  using namespace std;
3  class Node
4  {
5  public:
6      int data;
7      string kata;
8      Node *prev;
9      Node *next;
10 };
11 class DoublyLinkedList
12 {
13 public:
14     Node *head;
15     Node *tail;
16     DoublyLinkedList()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

[Running] cd "c:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3\" && g++ guided2.cpp -o guided2 && "c:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3\guided2
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: |

```

## Deskripsi program

Program ini adalah sebuah aplikasi sederhana yang mengimplementasikan Doubly Linked List menggunakan C++. Program ini memiliki beberapa fitur utama:

- Add data: Pengguna dapat menambahkan data baru ke dalam Doubly Linked List. Setiap data terdiri dari dua nilai: integer (data) dan string (kata).
- Delete data: Pengguna dapat menghapus data dari Doubly Linked List. Pilihan ini menghapus data dari bagian depan (head) dari daftar.

- Update data: Pengguna dapat memperbarui nilai data dan kata dari sebuah node di dalam Doubly Linked List.
- Clear data: Menghapus semua data dari Doubly Linked List.
- Display data: Menampilkan semua data yang tersimpan dalam Doubly Linked List.
- Exit: Keluar dari program.

Program ini menggunakan menu pilihan yang memungkinkan pengguna untuk memilih operasi yang ingin dilakukan terhadap Doubly Linked List. Program akan terus berjalan dan menampilkan menu hingga pengguna memilih untuk keluar.

## LATIHAN KELAS - UNGUIDED

### 1. Unguided 1

#### Source code

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

struct Node
{
    string nama;
    int usia;
    Node *next;
};

Node *head = NULL;

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

void insertBack(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 insertMiddle(string nama, int usia, int posisi)
{
    if (posisi <= 1 || head == NULL)
    {
        insertFront(nama, usia);
        return;
    }

    Node *newNode = new Node;
    newNode->nama = nama;
    newNode->usia = usia;

    Node *temp = head;
    for (int i = 1; i < posisi - 1 && temp != NULL; ++i)
    {
        temp = temp->next;
    }

    if (temp == NULL)
    {
        cout << "Posisi diluar range." << endl;
        return;
    }

    newNode->next = temp->next;
    temp->next = newNode;
}

void displayList()
{
    Node *temp = head;
    while (temp != NULL)
    {
        cout << temp->nama << "\t" << temp->usia << endl;
    }
}

```

```

        temp = temp->next;
    }
}

int main()
{
    string Namamu;
    int Usiamu;

    cout << "Masukan nama anda: ";
    getline(cin, Namamu);
    cout << "Masukkan usia anda: ";
    cin >> Usiamu;

    insertFront(Namamu, Usiamu);

    cout << "Masukkan nama dan usia siswa lain: (Masukkan
'exit' untuk berhenti)" << endl;
    while (true)
    {
        string nama;
        int usia;

        cout << "Nama: ";
        cin.ignore();
        getline(cin, nama);
        if (nama == "exit")
            break;

        cout << "Usia: ";
        cin >> usia;

        insertBack(nama, usia);
    }

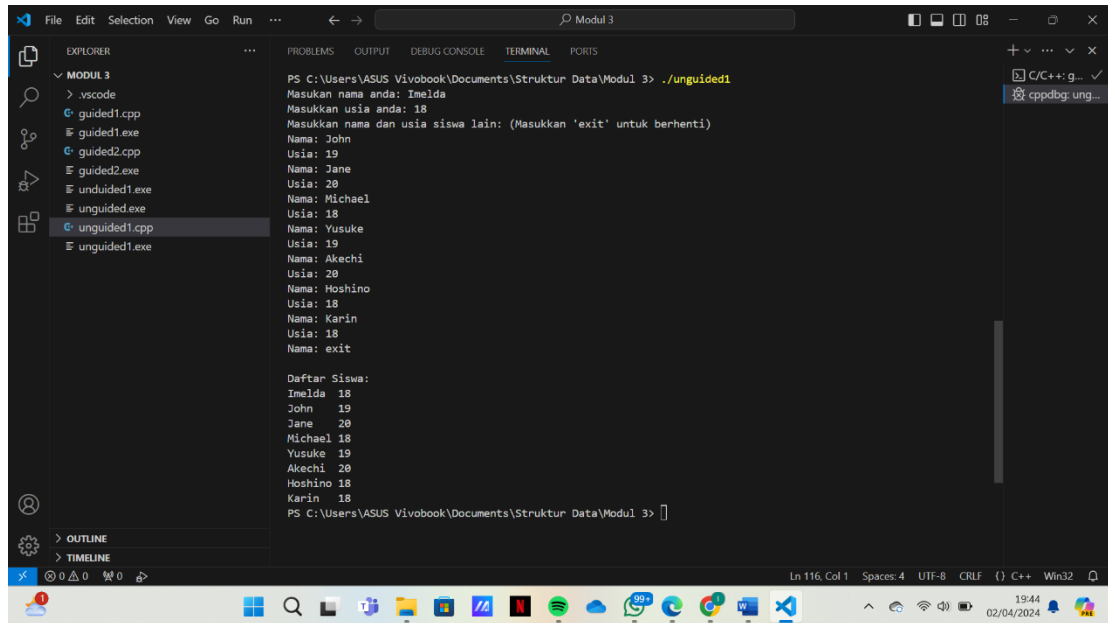
    cout << "\nDaftar Siswa:\n";
    displayList();

    return 0;
}

