

```

1  # Kelas Node untuk menyimpan data dan pointer ke node sebelumnya dan berikutnya
2  class Node:
3      def __init__(self, data):
4          self.data = data
5          self.prev = None
6          self.next = None
7
8  # Kelas DoubleLinkedList untuk operasi dasar linked list
9  class DoubleLinkedList:
10     def __init__(self):
11         self.head = None
12
13     # Tambah node di akhir
14     def tambah_node(self, data):
15         new_node = Node(data)
16         if self.head is None:
17             self.head = new_node
18         else:
19             current = self.head
20             while current.next:
21                 current = current.next
22             current.next = new_node
23             new_node.prev = current
24
25     # Tampilkan isi linked list
26     def tampil(self):
27         current = self.head
28         if not current:
29             print("Linked list kosong.")
30             return
31         print("Isi Linked List:")
32         while current:
33             print(current.data, end=" <-> " if current.next else "\n")
34             current = current.next
35
36     # Hapus node pertama
37     def hapus_awal(self):
38         if self.head is None:
39             print("Linked list kosong, tidak ada yang bisa dihapus.")
40             return
41         print(f"Node dengan data '{self.head.data}' dihapus dari awal.")
42         if self.head.next is None:
43             self.head = None
44         else:
45             self.head = self.head.next
46             self.head.prev = None
47
48     # Hapus node terakhir
49     def hapus_akhir(self):
50         if self.head is None:
51             print("Linked list kosong, tidak ada yang bisa dihapus.")
52             return
53         current = self.head
54         if current.next is None:
55             print(f"Node dengan data '{current.data}' dihapus dari akhir.")
56             self.head = None
57         else:
58             while current.next:
59                 current = current.next
60             print(f"Node dengan data '{current.data}' dihapus dari akhir.")
61             current.prev.next = None
62
63     # Hapus node berdasarkan nilai data
64     def hapus_berdasarkan_nilai(self, nilai):
65         if self.head is None:
66             print("Linked list kosong.")
67             return
68         current = self.head
69         while current:
70             if current.data == nilai:
71                 print(f"Node dengan nilai '{nilai}' berhasil dihapus.")
72                 if current.prev:
73                     current.prev.next = current.next
74                 else:
75                     self.head = current.next
76                 if current.next:
77                     current.next.prev = current.prev
78                 return
79             current = current.next
80         print(f"Node dengan nilai '{nilai}' tidak ditemukan.")
81
82 # Program utama
83 dll = DoubleLinkedList()
84 dll.tambah_node(10)
85 dll.tambah_node(20)
86 dll.tambah_node(30)
87 dll.tambah_node(40)
88
89 dll.tampil()           # 10 <-> 20 <-> 30 <-> 40
90
91 dll.hapus_awal()       # Hapus node 10
92 dll.tampil()           # 20 <-> 30 <-> 40
93
94 dll.hapus_akhir()      # Hapus node 40
95 dll.tampil()           # 20 <-> 30
96
97 dll.hapus_berdasarkan_nilai(20) # Hapus node 20
98 dll.tampil()           # 30
99
100 dll.hapus_berdasarkan_nilai(99) # Nilai tidak ditemukan
101 dll.tampil()           # 30

```

Penjelasan Tugas Modul 2.md

Kelas Node untuk menyimpan data dan pointer ke node sebelumnya dan berikutnya

```
class Node:
```

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

```
        self.data = data      # Menyimpan data pada node
```

```
        self.prev = None      # Menunjuk ke node sebelumnya (double linked list)
```

```
        self.next = None      # Menunjuk ke node berikutnya
```

Kelas DoubleLinkedList untuk operasi dasar linked list

```
class DoubleLinkedList:
```

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

```
        self.head = None      # Inisialisasi head (kepala) linked list
```

Tambah node di akhir

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

```
        new_node = Node(data) # Buat node baru dengan data
```

```
        if self.head is None:
```

```
            self.head = new_node # Jika list kosong, node ini jadi head
```

```
        else:
```

```
            current = self.head
```

```
            while current.next: # Loop ke node terakhir
```

```
                current = current.next
```

```
            current.next = new_node # Sambungkan node baru di akhir
```

```
            new_node.prev = current # Hubungkan balik ke node sebelumnya
```

Tampilkan isi linked list

```
    def tampil(self):
```

```
        current = self.head
```

```
        if not current:
```

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

```
            return
```

```
        print("Isi Linked List:")
```

```
        while current:
```

```

        print(current.data, end=" <-> " if current.next else "\n")
        current = current.next
# Hapus node pertama
def hapus_awal(self):
    if self.head is None:
        print("Linked list kosong, tidak ada yang bisa dihapus.")
        return
    print(f"Node dengan data '{self.head.data}' dihapus dari awal.")
    if self.head.next is None:
        self.head = None
    else:
        self.head = self.head.next
        self.head.prev = None
# Hapus node terakhir
def hapus_akhir(self):
    if self.head is None:
        print("Linked list kosong, tidak ada yang bisa dihapus.")
        return
    current = self.head
    if current.next is None:
        print(f"Node dengan data '{current.data}' dihapus dari akhir.")
        self.head = None
    else:
        while current.next:
            current = current.next
        print(f"Node dengan data '{current.data}' dihapus dari akhir.")
        current.prev.next = None
# Hapus node berdasarkan nilai data
def hapus_berdasarkan_nilai(self, nilai):
    if self.head is None:
        print("Linked list kosong.")

```

```

        return
    current = self.head
    while current:
        if current.data == nilai:
            print(f"Node dengan nilai '{nilai}' berhasil dihapus.")
            if current.prev:
                current.prev.next = current.next
            else:
                self.head = current.next
            if current.next:
                current.next.prev = current.prev
            return
        current = current.next
    print(f"Node dengan nilai '{nilai}' tidak ditemukan.")

# Program utama
dll = DoubleLinkedList()
dll.tambah_node(10)
dll.tambah_node(20)
dll.tambah_node(30)
dll.tambah_node(40)
dll.tampil()          # Tampilkan semua node
dll.hapus_awal()      # Hapus node pertama (10)
dll.tampil()          # Tampilkan hasil
dll.hapus_akhir()      # Hapus node terakhir (40)
dll.tampil()          # Tampilkan hasil
dll.hapus_berdasarkan_nilai(20) # Hapus node dengan data 20
dll.tampil()          # Tampilkan hasil
dll.hapus_berdasarkan_nilai(99) # Data 99 tidak ada
dll.tampil()          # Tampilkan hasil akhir

```