

Nama Kelompok : Agung Ismaun (2405037)
: Faldy Ardiansyah (2405033)
: Ahmad Sobirin (2405005)
Kelas : RPL 1D
Mata Kuliah : Struktur Data
Kelompok : 4

Latihan 1 Single Linked List

1. Code Single Linked list

```
1 class Node:
2     def __init__(self, data):
3         self.data = data
4         self.next = None
5
6 class SingleLinkedList:
7     def __init__(self):
8         self.head = None
9
10    def insert_at_beginning(self, data):
11        new_node = Node(data)
12        new_node.next = self.head
13        self.head = new_node
14
15    def insert_after(self, pos, data):
16        if self.head is None:
17            print("List kosong")
18            return
19
20        current = self.head
21        while current is not None:
22            if current.data == pos:
23                break
24            current = current.next
25
26        if current is None:
27            print(f"Data {pos} tidak ditemukan")
28        else:
29            new_node = Node(data)
30            new_node.next = current.next
31            current.next = new_node
32
33    def delete(self, data):
34        if self.head is None:
35            print("List kosong")
36            return
37
38        if self.head.data == data:
39            self.head = self.head.next
40            return
41
42        current = self.head
43        while current.next is not None:
44            if current.next.data == data:
45                break
46            current = current.next
47
48        if current.next is None:
49            print(f"Data {data} tidak ditemukan")
50        else:
51            current.next = current.next.next
52
53    def display(self):
54        current = self.head
55        while current is not None:
56            print(current.data, end=" -> ")
57            current = current.next
58        print("None")
59
60 sll = SingleLinkedList()
61 sll.insert_at_beginning(10)
62 sll.insert_at_beginning(20)
63 sll.insert_after(20, 15)
64 sll.insert_after(10, 5)
65 sll.display()
66
67 sll.delete(15)
68 sll.delete(5)
69 sll.display()
```

2. Hasil Running dari single Linked List

```
NXHVV@LAPTOP-A48L7N02 MINGW64 /e/AKULIAH/SEMESTER 2/Struktur Data/struktrdata (master)
$ python -u "e:\AKULIAH\SEMESTER 2\Struktur Data\struktrdata\singlelink.py"
20 -> 15 -> 10 -> 5 -> None
20 -> 10 -> None
```

3. Penjelasan Kode

a. Kelas Node

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

- Kelas ini mewakili satu simpul (node) dalam linked list.
- data menyimpan nilai/data node.
- next adalah pointer ke node berikutnya (default-nya None).

b. Kelas SingleLinkedList

Kelas ini berisi berbagai metode manipulasi pada linked list.

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

- Inisialisasi linked list. head adalah simpul awal, dimulai dengan None.

c. insert_at_beginning(data)

```
def insert_at_beginning(self, data):
    new_node = Node(data)
    new_node.next = self.head
    self.head = new_node
```

- Menambahkan node di awal linked list.
- Node baru akan menjadi head yang baru.

d. Insert_after(pos, data)

```
def insert_after(self, pos, data):
    if self.head is None:
        print("List kosong")
        return
```

- Menambahkan node **setelah node tertentu** (berdasarkan nilai pos).
- Jika nilai pos ditemukan, node baru disisipkan setelahnya.

e. `delete(data)`

```
def delete(self, data):
    if self.head is None:
        print("List kosong")
        return
```

- Menghapus node dengan nilai tertentu.
- Jika nilai ada di head, maka head langsung diganti.
- Jika nilai ada di tengah atau akhir, sambungan node diubah agar node tersebut dilewati.

f. `display()`

```
def display(self):
    current = self.head
    while current is not None:
        print(current.data, end=" -> ")
        current = current.next
    print("None")
```

- Menampilkan isi linked list dari **head sampai akhir**.
- Menyambungkan tiap data dengan `->` sampai None.

g.

```
sll = SingleLinkedList()
sll.insert_at_beginning(10)
sll.insert_at_beginning(20)
```