```

## Screenshoot program

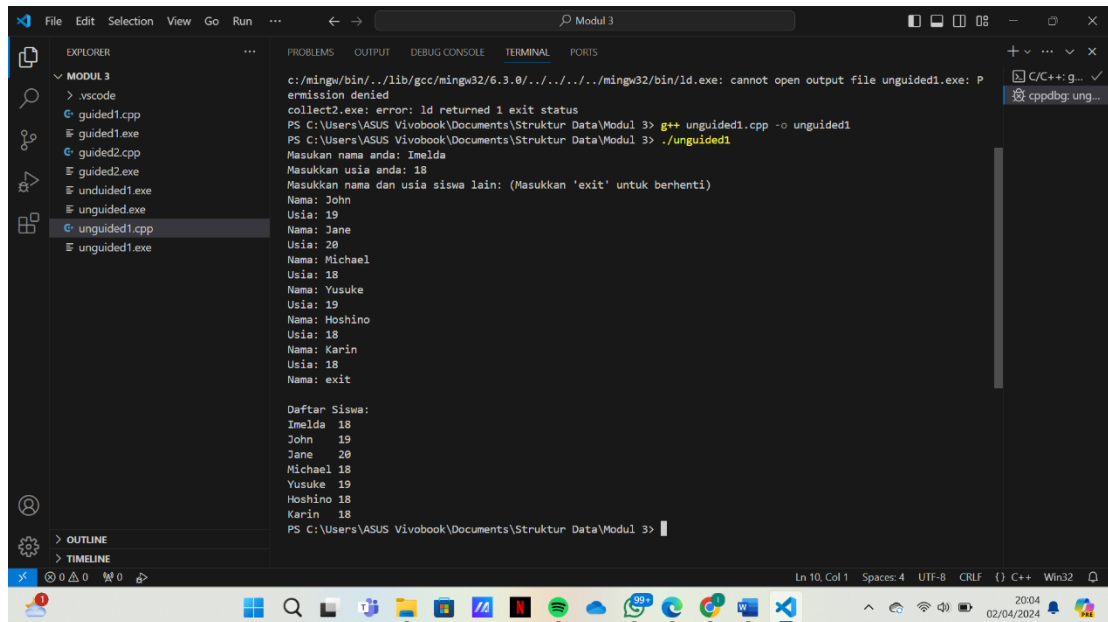
a.



```
PS C:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3> ./undguided1
Masukan nama anda: Imelda
Masukan usia anda: 18
Masukan nama dan usia siswa lain: (Masukkan 'exit' untuk berhenti)
Nama: John
Usia: 19
Nama: Jane
Usia: 20
Nama: Michael
Usia: 18
Nama: Yusuke
Usia: 19
Nama: Akechi
Usia: 20
Nama: Hoshino
Usia: 18
Nama: Karin
Usia: 18
Nama: exit

Daftar Siswa:
Imelda 18
John 19
Jane 20
Michael 18
Yusuke 19
Akechi 20
Hoshino 18
Karin 18
PS C:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3>
```

b. Hapus data Akechi



```
c:\mingw\bin\..\lib\gcc\mingw32\6.3.0\..\..\..\mingw32\bin\ld.exe: cannot open output file undguided1.exe: Permission denied
collect2.exe: error: ld returned 1 exit status
PS C:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3> g++ undguided1.cpp -o undguided1
PS C:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3> ./undguided1
Masukan nama anda: Imelda
Masukan usia anda: 18
Masukan nama dan usia siswa lain: (Masukkan 'exit' untuk berhenti)
Nama: John
Usia: 19
Nama: Jane
Usia: 20
Nama: Michael
Usia: 18
Nama: Yusuke
Usia: 19
Nama: Hoshino
Usia: 18
Nama: Karin
Usia: 18
Nama: exit

Daftar Siswa:
Imelda 18
John 19
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
PS C:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3>
```

