

**LAPORAN PRAKTIKUM STRUKTUR
DATA DAN ALGORITMA**

**MODUL III
SINGLE AND DOUBLE LINKED LIST**



Disusun Oleh :

NAMA : GALIH TRISNA
NIM : 2311102050

Dosen

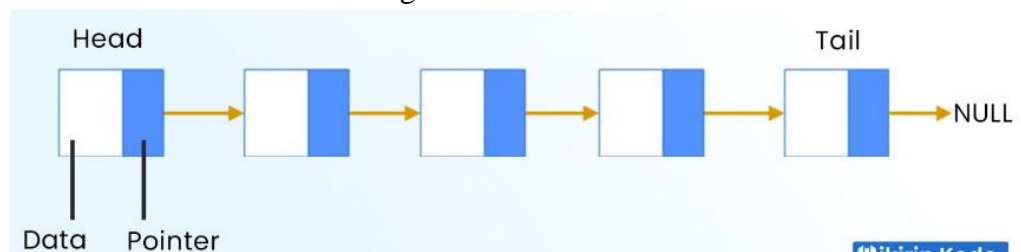
Wahyu Andi Saputra, S.Pd., M.Eng

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFROMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

A. Dasar Teori

a. Single Linked List

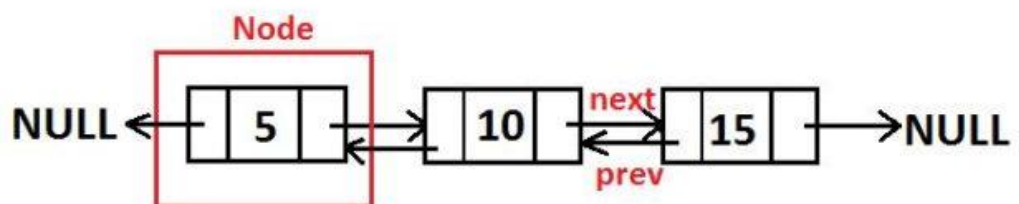
Single Linked List atau Singly Linked List merupakan linked list yang hanya memiliki satu variabel pointer untuk menunjuk ke node lainnya, variabel pointer ini biasanya dinamakan next. Pada dasarnya Single Linked List ini adalah linked list yang berbentuk umum seperti yang dijelaskan sebelumnya. Seperti yang dapat dilihat dari ilustrasi di atas, terdapat 5 node yang terhubung menjadi suatu single linked list. Node pertama biasa disebut head, pointer pada node ini menunjuk ke node selanjutnya dan begitu seterusnya hingga node terakhir yang pointernya menunjuk ke NULL. Hal itu menandakan bahwa node itu adalah node terakhir atau biasa disebut dengan node tail.



b. Double Linked List

Dalam pembahasan artikel sebelumnya telah diperkenalkan Single Linked List, yakni linked list dengan sebuah pointer penghubung. Dalam artikel ini, dibahas pula varian linked list dengan 2 pointer penunjuk, yakni Doubly linked list yang memiliki pointer penunjuk 2 arah, yakni ke arah node sebelum (previos/prev) dan node sesudah (next).

Representasi sebuah doubly 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.

B. Guided

Guided 1

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

struct Node
{
    int data;
    Node *next;
};

Node *head;
Node *tail;

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

bool isEmpty()
{
    return head == NULL;
}

void insertDepan(int nilai)
{
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}

void insertBelakang(int nilai)
```

```

{
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

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

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

```

```

        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

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

void hapusBelakang()
{
    if (!isEmpty())
    {
        if (head != tail)
        {
            Node *hapus = tail;
            Node *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)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *hapus;
        Node *bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++)
        {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

void ubahDepan(int data)
{
    if (!isEmpty())
    {
        head->data = data;
    }
}

```

```

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

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

void ubahBelakang(int data)
{
    if (!isEmpty())
    {
        tail->data = data;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

```



```

}

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

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

int main()
{
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
}

```

```

    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();
    return 0;
}

```

Screenshots Output