- Tambah 10 → [10]
- Tambah 20 di depan → [20 -> 10]

h. `sll.insert_after(20, 15)`

Tambah 15 setelah node bernilai 20 → [20 -> 15 -> 10]

i. `sll.insert_after(10, 5)`

Tambah 5 setelah node bernilai 10 → [20 -> 15 -> 10 -> 5]

```
j. sll.display()
# Output: 20 -> 15 -> 10 -> 5 -> None

sll.delete(15)
sll.delete(5)
sll.display()
# Output: 20 -> 10 -> None
```

Node 15 dan 5 dihapus → tersisa [20 -> 10]

Latihan 2 Double Link List

1. Code Double Link List

```
1 class DNode:
2     def __init__(self, data):
3         self.data = data
4         self.prev = None
5         self.next = None
6
7 class DoubleLinkedList:
8     def __init__(self):
9         self.head = None
10
11     def insert_before(self, target_data, data):
12         current = self.head
13         if current is None:
14             print("List kosong.")
15             return
16         if current.data == target_data:
17             new_node = DNode(data)
18             new_node.next = self.head
19             self.head.prev = new_node
20             self.head = new_node
21             return
22         while current and current.data != target_data:
23             current = current.next
24         if current is None:
25             print("Data target tidak ditemukan.")
26             return
27         new_node = DNode(data)
28         new_node.prev = current.prev
29         new_node.next = current
30         if current.prev:
31             current.prev.next = new_node
32         current.prev = new_node
33
34     def delete(self, data):
35         current = self.head
36         while current and current.data != data:
37             current = current.next
38         if current is None:
39             print("Data tidak ditemukan.")
40             return
41         if current.prev:
42             current.prev.next = current.next
43         if current.next:
44             current.next.prev = current.prev
45         if current == self.head:
46             self.head = current.next
47
48     def display(self):
49         current = self.head
50         while current:
51             print(current.data, end=" -> ")
52             current = current.next
53         print("None")
54
55 # Contoh penggunaan
56 dll = DoubleLinkedList()
57 dll.head = DNode(10)
58 dll.insert_before(10, 5)
59 dll.insert_before(10, 7)
60 dll.display()
61 dll.delete(7)
62 dll.display()
63
```

2. Hasil Running

```
PS D:\Materi dan Tugas Semester 2\Tugas Struktur Data\tugas_struktur_data> python double_linklist.py
5 -> 7 -> 10 -> None
5 -> 10 -> None
```

3. Penjelasan Code

a. Class Node

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

Setiap DNode menyimpan:

- data (isi node)
- prev (penunjuk ke node sebelumnya)
- next (penunjuk ke node selanjutnya)

b. Class Double Linked List

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

Membuat linked list kosong. self.head adalah node pertama (bisa None jika kosong).

c. Method insert_before()

```
def insert_before(self, target_data, data):
```

Tujuan: **menambahkan node data sebelum node yang bernilai target_data**

Prosesnya:

Jika list kosong, operasi dibatalkan:

```
current = self.head
if current is None:
    print("List kosong.")
    return
```

Jika target ada di kepala list, maka node baru dimasukkan paling awal:

```

if current.data == target_data:
    new_node = DNode(data)
    new_node.next = self.head
    self.head.prev = new_node
    self.head = new_node
    return

```

Mencari node yang datanya = target_data. Jika tidak ada, berhenti:

```

while current and current.data != target_data:
    current = current.next
if current is None:
    print("Data target tidak ditemukan.")
    return

```

Menyesuaikan semua koneksi:

- prev.next dari node sebelumnya sekarang mengarah ke node baru
- current.prev dari target sekarang adalah node baru

```

new_node = DNode(data)
new_node.prev = current.prev
new_node.next = current
if current.prev:
    current.prev.next = new_node
current.prev = new_node

```

d. Method delete()

Menghapus node yang berisi data:

```

def delete(self, data):

```

Mencari node yang mengandung data tersebut:

```

current = self.head
while current and current.data != data:
    current = current.next

```

Jika tidak ketemu, operasi dibatalkan:

```

if current is None:
    print("Data tidak ditemukan.")
    return