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



```
PS C:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3> ./unguided1
Masukkan nama anda: Imelda
Masukkan usia anda: 18
Masukkan nama dan usia siswa lain: (Masukkan 'exit' untuk berhenti)
Nama: John
Usia: 19
Nama: Futaba
Usia: 18
Nama: Jane
Usia: 20
Nama: Michael
Usia: 18
Nama: Yusuke
Usia: 19
Nama: Hoshino
Usia: 18
Nama: Karin
Usia: 18
Nama: exit

Daftar Siswa:
Imelda 18
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
PS C:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3>
```

d. Tambahkan data berikut diawal : Igor 20

```
Masukkan nama anda: Imelda
Masukkan usia anda: 18
Masukkan nama dan usia siswa lain: (Masukkan 'exit' untuk berhenti)
Nama: Igor
Usia: 20
Nama: John
Usia: 19
Nama: Futaba
Usia: 18
Nama: Jane
Usia: 20
Nama: Michael
Usia: 18
Nama: Yusuke
Usia: 19
Nama: Hoshino
Usia: 18
Nama: Karin
Usia: 18
Nama: exit

Daftar Siswa:
Imelda 18
Igor 20
John 19
Futaba 18
Jane 20
Michael 18
Yusuke 19
Hoshino 18
Karin 18
PS C:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3>
```

e. Ubah data Michael menjadi : Reyn 18

```
File Edit Selection View Go Run ...  
MODUL 3  
> .vscode  
  guided1.cpp  
  guided1.exe  
  guided2.cpp  
  guided2.exe  
  undguided1.exe  
  undguided1.exe  
  undguided.exe  
  undguided.exe  
  undguided1.cpp  
  undguided1.exe  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
Masukan nama anda: Imelda  
Masukan usia anda: 18  
Masukan nama dan usia siswa lain: (Masukkan 'exit' untuk berhenti)  
Nama: Igor  
Usia: 20  
Nama: John  
Usia: 19  
Nama: Futaba  
Usia: 18  
Nama: Jane  
Usia: 20  
Nama: Reyn  
Usia: 18  
Nama: Yusuke  
Usia: 19  
Nama: Hoshino  
Usia: 18  
Nama: Karin  
Usia: 18  
Nama: exit  
Daftar Siswa:  
Imelda 18  
Igor 20  
John 19  
Futaba 18  
Jane 20  
Reyn 18  
Yusuke 19  
Hoshino 18  
Karin 18  
PS C:\Users\ASUS Vivobook\Documents\Struktur Data\Modul 3>  
Ln 18, Col 26 Spaces: 4 UTF-8 CRLF C++ Win32  
20:31 02/04/2024
```

## Deskripsi program

Program di atas adalah implementasi dari Single Linked List yang memungkinkan pengguna untuk memasukkan nama dan usia mereka, serta memasukkan nama dan usia siswa lainnya. Program kemudian menampilkan daftar nama dan usia siswa yang dimasukkan.

Deskripsi singkat dari program tersebut adalah sebagai berikut:

- Program meminta pengguna untuk memasukkan nama dan usia mereka.
- Data nama dan usia pengguna dimasukkan ke depan linked list menggunakan fungsi `insertFront`.
- Program meminta pengguna untuk memasukkan nama dan usia siswa lainnya dengan mengetikkan "exit" untuk mengakhiri proses input.
- Setiap data nama dan usia siswa yang dimasukkan oleh pengguna ditambahkan ke belakang linked list menggunakan fungsi `insertBack`.
- Setelah semua data dimasukkan, program menampilkan daftar nama dan usia siswa dengan menggunakan fungsi `displayList`.

Program ini menggunakan struktur data linked list untuk menyimpan dan mengelola data siswa dengan cara yang efisien dan fleksibel.