```

PS C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\270324> cd "c:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\270324\" ; if ($?) { g++ guided1.cpp -o guided1 } ; if ($?) { .\guided1 }
3
3 5
2 3 5
1 2 3 5
2 3 5
2 3
2 7 3
2 3
1 3
1 8
1 11
PS C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\270324>

```

Deskripsi: Program di atas adalah sebuah program untuk mengelola sebuah Single linked list yang berisi bilangan bulat. Program ini dapat melakukan beberapa operasi dasar seperti menambahkan data di depan, belakang, atau di tengah linked list, menghapus data di depan, belakang, atau di tengah linked list, mengubah nilai data di depan, belakang, atau di tengah linked list, menampilkan isi linked list, dan menghapus semua data linked list. Data pada linked list ini terdiri dari sebuah struct Node yang berisi int data untuk menyimpan data bilangan bulat dan Node *next untuk menunjukkan ke data selanjutnya dalam linked list.

Dalam program ini terdapat beberapa fungsi yaitu:

- `init()`: Menginisialisasi head dan tail linked list menjadi NULL.

- isEmpty(): Mengembalikan true jika linked list kosong.
- insertDepan(int nilai): Menambahkan data baru di depan linked list.
- insertBelakang(int nilai): Menambahkan data baru di belakang linked list.
- hitungList(): Menghitung jumlah data dalam linked list.
- insertTengah(int data, int posisi): Menambahkan data baru di posisi tertentu dalam linked list.
- hapusDepan(): Menghapus data pertama dalam linked list.
- hapusBelakang(): Menghapus data terakhir dalam linked list.
- hapusTengah(int posisi): Menghapus data pada posisi tertentu dalam linked list.
- ubahDepan(int data): Mengubah data pertama dalam linked list.
- ubahTengah(int data, int posisi): Mengubah data pada posisi tertentu dalam linked list.
- ubahBelakang(int data): Mengubah data terakhir dalam linked list.
- clearList(): Menghapus semua data dalam linked list.
- tampil(): Menampilkan seluruh data dalam linked list.

semua fungsi diatas digunakan didalam fungsi main() untuk menjalankan berbagai operasi pada linked list seperti menambahkan, menghapus, mengubah, dan menampilkan data-data dalam linked list.

Guided 2

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

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

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

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

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

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

        head = newNode;
    }
}
```

```

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

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

    delete temp;
}

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

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

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

```

```

        delete temp;
    }
    head = nullptr;
    tail = nullptr;
}

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

};

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

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

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

```

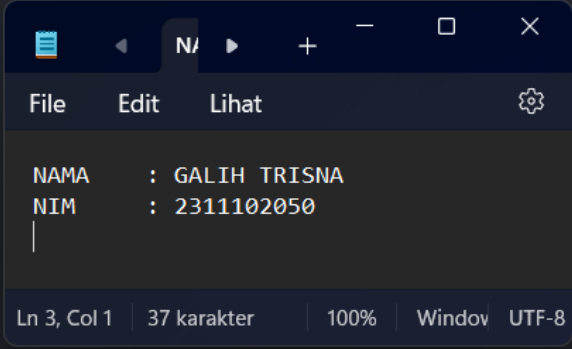
```

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

```

Screenshots Output

```
PS C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\270324> cd "C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\270324\" ; if ($?) { g++ guided2.cpp -o guided2 } ; if ($?) { .\guided2 }  
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: 23  
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: 20  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit  
Enter your choice: 3  
Enter old data: 20  
Enter new data: 90  
1. Add data  
2. Delete data  
3. Update data  
4. Clear data  
5. Display data  
6. Exit
```

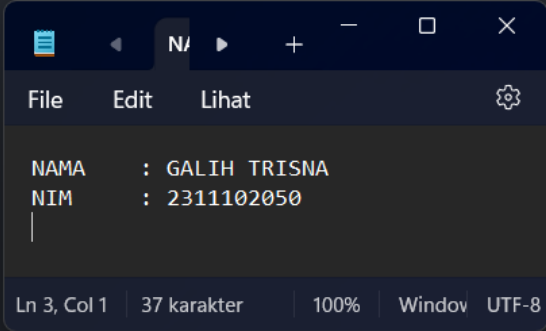


The screenshot shows a Notepad++ window with the following content:

NAMA	: GALIH TRISNA
NIM	: 2311102050

The status bar at the bottom of the window displays: Ln 3, Col 1 | 37 karakter | 100% | Window | UTF-8.


```
07 Exit
Enter your choice: 2
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 5
23
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 2
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 4
1. Add data
2. Delete data
3. Update data
4. Clear data
5. Display data
6. Exit
Enter your choice: 6
PS C:\Users\galih\OneDrive\Dokumen\belajar\Struktur data dan algoritma\270324>
```



The screenshot shows a Notepad window with a dark theme. The title bar reads 'N'. The menu bar includes 'File', 'Edit', 'Lihat', and a settings icon. The text content is as follows:

NAMA	:	GALIH TRISNA
NIM	:	2311102050

Below the text, the status bar shows 'Ln 3, Col 1', '37 karakter', '100%', 'Window', and 'UTF-8'.