```

- Jika node yang dihapus bukan di kepala, sesuaikan pointer node sebelumnya dan sesudahnya
- Jika node yang dihapus adalah kepala (head), ubah head ke node berikutnya

```

if current.prev:
    current.prev.next = current.next
if current.next:
    current.next.prev = current.prev
if current == self.head:
    self.head = current.next

```

e. Method display()

```

def display(self):
    current = self.head
    while current:
        print(current.data, end=" -> ")
        current = current.next
    print("None")

```

Menampilkan isi linked list dari depan ke belakang, formatnya:

```

PS D:\Materi dan Tugas Semester 2\Tugas Struktur Data\tugas_struktur_data> python double_linklist.py
5 -> 7 -> 10 -> None
5 -> 10 -> None

```

f. Contoh Penggunaan

```

# Contoh penggunaan
dll = DoubleLinkedList()
dll.head = DNode(10)
dll.insert_before(10, 5)
dll.insert_before(10, 7)
dll.display()
dll.delete(7)
dll.display()

```

Latihan 3 Circular Linked List

1. Code Circular Linked List

```
1 class CNode:
2     def __init__(self, data):
3         self.data = data
4         self.next = None
5
6 class CircularLinkedList:
7     def __init__(self):
8         self.head = None
9
10    def insert_at_beginning(self, data):
11        new_node = CNode(data)
12
13        if not self.head:
14            self.head = new_node
15            new_node.next = new_node
16
17        else:
18            current = self.head
19            while current.next != self.head:
20                current = current.next
21            new_node.next = self.head
22            current.next = new_node
23            self.head = new_node
24
25    def delete(self, data):
26        if not self.head:
27            return
28        current = self.head
29        prev = None
30        while True:
31            if current.data == data:
32                if prev:
33                    prev.next = current.next
34                else:
35                    if current.next == self.head:
36                        self.head = None
37                        return
38                    else:
39                        tail = self.head
40                        while tail.next != self.head:
41                            tail = tail.next
42                        self.head = current.next
43                        tail.next = self.head
44                return
31            prev = current
32            current = current.next
47            if current == self.head:
48                break
49
50    def display(self):
51        if not self.head:
52            print("List kosong.")
53            return
54        current = self.head
55        while True:
56            print(current.data, end=" -> ")
57            current = current.next
58            if current == self.head:
59                break
60        print("(kembali ke head)")
61
62 cll = CircularLinkedList()
63 cll.insert_at_beginning(30)
64 cll.insert_at_beginning(20)
65 cll.insert_at_beginning(10)
66 cll.display() # Output: 10 -> 20 -> 30 -> (kembali ke head)
67 cll.delete(20)
68 cll.display() # Output: 10 -> 30 -> (kembali ke head)
69
```


2. Hasil running

```
✓ class CNode: ...  
  
10 -> 20 -> 30 -> (kembali ke head)  
10 -> 30 -> (kembali ke head)
```

3. Penjelasan Code

a. Class node

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

- data: menyimpan isi dari node (bisa angka, string, dsb).
- next: menunjuk ke node berikutnya. Karena ini Circular Linked List, maka node terakhir akan menunjuk ke node pertama.

b. Circular Linked List

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

- head: titik awal dari list.
- Jika list kosong, head = None.

c. Metode insert at beginning

```
def insert_at_beginning(self, data):  
    new_node = CNode(data)
```

- Membuat node baru dengan data yang diberikan.

d. Metode if not

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

- Jika list kosong, jadikan new_node sebagai head, dan arahkan next-nya ke dirinya sendiri
→ membentuk lingkaran.

```

else:
    current = self.head
    while current.next != self.head:
        current = current.next
    new_node.next = self.head
    current.next = new_node
    self.head = new_node

```

- Menemukan node terakhir (yang menunjuk kembali ke head).
- Update pointer untuk mempertahankan struktur circular.
- Geser head ke node baru (karena kita menyisipkan di depan).
- e. Metode delete

```

def delete(self, data):
    if not self.head:
        return
    current = self.head
    prev = None
    while True:
        if current.data == data:

