

# **TUGAS STRUKTUR DATA**

*Tugas Ini Dibuat Guna Memenuhi Tugas Struktur Data*

**Dosen pengampu:**

**Adam bachtiar, s.kom, M.MT**



**Disusun Oleh :**

**Alief Hikham Pratama**

**NIM : 24241049**

**PROGRAM STUDI PENDIDIKAN TEKNOLOGI INFORMASI**

**FAKULTAS SAINS, TEHNIK DAN TERAPAN**

**UNIVERSITAS PENDIDIKAN MANDALIKA MATARAM**

**2025**

```
Welcome  array.py  praktek22.py  modul2B.py X  main.py
modul2B.py > ...
7  class DoubleLinkedList:
55  def delete_berdasarkan_nilai(self, target):
73      else:
74          # Node terakhir
75          curr.prev.next = None
76          return
77          curr = curr.next
78      print(f>Data {target} tidak ditemukan dalam linked list.")
79
80  # Contoh penggunaan
81  dll = DoubleLinkedList()
82  dll.append(10)
83  dll.append(20)
84  dll.append(30)
85  dll.append(40)
86
87  print("linked list awal:")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\elsan\OneDrive\Dokumen\modul 2> & C:/Users/elsan/AppData/Local/Micr
Linked list awal:
10 <-> 20 <-> 30 <-> 40 <-> None

Hapus node awal:
20 <-> 30 <-> 40 <-> None

Hapus node akhir:
20 <-> 30 <-> None

Hapus node dengan nilai 20:
30 <-> None

Coba hapus data yang tidak ada (50):
Data 50 tidak ditemukan dalam linked list.
PS C:\Users\elsan\OneDrive\Dokumen\modul 2>
```

## 1. Kelas Node

class Node:

```
def __init__(self, data):
```

```
    self.data = data
```

```
    self.prev = None
```

```
    self.next = None
```

- Node merepresentasikan **satu simpul (node)** dalam linked list.
- data: Menyimpan nilai data.
- prev: Menunjuk ke node sebelumnya.
- next: Menunjuk ke node berikutnya.
- Saat node dibuat, prev dan next di-set ke None.

### 1. Kelas DoubleLinkedList

class DoubleLinkedList:

```
def __init__(self):
```

```
    self.head = None
```

- head adalah pointer ke node pertama dalam linked list. Awalnya None (kosong).

### 2. Metode append(data)

```
def append(self, data):
```

```
    new_node = Node(data)
```

- Membuat node baru dengan data data.

```
    if not self.head:
```

```
        self.head = new_node
```

```
    return
```

- Jika list kosong (head masih None), maka node baru menjadi head.

```
    curr = self.head
```

```
    while curr.next:
```

```
        curr = curr.next
```

- Menelusuri sampai node terakhir (curr.next == None).

```
    curr.next = new_node
```

```
    new_node.prev = curr
```

- Hubungkan node baru ke node terakhir: update next dan prev.

### 3. Metode display()

```
def display(self):
```

```
    curr = self.head
```

```
    while curr:
```

```
        print(curr.data, end=" <-> ")
```

```
        curr = curr.next
```

```
    print("None")
```

- Menampilkan seluruh isi linked list.
- Menggunakan curr untuk menelusuri list dari depan ke belakang.

- Format tampilan: data1 <-> data2 <-> ... <-> None.

#### 4. Metode delete\_awal()

```
def delete_awal(self):
```

```
    if not self.head:
```

```
        print("Linked list kosong!")
```

```
        return
```

- Cek apakah list kosong.

```
    if not self.head.next:
```

```
        self.head = None
```

- Jika hanya ada satu node, hapus dengan meng-set head = None.

```
    else:
```

```
        self.head = self.head.next
```

```
        self.head.prev = None
```

- Jika ada lebih dari satu node, head digeser ke node berikutnya dan prev-nya dihapus.

#### 5. Metode delete\_akhir()

```
def delete_akhir(self):
```

```
    if not self.head:
```

```
        print("Linked list kosong!")
```

```
        return
```

- Cek list kosong.

```
    curr = self.head
```

```
    if not curr.next:
```

```
        self.head = None
```

```
        return
```

- Jika hanya ada satu node, hapus.

```
    while curr.next:
```

```
        curr = curr.next
```

- Telusuri ke node terakhir.

```
    curr.prev.next = None
```

- Putuskan hubungan node terakhir dari sebelumnya.

## 6. Metode `delete_berdasarkan_nilai(target)`

```
def delete_berdasarkan_nilai(self, target):
```

```
    if not self.head:
```

```
        print("Linked list kosong!")
```

```
        return
```

- Cek list kosong.

```
    curr = self.head
```

```
    if curr.data == target:
```

```
        self.delete_awal()
```

```
        return
```

- Jika data yang akan dihapus ada di node pertama, panggil `delete_awal`.

```
    while curr:
```

```
        if curr.data == target:
```

- Loop cari node yang datanya sama dengan target.

```
            if curr.next:
```

```
                curr.prev.next = curr.next
```

```
                curr.next.prev = curr.prev
```

- Jika node berada di tengah, sambungkan node sebelumnya dengan yang sesudahnya.

```
            else:
```

```
                curr.prev.next = None
```

- Jika node berada di akhir, putuskan dari sebelumnya.

```
            return
```

```
    curr = curr.next
```

```
    print(f"Data {target} tidak ditemukan dalam linked list.")
```

- Jika tidak ditemukan, tampilkan pesan.

## 7. Penggunaan Program

```
dll = DoubleLinkedList()
```

```
dll.append(10)
```

```
dll.append(20)
```

```
dll.append(30)
```

```
dll.append(40)
```

- Membuat objek dll dan menambahkan empat data ke linked list.

```
print("Linked list awal:")
```

```
dll.display()
```

- Tampilkan isi awal list.

```
print("\nHapus node awal:")
```

```
dll.delete_awal()
```

```
dll.display()
```

- Hapus node pertama (10), lalu tampilkan list.

```
print("\nHapus node akhir:")
```

```
dll.delete_akhir()
```

```
dll.display()
```

- Hapus node terakhir (40), lalu tampilkan list.

```
print("\nHapus node dengan nilai 20:")
```

```
dll.delete_berdasarkan_nilai(20)
```

```
dll.display()
```

- Hapus node dengan nilai 20 (di tengah).

```
print("\nCoba hapus data yang tidak ada (50):")
```

```
dll.delete_berdasarkan_nilai(50)
```

- Mencoba menghapus data yang tidak ada (menampilkan pesan error).