Deskripsi: Program di atas adalah sebuah program yang mengimplementasikan Doubly Linked List menggunakan class. Program ini dapat melakukan beberapa operasi dasar pada Doubly Linked List, seperti menambahkan data, menghapus data, mengubah data, menghapus semua data, dan menampilkan data.

Didalam program terdapat 2 class yaitu:

- Node: class ini merepresentasikan simpul atau node dalam Doubly Linked List. Setiap node memiliki tiga anggota data yaitu int data untuk menyimpan nilai data, Node *prev untuk menunjukkan ke simpul sebelumnya, dan Node *next untuk menunjukkan ke simpul berikutnya.
- DoublyLinkedList: class ini berisi Doubly Linked List. Class ini memiliki dua anggota data yaitu Node *head yang menunjukkan ke simpul pertama dalam linked list, dan Node *tail yang menunjukkan ke simpul terakhir dalam linked list. Class ini juga dapat melakukan operasi pada linked list, seperti menambahkan data (push), menghapus data (pop), mengubah data (update), menghapus semua data (deleteAll), dan menampilkan data (display).

Di dalam fungsi `main()`, program menampilkan menu untuk pengguna. Pengguna dapat memilih menu untuk menambahkan data baru, menghapus data, mengubah data, menghapus semua data, menampilkan data, atau keluar dari program. Program ini menggunakan switch-case untuk membuat menu dan menjalankan pilihan pengguna. Berikut adalah pilihan yang terdapat pada menu

- Add data: Menambahkan data baru ke depan Doubly Linked List.
- Delete data: Menghapus data dari depan Doubly Linked List.
- Update data: Mengubah data tertentu dalam Doubly Linked List.
- Clear data: Menghapus seluruh data dari Doubly Linked List.
- Display data: Menampilkan seluruh data dalam Doubly Linked List.
- Exit: Keluar dari program.

Program menggunakan perulangan `do while` yang tidak akan berhenti mengulang sampai mendapatkan inputan dari pengguna untuk keluar program.

C. Unguided/Tugas

Unguided 1

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

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

Node *head;
Node *tail;

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

bool isEmpty()
{
    return head == NULL;
}

void insertDepan(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}
```

```
void insertBelakang(string nama, int usia)
{
    Node *baru = new Node;
    baru->nama = nama;
    baru->usia = usia;
    baru->next = NULL;
    if (isEmpty())
    {
        head = tail = baru;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}

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

void insertTengah(string nama, int usia, 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 = new Node();
        baru->nama = nama;
```

```

        baru->usia = usia;
        Node *bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

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

void hapusBelakang()
{
    if (!isEmpty())
    {
        if (head != tail)
        {
            Node *hapus = tail;
            Node *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)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *hapus;
        Node *bantu = head;
        for (int nomor = 1; nomor < posisi - 1; nomor++)
        {
            bantu = bantu->next;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
}

void ubahDepan(string nama, int usia)
{

```

```

    if (!isEmpty())
    {
        head->nama = nama;
        head->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

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

void ubahBelakang(string nama, int usia)
{
    if (!isEmpty())
    {

```

```

        tail->nama = nama;
        tail->usia = usia;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

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

void tampil()
{
    if (!isEmpty())
    {
        Node *bantu = head;
        while (bantu != NULL)
        {
            cout << bantu->nama << " ";
            cout << bantu->usia << " , ";
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    init();

```