## 2. Unguided 2

### Source code

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

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 << "Nama Produk\tHarga" << endl;
    Node *current = head;
    while (current != NULL)
    {
        cout << current->nama << "\t\t" << 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 choice;
    string nama;
    int harga;
    int index;

```

```

do
{
    cout << "Menu:" << endl;
    cout << "1. Tambah Data" << endl;
    cout << "2. Hapus Data" << endl;
    cout << "3. Update Data" << endl;
    cout << "4. Tambah Data pada Urutan Tertentu" << endl;
    cout << "5. Hapus Data pada Urutan Tertentu" << endl;
    cout << "6. Hapus Semua Data" << endl;
    cout << "7. Tampilkan Data" << endl;
    cout << "8. Keluar" << endl;
    cout << "Pilih: ";
    cin >> choice;
    switch (choice)
    {
        case 1:
            for (int i = 0; i < 5; ++i)
            { cout << "Nama Produk " << i + 1 << ": ";
              cin >> nama;
              cout << "Harga " << i + 1 << ": ";
              cin >> harga;
              dll.addData(nama, harga);
            }
            break;
        case 2:
            cout << "Index: ";
            cin >> index;
            dll.deleteDataAt(index);
            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;

```

```

        cout << "Nama Produk: ";
        cin >> nama;
        cout << "Harga: ";
        cin >> harga;
        dll.addDataAt(index, nama, harga);
        break;
    case 5:
        cout << "Index: ";
        cin >> index;
        dll.deleteDataAt(index);
        break;
    case 6:
        dll.clearData();
        break;
    case 7:
        dll.displayData();
        break;
    case 8:
        break;
    default:
        cout << "Pilihan tidak valid" << endl;
        break;
    }
    cout << endl;
} while (choice != 8);
return 0;
}

```

### Screenshoot program

1. Tambahkan produk Azarine dengan harga 65000 diantaraSomethinc dan Skintific



```
Menu:
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 5
Index: 4

Menu:
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7
Nama Produk      Harga
Originote         60000
Somethinc         150000
Azarine           65000
Skintific         100000
Hanasui           30000

Menu:
1. Tambah Data
2. Hapus Data
```

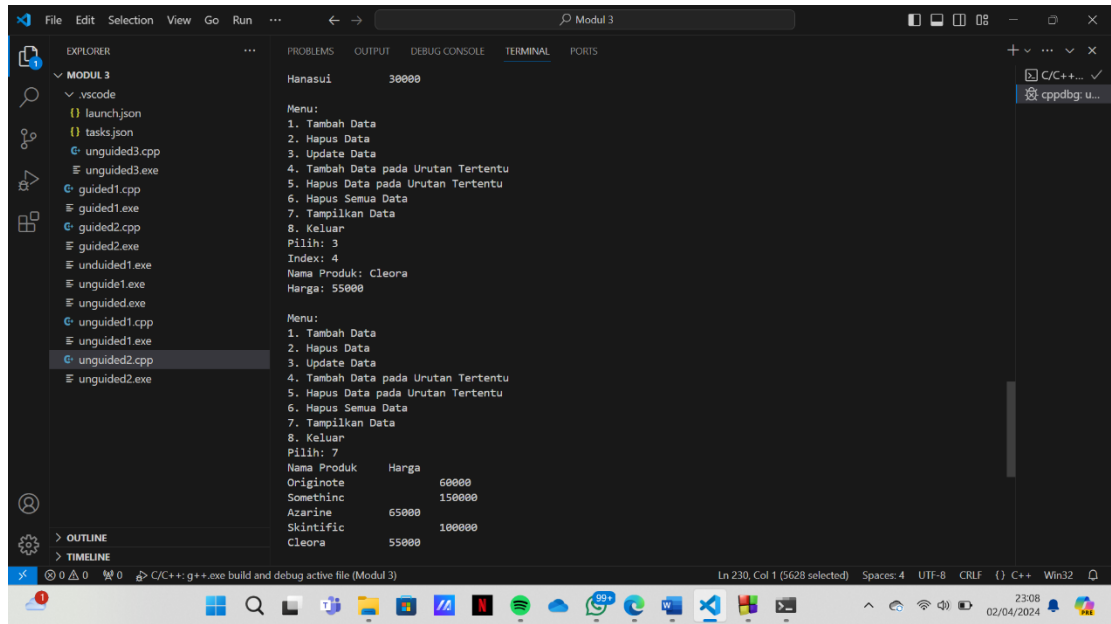
## 2. Hapus produk wardah

```
Menu:
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 4
Index: 2
Nama Produk: Azarine
Harga: 65000

