TUGAS MODUL 2 STRUKTUR DATA

MATA KULIAH
STRUKTUR DATA
Ditulis oleh: Ezi andrean (24241027)

SOAL :

Buatlah sebuah program yang berisi fungsi untuk menghapus double linked-list dengan ketentuan sebagai berikut :

1. **Menghapus node awal dari double linked-list**
2. **Menghapus node akhir dari double linked-list**
3. **Menghapus node double linked-list berdasarkan pada nilai data nya**

HASIL JAWABAN :

OUTPUT JAWABAN SOAL 1

```
Sebelum penghapusan:
10 <-> 20 <-> 30 <-> None
Menghapus node dengan data: 10
Setelah penghapusan:
20 <-> 30 <-> None
```

OUTPUT JAWABAN SOAL 2

```
Sebelum penghapusan node akhir:
10 <-> 20 <-> 30 <-> None
Menghapus node dengan data: 30
Setelah penghapusan node akhir:
10 <-> 20 <-> None
```

OUTPUT JAWABAN SOAL 3

```
Sebelum penghapusan:
10 <-> 20 <-> 30 <-> 40 <-> None
Menghapus node dengan data: 30
Setelah penghapusan node dengan nilai 30:
10 <-> 20 <-> 40 <-> None
```

PENJELASAN JAWABAN SOAL 1

Baris 1 :

```python
class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
        self.next = None
```

Penjelasan :

- **class Node: Membuat kelas Node untuk menyimpan data dan dua pointer (prev & next).**
- **self.data: Menyimpan nilai/data dalam node.**
- **self.prev: Menunjuk ke node sebelumnya (karena ini double linked list).**
- **self.next: Menunjuk ke node selanjutnya.**

Baris 2 :

```python
class DoubleLinkedList:
    def __init__(self):
        self.head = None
```

Penjelasan:

- **class DoubleLinkedList: Membuat kelas untuk mengelola seluruh double linked list.**
- **self.head = None: Awalnya, list masih kosong (tidak ada node).**

Baris 3 :

```python
def tambah_di_akhir(self, data):
    new_node = Node(data)
```

Penjelasan:

- **Fungsi untuk menambahkan node baru di akhir list.**
- **new_node = Node(data): Membuat node baru dengan data yang diberikan.**

Baris 4 :

```
if self.head is None:
    self.head = new_node
    return
```

Penjelasan:

- **Jika list masih kosong (head masih None), maka node baru langsung jadi head.**
- **Setelah itu return untuk keluar dari fungsi.**

Baris 5 :

```
curr = self.head
while curr.next:
    curr = curr.next
curr.next = new_node
new_node.prev = curr
```

Penjelasan:

- **Kalau list sudah ada isinya, cari node terakhir (curr).**
- **Setelah ketemu node terakhir, sambungkan new_node ke belakangnya.**
- **Atur juga pointer prev dari node baru supaya menunjuk ke node sebelumnya.**

Baris 6 :

```
def hapus_node_awal(self):
    if self.head is None:
        print("Linked list kosong. Tidak ada yang bisa dihapus.")
        return
```

Penjelasan:

- **Fungsi ini untuk menghapus node pertama.**
- **Jika list kosong, tampilkan pesan dan keluar dari fungsi.**

Baris 7 :

```
print(f"Menghapus node dengan data: {self.head.data}")
if self.head.next is None:
    self.head = None  # List jadi kosong
else:
    self.head = self.head.next
    self.head.prev = None
```

Penjelasan:

- **Menampilkan data dari node yang akan dihapus.**
- **Jika hanya ada 1 node, setelah dihapus, list jadi kosong (head = None).**
- **Jika ada lebih dari 1 node:**
- **Geser head ke node berikutnya.**
- **Set prev dari node baru ini ke None karena dia jadi head sekarang.**

Baris 8 :

```
def tampilkan(self):
    curr = self.head
    while curr:
        print(curr.data, end=" <-> ")
        curr = curr.next
    print("None")
```

Penjelasan:

- **Menampilkan seluruh isi linked list dari depan ke belakang.**
- **Setiap node dicetak, diikuti oleh panah "<->", lalu lanjut ke node berikutnya.**

Baris 9 :

```
# Contoh penggunaan
dll = DoubleLinkedList()
dll.tambah_di_akhir(10)
dll.tambah_di_akhir(20)
dll.tambah_di_akhir(30)
```

Penjelasan:

- **Membuat objek linked list (dll).**
- **Menambahkan 3 node dengan nilai 10, 20, dan 30.**