```

int menu, usia, posisi;
string nama;
cout << "\n# Menu Linked List Mahasiswa #" << endl;
do
{
    cout << "\n 1. Insert Depan"
        << "\n 2. Insert Belakang"
        << "\n 3. Insert Tengah"
        << "\n 4. Hapus Depan"
        << "\n 5. Hapus Belakang"
        << "\n 6. Hapus Tengah"
        << "\n 7. Ubah Depan"
        << "\n 8. Ubah Belakang"
        << "\n 9. Ubah Tengah"
        << "\n 10. Tampilkan"
        << "\n 0. Keluar Program"
        << "\n Pilihan : ";

    cin >> menu;
    switch (menu)
    {
        case 1:
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";
            cin >> usia;
            insertDepan(nama, usia);
            cout << endl;
            tampil();
            break;
        case 2:
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";
            cin >> usia;
            insertBelakang(nama, usia);
            cout << endl;
            tampil();
            break;
        case 3:
            cout << "Masukkan Posisi : ";
            cin >> posisi;
            cout << "Masukkan Nama : ";
            cin >> nama;
            cout << "Masukkan Usia : ";

```

```
        cin >> usia;
        insertTengah(nama, usia, posisi);
        cout << endl;
        tampil();
        break;
    case 4:
        hapusDepan();
        cout << endl;
        tampil();
        break;
    case 5:
        hapusBelakang();
        cout << endl;
        tampil();
        break;
    case 6:
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        hapusTengah(posisi);
        cout << endl;
        tampil();
        break;
    case 7:
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahDepan(nama, usia);
        cout << endl;
        tampil();
        break;
    case 8:
        cout << "Masukkan Nama : ";
        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahBelakang(nama, usia);
        cout << endl;
        tampil();
        break;
    case 9:
        cout << "Masukkan Posisi : ";
        cin >> posisi;
        cout << "Masukkan Nama : ";
```

```

        cin >> nama;
        cout << "Masukkan Usia : ";
        cin >> usia;
        ubahTengah(nama, usia, posisi);
        cout << endl;
        tampil();
        break;
    case 10:
        tampil();
        break;

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

```

Screenshots Output

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

```

1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 1
Masukkan Nama : Galih
Masukkan Usia : 18

Galih 18 , John 19 , Jane 20 , Michael 18 , Yusuke 19 , Akechi 20 , Hoshino 18 , Karin 18 ,

```

File Edit Lihat
⚙

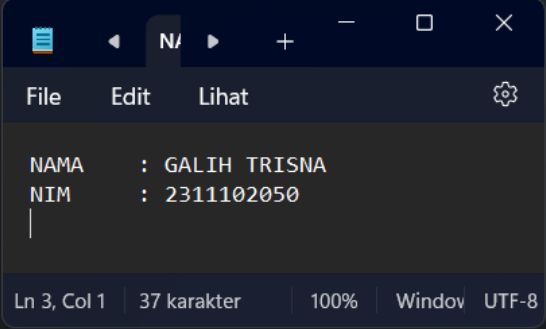
NAMA : GALIH TRISNA
NIM : 2311102050
|

Ln 3, Col 1 | 37 karakter | 100% | Window UTF-8

b. Hapus data Akechi

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 6
Masukkan Posisi : 6

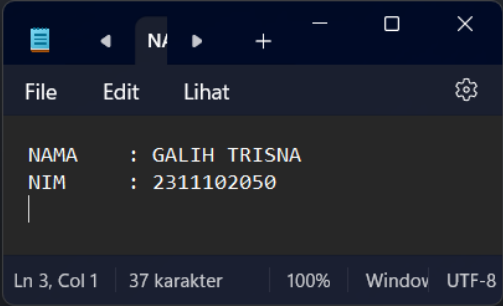
Galih 18 , John 19 , Jane 20 , Michael 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,
```



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

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 3
Masukkan Posisi : 3
Masukkan Nama : Futaba
Masukkan Usia : 18

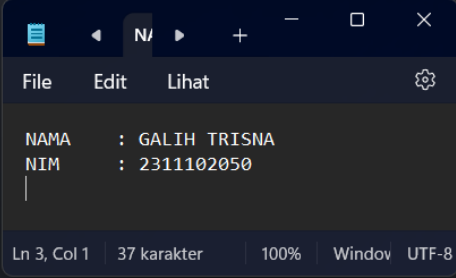
Galih 18 , John 19 , Futaba 18 , Jane 20 , Michael 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,
```



d. Tambahkan data berikut diawal : Igor 20

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 1
Masukkan Nama : Igor
Masukkan Usia : 20

Igor 20 , Galih 18 , John 19 , Futaba 18 , Jane 20 , Michael 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,
```



e. Ubah data Michael menjadi : Reyn 18

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 9
Masukkan Posisi : 6
Masukkan Nama : Reyn
Masukkan Usia : 18