```

- Mulai dari head, dan siapkan prev untuk melacak node sebelumnya.
- Jika data cocok, kita akan hapus current.

```

        if prev:
            prev.next = current.next
        else:
            if current.next == self.head:
                self.head = None
            return

```

- Kalau hanya satu node di list → hapus dan kosongkan list.

```

        else:
            tail = self.head
            while tail.next != self.head:
                tail = tail.next
            self.head = current.next
            tail.next = self.head
    return

```

- Jika lebih dari satu node → diperbarui hingga node terakhir

```

prev = current
current = current.next
if current == self.head:
    break

```

- Kembali ke awal, artinya data tidak ditemukan
- f. Metode display

```

def display(self):
    if not self.head:
        print("List kosong.")
    return

```

- Tampilkan pesan jika list kosong.

```

current = self.head
while True:
    print(current.data, end=" -> ")
    current = current.next
    if current == self.head:
        break
print("(kembali ke head)")

```

- Mulai dari head, cetak semua data.
- Karena ini circular, kita berhenti saat current kembali ke head.
- g. Contoh penggunaan

```

c1l = CircularLinkedList()
c1l.insert_at_beginning(30)
c1l.insert_at_beginning(20)
c1l.insert_at_beginning(10)
c1l.display() # Output: 10 -> 20 -> 30 -> (kembali ke head)
c1l.delete(20)
c1l.display() # Output: 10 -> 30 -> (kembali ke head)

```

Tugas Linked List

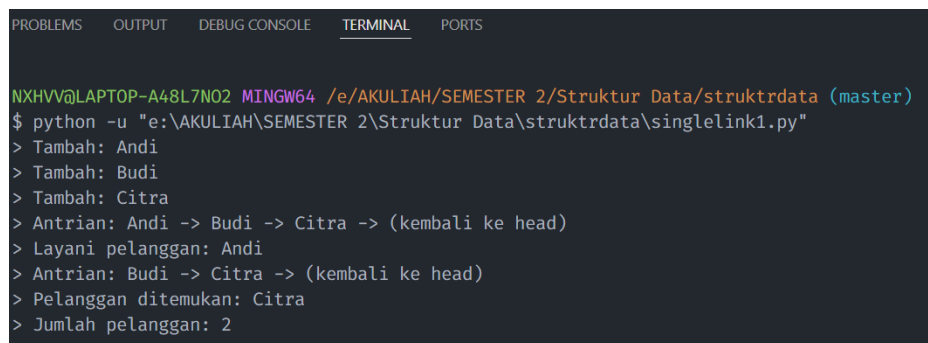
1. Code Single Linked List

```

1 class Node:
2     def __init__(self, nama):
3         self.nama = nama
4         self.next = None
5
6 class CircularQueue:
7     def __init__(self):
8         self.tail = None
9         self.size = 0
10
11     def tambah_pelanggan(self, nama):
12         new_node = Node(nama)
13         if not self.tail:
14             self.tail = new_node
15             self.tail.next = new_node
16         else:
17             new_node.next = self.tail.next
18             self.tail.next = new_node
19             self.tail = new_node
20         self.size += 1
21         print(f"> Tambah: {nama}")
22
23     def layani_pelanggan(self):
24         if not self.tail:
25             print("> Antrian kosong")
26             return
27         head = self.tail.next
28         print(f"> Layani pelanggan: {head.nama}")
29         if self.tail == head:
30             self.tail = None
31         else:
32             self.tail.next = head.next
33             self.size -= 1
34
35     def tampilkan_antrian(self):
36         if not self.tail:
37             print("> Antrian kosong")
38             return
39         hasil = []
40         current = self.tail.next
41         while True:
42             hasil.append(current.nama)
43             current = current.next
44             if current == self.tail.next:
45                 break
46         print("> Antrian: " + " -> ".join(hasil) + " -> (kembali ke head)")
47
48     def cari_pelanggan(self, nama):
49         if not self.tail:
50             print("> Antrian kosong")
51             return False
52         current = self.tail.next
53         while True:
54             if current.nama == nama:
55                 print(f"> Pelanggan ditemukan: {nama}")
56                 return True
57             current = current.next
58             if current == self.tail.next:
59                 break
60         print(f"> Pelanggan tidak ditemukan: {nama}")