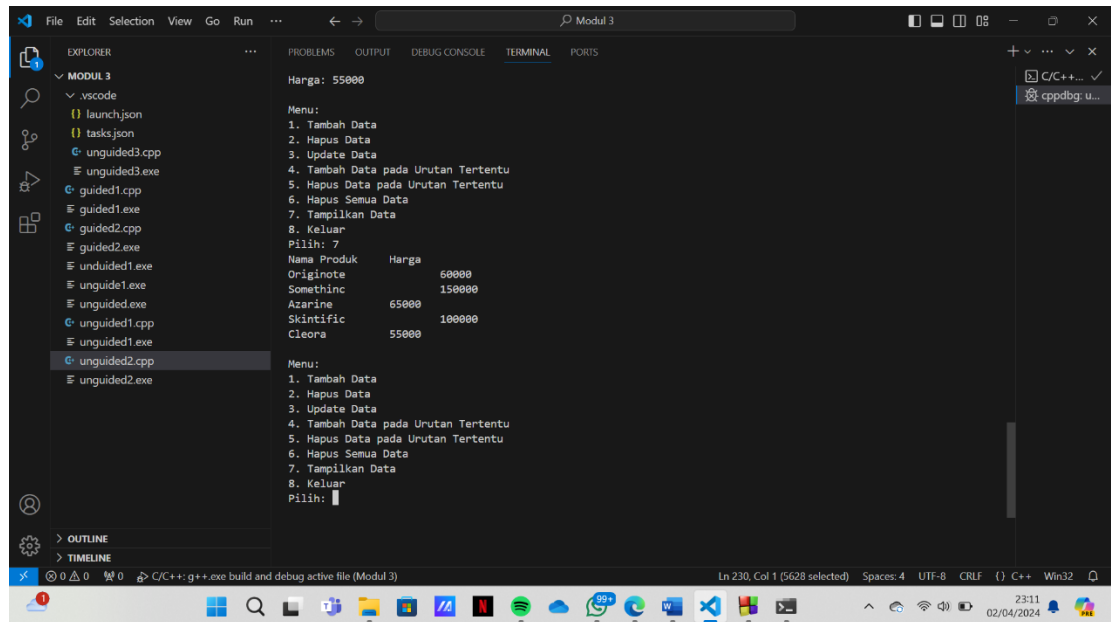
Menu:
1. Tambah Data
2. Hapus Data
3. Update Data
4. Tambah Data pada Urutan Tertentu
5. Hapus Data pada Urutan Tertentu
6. Hapus Semua Data
7. Tampilkan Data
8. Keluar
Pilih: 7
Nama Produk      Harga
Originote         60000
Somethinc         150000
Azarine           65000
Skintific         100000
Wardah            50000
Hanasui           30000

Menu:
```

## 3. Update produk Hanasui menjadi Cleora dengan harga 55.000



#### 4. Tampilkan menu



### Deskripsi program

Program ini adalah implementasi dari struktur data Double Linked List yang memungkinkan pengguna untuk melakukan operasi-operasi dasar seperti menambah, menghapus, memperbarui, dan menampilkan data produk. Program memiliki menu-menu interaktif yang memungkinkan pengguna untuk memilih operasi yang ingin dilakukan.

Pengguna dapat melakukan operasi-operasi berikut melalui menu yang disediakan:

- Menambah data produk.
- Menghapus data produk berdasarkan indeks.
- Memperbarui data produk berdasarkan indeks.
- Menambah data produk pada urutan tertentu.
- Menghapus data produk pada urutan tertentu.
- Menghapus semua data produk.
- Menampilkan semua data produk yang tersimpan.
- Keluar dari program.

Setiap operasi akan dieksekusi sesuai dengan pilihan pengguna dan program akan terus berjalan hingga pengguna memilih untuk keluar dari program.

## **BAB IV**

### **KESIMPULAN**

Dari praktikum diatas dapat kita simpulkan bahwa ada perbedaan antara Single Linked List dan Double Linked List yaitu :

- Single Linked List: Setiap node memiliki satu pointer yang menunjuk ke node berikutnya dalam list. Traversal hanya bisa dilakukan ke arah maju (dari depan ke belakang).
- Double Linked List: Setiap node memiliki dua pointer: satu untuk menunjuk ke node sebelumnya dan satu untuk menunjuk ke node berikutnya. Ini memungkinkan traversal maju dan mundur dalam list serta operasi-insert dan delete yang lebih efisien pada kedua ujung list.

Secara sangat singkat, penerapan Single Linked List dan Double Linked List dalam pemrograman melibatkan representasi data terhubung yang memungkinkan operasi-operasi seperti penambahan, penghapusan, dan penelusuran data. Single Linked List menggunakan satu pointer untuk setiap node, sementara Double Linked List menggunakan dua pointer untuk setiap node untuk mengizinkan penelusuran maju dan mundur.