Igor 20 , Galih 18 , John 19 , Futaba 18 , Jane 20 , Reyn 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,
```

f. Tampilkan seluruh data

```
1. Insert Depan
2. Insert Belakang
3. Insert Tengah
4. Hapus Depan
5. Hapus Belakang
6. Hapus Tengah
7. Ubah Depan
8. Ubah Belakang
9. Ubah Tengah
10. Tampilkan
0. Keluar Program
Pilihan : 10

Igor 20 , Galih 18 , John 19 , Futaba 18 , Jane 20 , Reyn 18 , Yusuke 19 , Hoshino 18 , Karin 18 ,
```

Deskripsi: Program di atas adalah program untuk mengelola linked list yang berisi data mahasiswa. Setiap node dalam linked list memiliki dua atribut yaitu nama (string) dan usia (integer). Program ini dapat melakukan beberapa operasi dasar pada linked list seperti menambahkan data di depan, belakang, atau di tengah linked list, menghapus data dari depan, belakang, atau dari posisi tertentu, mengubah data di depan, belakang, atau di tengah linked list, dan menampilkan semua data dalam linked list. Berikut adalah fungsi fungsi yang ada didalam program diatas :

- `init()`: Inisialisasi pointer head dan tail menjadi NULL untuk menandakan bahwa linked list masih kosong.
- `isEmpty()`: Fungsi yang mengembalikan nilai boolean true jika linked list kosong.
- `insertDepan(string nama, int usia)`: Menambahkan data mahasiswa baru di depan linked list.
- `insertBelakang(string nama, int usia)`: Menambahkan data mahasiswa baru di belakang linked list.

- `hitungList()`: Menghitung jumlah data dalam linked list.
- `insertTengah(string nama, int usia, int posisi)`: Menambahkan data mahasiswa baru pada posisi tertentu dalam linked list.
- `hapusDepan()`: Menghapus data mahasiswa dari depan linked list.
- `hapusBelakang()`: Menghapus data mahasiswa dari belakang linked list.
- `hapusTengah(int posisi)`: Menghapus data mahasiswa dari posisi tertentu dalam linked list.
- `ubahDepan(string nama, int usia)`: Mengubah data mahasiswa di depan linked list.
- `ubahTengah(string nama, int usia, int posisi)`: Mengubah data mahasiswa pada posisi tertentu dalam linked list.
- `ubahBelakang(string nama, int usia)`: Mengubah data mahasiswa di belakang linked list.
- `clearList()`: Menghapus semua data dalam linked list.
- `tampil()`: Menampilkan semua data mahasiswa dalam linked list.

Di dalam fungsi `main()`, program menampilkan menu pilihan untuk pengguna. Pengguna dapat memilih pilihan untuk Menambahkan data baru di depan, belakang, atau di tengah linked list, menghapus data, mengubah data, atau menampilkan semua data dalam linked list. Program menggunakan perulangan `do while` yang akan terus berulang sampai mendapatkan inputan dari pengguna untuk keluar dari program (memilih pilihan 0).

Unguided 2

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

class Node
{
public:
    string namaProduk;
    int harga;
    Node *prev;
    Node *next;
};

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

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

    void push(string namaProduk, int harga)
    {
        Node *newNode = new Node;
        newNode->namaProduk = namaProduk;
        newNode->harga = harga;
        newNode->prev = nullptr;
        newNode->next = head;

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

```

        head = newNode;
    }

    void pushCenter(string namaProduk, int harga, int posisi)
    {
        if (posisi < 0)
        {
            cout << "Posisi harus bernilai non-negatif." <<
endl;
            return;
        }