Baris 10 :

```
print("Sebelum penghapusan:")
dll.tampilkan()

dll.hapus_node_awal()

print("Setelah penghapusan:")
dll.tampilkan()
```

Penjelasan:

- **Menampilkan isi list sebelum penghapusan.**
- **Memanggil fungsi hapus_node_awal() untuk menghapus node pertama.**
- **Menampilkan isi list setelah penghapusan.**

PENJELASAN JAWBAN SOAL 2

Baris 1 :

```python
class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
        self.next = None
```

Penjelasan:

- **class Node: Membuat struktur dasar dari node.**
- **self.data: Menyimpan nilai/data dalam node.**
- **self.prev: Menyimpan referensi ke node sebelumnya.**
- **self.next: Menyimpan referensi ke node sesudahnya.**

Baris 2 :

```python
class DoubleLinkedList:
    def __init__(self):
        self.head = None
```

Penjelasan:

- **class DoubleLinkedList: Membuat class untuk list-nya.**
- **self.head = None: Awalnya list masih kosong (belum ada node).**

Baris 3 :

```python
def tambah_di_akhir(self, data):
    new_node = Node(data)
```

Penjelasan:

- **Membuat method untuk menambahkan node baru di akhir list.**
- **new_node = Node(data): Buat node baru berisi data.**

Baris 4 :

```python
if self.head is None:
    self.head = new_node
    return
```

Penjelasan:

- Kalau list masih kosong (head = None), maka node baru langsung jadi kepala list.
- Fungsi berhenti setelah itu.

Baris 5 :

```python
curr = self.head
while curr.next:
    curr = curr.next
```

Penjelasan:

- **Mulai dari head, terus maju ke node berikutnya hingga mencapai node terakhir (curr.next == None).**

Baris 6 :

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

Penjelasan:

- **Sambungkan new_node setelah curr.**
- **Hubungkan balik new_node ke curr melalui pointer prev.**

Baris 7 :

```
def hapus_node_akhir(self):
    if self.head is None:
        print("Linked list kosong. Tidak ada node yang bisa dihapus.")
        return
```

Penjelasan:

- **Cek apakah list kosong.**
- **Jika kosong, tampilkan pesan dan keluar dari fungsi.**

Baris 8 :

```
if self.head.next is None:
    # Hanya ada satu node
    print(f"Menghapus node dengan data: {self.head.data}")
    self.head = None
    return
```

Penjelasan:

- **Jika hanya ada satu node di list:**
- **Cetak pesan bahwa node akan dihapus.**
- **Setelah itu, set head jadi None karena list akan kosong.**

Baris 9 :

```
curr = self.head
while curr.next:
    curr = curr.next
```

Penjelasan:

- **Jika list berisi lebih dari satu node:**
- **Cari node terakhir (yang next-nya None).**

Baris 10 :

```
print(f"Menghapus node dengan data: {curr.data}")
curr.prev.next = None
curr.prev = None
```

Penjelasan:

- **Cetak nilai node yang akan dihapus.**
- **Putuskan hubungan dari node sebelumnya ke node terakhir:**
- **curr.prev.next = None: node sebelumnya tidak menunjuk ke curr lagi.**
- **curr.prev = None: pointer balik node terakhir dihapus juga.**

Baris 11 :

```
def tampilkan(self):
    curr = self.head
    while curr:
        print(curr.data, end=" <-> ")
        curr = curr.next
    print("None")
```

Penjelasan:

- **Method untuk menampilkan isi list dari depan ke belakang.**
- **Tampilkan setiap data dan panah, lalu lanjut ke node berikutnya.**
- **Cetak None di akhir untuk menandai akhir list.**

Baris 12 :

```
# Contoh penggunaan
dll = DoubleLinkedList()
dll.tambah_di_akhir(10)
dll.tambah_di_akhir(20)
dll.tambah_di_akhir(30)
```

Penjelasan:

- **Membuat objek linked list (dll).**
- **Menambahkan 3 node berisi data: 10 → 20 → 30.**

Baris 13 :

```
print("Sebelum penghapusan node akhir:")
dll.tampilkan()

dll.hapus_node_akhir()

print("Setelah penghapusan node akhir:")
dll.tampilkan()
```

Penjelasan:

- **Tampilkan isi list sebelum dilakukan penghapusan.**
- **Memanggil fungsi untuk menghapus node terakhir (node dengan data 30).**
- **Tampilkan isi list setelah node terakhir dihapus.**

PENJELASAN JAWABAN SOAL 3

Baris 1 :

```
class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
        self.next = None
```

Penjelasan:

- **Membuat class Node untuk menyimpan data dan pointer.**
- **data: Menyimpan nilai yang diberikan.**
- **prev: Menunjuk ke node sebelumnya.**
- **next: Menunjuk ke node setelahnya.**

Baris 2 :

```
class DoubleLinkedList:
    def __init__(self):
        self.head = None
```

Penjelasan:

- **Membuat class DoubleLinkedList untuk mengelola linked list.**
- **self.head: Awalnya list kosong (tidak ada node).**

Baris 3 :

```
def tambah_di_akhir(self, data):
    new_node = Node(data)
```

Penjelasan:

- **Membuat method untuk menambah node di akhir.**
- **Buat objek Node baru dengan nilai data.**

Baris 4 :

```
if self.head is None:
    self.head = new_node
    return
```

Penjelasan:

- **Jika list masih kosong, node baru jadi head.**
- **Langsung keluar dari fungsi setelah menambahkan node pertama.**

Baris 5 :

```
curr = self.head
while curr.next:
    curr = curr.next
```

Penjelasan:

- **Kalau list sudah ada isinya, cari node terakhir.**
- **Lanjut ke node berikutnya sampai curr.next == None.**

Baris 6 :

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

Penjelasan:

- **Sambungkan node terakhir dengan new_node.**
- **Atur pointer balik (prev) dari new_node ke node sebelumnya.**

Baris 7 :

```
def hapus_berdasarkan_nilai(self, nilai):
    if self.head is None:
        print("Linked list kosong. Tidak ada node yang bisa dihapus.")
        return
```

Penjelasan:

- **Method untuk menghapus node berdasarkan nilai datanya.**
- **Jika list kosong, tampilkan pesan dan keluar dari fungsi.**

Baris 8 :

```
curr = self.head

# Jika node yang ingin dihapus adalah head
if curr.data == nilai:
    print(f"Menghapus node dengan data: {nilai}")
    self.head = curr.next
    if self.head:
        self.head.prev = None
    return
```

Penjelasan:

- **Mulai dari node pertama (head) untuk mencari node yang ingin dihapus.**
- **Cek apakah node yang ingin dihapus adalah head.**
- **Jika ya geser head ke node berikutnya.**
- **Jika node baru masih ada, set prev-nya ke None.**
- **Setelah itu keluar dari fungsi.**

Baris 9 :

```
# Mencari node dengan nilai yang dicari
while curr and curr.data != nilai:
    curr = curr.next
```

Penjelasan:

- **Jika node pertama bukan yang dicari, telusuri node satu per satu sampai menemukan data yang cocok atau sampai akhir list.**

Baris 10 :

```
if curr is None:
    print(f"Node dengan data {nilai} tidak ditemukan.")
    return
```

Penjelasan:

- **Jika setelah ditelusuri tidak ada node dengan data yang dimaksud, tampilkan pesan bahwa node tidak ditemukan.**

Baris 11 :

```
print(f"Menghapus node dengan data: {nilai}")
# Menyesuaikan pointer prev dan next
if curr.next:
    curr.next.prev = curr.prev
if curr.prev:
    curr.prev.next = curr.next
```

Penjelasan:

- **Tampilkan pesan bahwa node dengan data yang cocok akan dihapus**
- **Update pointer next dari node sebelumnya dan prev dari node berikutnya agar melewati node yang akan dihapus.**
- **Ini memutus hubungan node yang dihapus dari list.**

Baris 12 :

```
def tampilkan(self):
    curr = self.head
    while curr:
        print(curr.data, end=" <-> ")
        curr = curr.next
    print("None")
```

Penjelasan:

- **Menampilkan isi linked list dari depan ke belakang.**
- **Menelusuri setiap node dan mencetak isinya.**

Baris 13 :

```
# Contoh penggunaan
dll = DoubleLinkedList()
dll.tambah_di_akhir(10)
dll.tambah_di_akhir(20)
dll.tambah_di_akhir(30)
dll.tambah_di_akhir(40)
```

Penjelasan:

- **Membuat list dan menambahkan empat node: 10 → 20 → 30 → 40.**

Baris 14 :

```
print("Sebelum penghapusan:")
dll.tampilkan()

dll.hapus_berdasarkan_nilai(30)

print("Setelah penghapusan node dengan nilai 30:")
dll.tampilkan()
```

Penjelasan:

- **Menampilkan isi list sebelum penghapusan node.**
- **Menghapus node yang memiliki data 30 dari linked list.**
- **Menampilkan isi list setelah node dengan data 30 dihapus.**