        Node *newNode = new Node;
        newNode->namaProduk = namaProduk;
        newNode->harga = harga;

        if (posisi == 0 || head == nullptr)
        {
            newNode->prev = nullptr;
            newNode->next = head;

            if (head != nullptr)
            {
                head->prev = newNode;
            }
            else
            {
                tail = newNode;
            }
            head = newNode;
        }
        else
        {
            Node *temp = head;
            int count = 0;
            while (temp != nullptr && count < posisi)
            {
                temp = temp->next;
                count++;
            }

            if (temp == nullptr)
            {

```



```

        newNode->prev = tail;
        newNode->next = nullptr;
        tail->next = newNode;
        tail = newNode;
    }
    else
    {
        newNode->prev = temp->prev;
        newNode->next = temp;
        temp->prev->next = newNode;
        temp->prev = newNode;
    }
}

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

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

    delete temp;
}

void popCenter(int posisi)
{
    if (head == nullptr)
    {
        cout << "List kosong. Tidak ada yang bisa dihapus."
<< endl;
        return;
    }

```

```

        if (posisi < 0)
        {
            cout << "Posisi harus bernilai non-negatif." <<
endl;
            return;
        }

        if (posisi == 0)
        {
            Node *temp = head;
            head = head->next;

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

            delete temp;
        }
        else
        {
            Node *temp = head;
            int count = 0;
            while (temp != nullptr && count < posisi)
            {
                temp = temp->next;
                count++;
            }

            if (temp == nullptr)
            {
                cout << "Posisi melebihi ukuran list. Tidak ada
yang dihapus." << endl;
                return;
            }

            if (temp == tail)
            {
                tail = tail->prev;
            }
        }
    }
}

```

```

        tail->next = nullptr;
        delete temp;
    }
    else
    {
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
        delete temp;
    }
}

bool update(string oldNamaProduk, string newNamaProduk, int
newHarga)
{
    Node *current = head;

    while (current != nullptr)
    {
        if (current->namaProduk == oldNamaProduk)
        {
            current->namaProduk = newNamaProduk;
            current->harga = newHarga;
            return true;
        }
        current = current->next;
    }
    return false;
}

bool updateCenter(string newNamaProduk, int newHarga, int
posisi)
{
    if (head == nullptr)
    {
        cout << "List kosong. Tidak ada yang dapat
diperbarui." << endl;
        return false;
    }

    if (posisi < 0)
    {
        cout << "Posisi harus bernilai non-negatif." <<
endl;

```

```

        return false;
    }

    Node *current = head;
    int count = 0;

    while (current != nullptr && count < posisi)
    {
        current = current->next;
        count++;
    }

    if (current == nullptr)
    {
        cout << "Posisi melebihi ukuran list. Tidak ada
yang diperbarui." << endl;
        return false;
    }

    current->namaProduk = newNamaProduk;
    current->harga = newHarga;
    return true;
}

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

void display()
{
    if (head == nullptr)
    {
        cout << "List kosong." << endl;
        return;
    }

```

```

        Node *current = head;

        cout << setw(37) << setfill('-') << "-" << setfill(' ')
<< endl;
        cout << "| " << setw(20) << left << "Nama Produk"
            << " | " << setw(10) << "Harga"
            << " |" << endl;
        cout << setw(37) << setfill('-') << "-" << setfill(' ')
<< endl;

        while (current != nullptr)
        {
            cout << "| " << setw(20) << left << current-
>namaProduk << " | " << setw(10) << current->harga << " |" <<
endl;

            current = current->next;
        }
        cout << setw(37) << setfill('-') << "-" << setfill(' ')
<< endl;
    }
};

int main()
{
    DoublyLinkedList list;
    int choice;
    cout << endl
        << "Toko Skincare Purwokerto" << endl;
    do
    {
        cout << "1. Tambah data" << endl;
        cout << "2. Hapus data" << endl;
        cout << "3. Update data" << endl;
        cout << "4. Tambah Data Urutan Tertentu" << endl;
        cout << "5. Hapus Data Urutan Tertentu" << endl;
        cout << "6. Hapus Seluruh Data" << endl;
        cout << "7. Tampilkan data" << endl;
        cout << "8. Exit" << endl;

        cout << "Pilihan : ";
        cin >> choice;

        switch (choice)

```

```

{
case 1:
{
    string namaProduk;
    int harga;
    cout << "Masukkan nama produk: ";
    cin.ignore();
    getline(cin, namaProduk);
    cout << "Masukkan harga produk: ";
    cin >> harga;
    list.push(namaProduk, harga);
    break;
}
case 2:
{
    list.pop();
    break;
}
case 3:
{
    string newNamaProduk;
    int newHarga, posisi;
    cout << "Masukkan posisi produk: ";
    cin >> posisi;
    cout << "Masukkan nama baru produk: ";
    cin >> newNamaProduk;
    cout << "Masukkan harga baru produk: ";
    cin >> newHarga;
    bool updatedCenter =
list.updateCenter(newNamaProduk, newHarga, posisi);
    if (!updatedCenter)
    {
        cout << "Data not found" << endl;
    }
    break;
}
case 4:
{
    string namaProduk;
    int harga, posisi;
    cout << "Masukkan posisi data produk: ";
    cin >> posisi;
    cout << "Masukkan nama produk: ";
    cin.ignore();

```

```

        getline(cin, namaProduk);
        cout << "Masukkan harga produk: ";
        cin >> harga;
        list.pushCenter(namaProduk, harga, posisi);
        break;
    }
    case 5:
    {
        int posisi;
        cout << "Masukkan posisi data produk: ";
        cin >> posisi;
        list.popCenter(posisi);
        break;
    }
    case 6:
    {
        list.deleteAll();
        break;
    }
    case 7:
    {
        list.display();
        break;
    }
    case 8:
    {
        return 0;
    }
    default:
    {
        cout << "Invalid choice" << endl;
        break;
    }
}
} while (choice != 8);

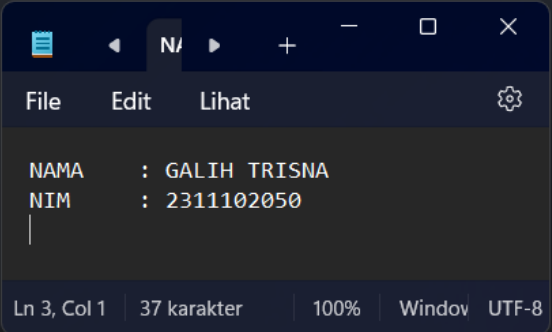
return 0;
}

```

Screenshots Output

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

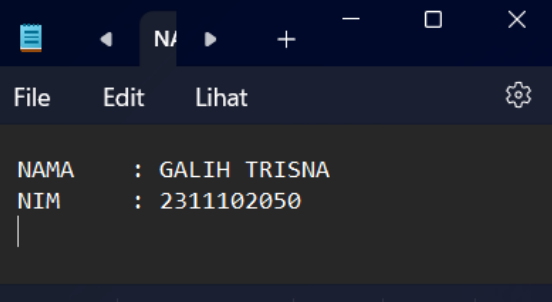
```
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 4
Masukkan posisi data produk: 2
Masukkan nama produk: Azarine
Masukkan harga produk: 65000
1. Tambah data
```



Ln 3, Col 1 | 37 karakter | 100% | Window UTF-8

- b. Hapus produk wardah

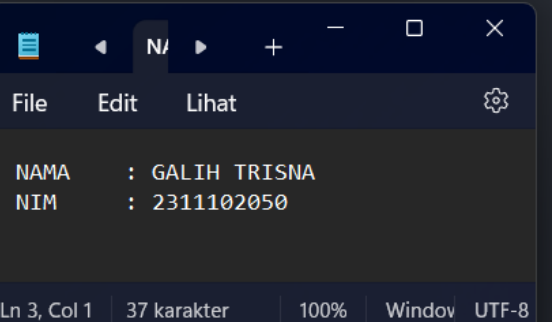
```
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 5
Masukkan posisi data produk: 4
```



Ln 3, Col 1 | 37 karakter | 100% | Window UTF-8

- c. Update produk Hanasui menjadi Cleora dengan harga 55.000

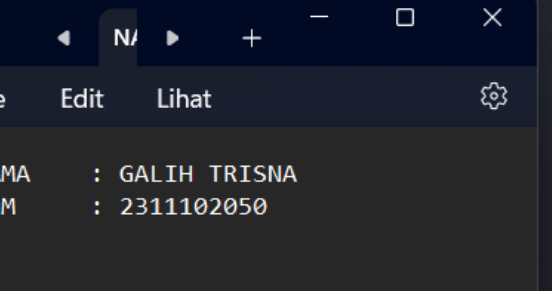
```
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
Pilihan : 3
Masukkan posisi produk: 4
Masukkan nama baru produk: Cleora
Masukkan harga baru produk: 55000
```



Ln 3, Col 1 | 37 karakter | 100% | Window UTF-8

- d. Tampilkan menu

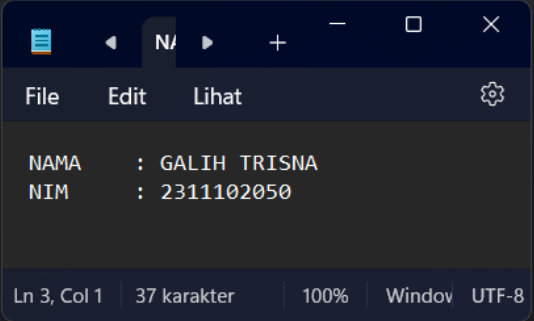
```
Toko Skincare Purwokerto
1. Tambah data
2. Hapus data
3. Update data
4. Tambah Data Urutan Tertentu
5. Hapus Data Urutan Tertentu
6. Hapus Seluruh Data
7. Tampilkan data
8. Exit
```



Ln 3, Col 1 | 37 karakter | 100% | Window UTF-8


```
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
Pilihan : 7
```

Nama Produk	Harga
Originote	60000
Somethinc	150000
Azarine	65000
Skintific	100000
Cleora	55000



Deskripsi: Program di atas adalah sebuah program untuk mengelola daftar produk skincare di sebuah toko. Program ini menggunakan doubly linked list untuk menyimpan data setiap produk, seperti nama produk dan harganya. Berikut adalah fungsi fungsi yang ada di dalam program diatas:

- `push(string namaProduk, int harga)`: Menambahkan sebuah node baru ke dalam linked list di bagian depan.
- `pushCenter(string namaProduk, int harga, int posisi)`: Menambahkan sebuah node baru ke dalam linked list pada posisi tertentu.
- `pop()`: Menghapus node teratas dari linked list.
- `popCenter(int posisi)`: Menghapus node pada posisi tertentu dari linked list.
- `update(string oldNamaProduk, string newNamaProduk, int newHarga)`: Memperbarui data dari produk skincare berdasarkan nama produk.
- `updateCenter(string newNamaProduk, int newHarga, int posisi)`: Memperbarui data dari produk skincare pada posisi tertentu dalam linked list.
- `deleteAll()`: Menghapus semua node dari linked list.
- `display()`: Menampilkan semua data produk skincare yang tersimpan dalam linked list.

Di dalam fungsi `main()`, program memiliki menu untuk dipilih oleh pengguna. Pengguna dapat menambahkan produk baru, menghapus produk, memperbarui informasi produk, menambahkan produk pada posisi tertentu, menghapus produk pada posisi tertentu, menghapus semua produk, dan menampilkan daftar produk yang ada. Program akan terus berjalan kecuali jika pengguna memilih untuk keluar (pilihan 8).

D. Kesimpulan

Single Linked List adalah struktur data yang terdiri dari serangkaian node yang terhubung satu sama lain melalui pointer 'next', dimana node pertama disebut head dan node terakhir menunjuk ke NULL. Sedangkan Doubly Linked List adalah varian dari linked list yang memiliki dua pointer penunjuk, yaitu ke node sebelumnya ('previous/prev') dan ke node berikutnya ('next'). Doubly Linked List memungkinkan traversing maju dan mundur, memudahkan operasi seperti penghapusan atau penambahan node di antara dua node yang ada. Di kedua jenis linked list ini, HEAD menunjuk pada node pertama dan TAIL menunjuk pada node terakhir, dengan nilai NULL menandakan linked list kosong. Perbedaan utama antara keduanya adalah kemampuan Doubly Linked List untuk bergerak maju dan mundur, sementara Single Linked List hanya bergerak maju.

E. Referensi

Mikirin Kode. (n.d.). Single Linked List. Diakses pada 29 Maret 2024, dari <https://mikirinkode.com/single-linked-list/>

Sekolah Ilmu Komputer (SICS) Binus University. (2017, 15 Maret). Doubly Linked List. Diakses pada 29 Maret 2024, dari <https://socs.binus.ac.id/2017/03/15/doubly-linked-list/>