```

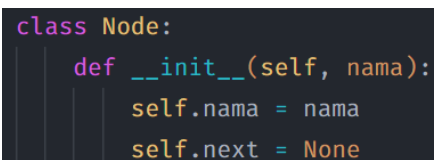
2. Hasil Running



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
NXHVVV@LAPTOP-A48L7N02 MINGW64 /e/AKULIAH/SEMESTER 2/Struktur Data/struktrdata (master)
$ python -u "e:\AKULIAH\SEMESTER 2\Struktur Data\struktrdata\singlelink1.py"
> Tambah: Andi
> Tambah: Budi
> Tambah: Citra
> Antrian: Andi -> Budi -> Citra -> (kembali ke head)
> Layani pelanggan: Andi
> Antrian: Budi -> Citra -> (kembali ke head)
> Pelanggan ditemukan: Citra
> Jumlah pelanggan: 2
```

3. Penjelasan Code

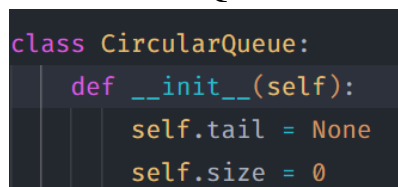
a. Kelas Node



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

- Setiap Node mewakili satu pelanggan.
- Atribut nama menyimpan nama pelanggan.
- Atribut next menunjuk ke node berikutnya (untuk membentuk circular linked list).

b. Kelas CircularQueue



```
class CircularQueue:
    def __init__(self):
        self.tail = None
        self.size = 0
```

- **tail** menunjuk ke node terakhir (yang menyambung ke head).
- **size** menyimpan jumlah node dalam antrian.

c. Menambah pelanggan

```
def tambah_pelanggan(self, nama):
    new_node = Node(nama)
    if not self.tail:
        self.tail = new_node
        self.tail.next = new_node
    else:
        new_node.next = self.tail.next
        self.tail.next = new_node
        self.tail = new_node
    self.size += 1
    print(f"> Tambah: {nama}")
```

- Jika antrian kosong: node pertama menunjuk ke dirinya sendiri.
- Jika tidak kosong: node baru ditambahkan setelah tail, dan menjadi tail baru.
- Karena circular, tail.next selalu menunjuk ke head.

d. Melayani Pelanggan (hapus dari depan/head)

```
def layani_pelanggan(self):
    if not self.tail:
        print("> Antrian kosong")
        return
    head = self.tail.next
    print(f"> Layani pelanggan: {head.nama}")
    if self.tail == head:
        self.tail = None
    else:
        self.tail.next = head.next
    self.size -= 1
```

- head adalah node di depan (yaitu tail.next).
- Jika hanya 1 node (tail == head), maka setelah dilayani, antrian menjadi kosong.
- Jika lebih dari 1 node, hapus head dengan mengubah tail.next.

e. Menampilkan seluruh antrian

```
def tampilkan_antrian(self):
    if not self.tail:
        print("> Antrian kosong")
        return
    hasil = []
    current = self.tail.next
    while True:
        hasil.append(current.nama)
        current = current.next
        if current == self.tail.next:
            break
    print("> Antrian: " + " -> ".join(hasil) + " -> (kembali ke head)")
```

- Mulai dari head (tail.next), iterasi hingga kembali ke head.
- Disimpan di list hasil untuk ditampilkan dengan panah (->).

f. Mencari pelanggan

```
def cari_pelanggan(self, nama):
    if not self.tail:
        print("> Antrian kosong")
        return False
    current = self.tail.next
    while True:
        if current.nama == nama:
            print(f"> Pelanggan ditemukan: {nama}")
```

- Mencari nama mulai dari head.
- Berhenti jika ditemukan atau sudah satu putaran penuh.

g. Menampilkan jumlah pelanggan

```
def jumlah_pelanggan(self):
    print(f"> Jumlah pelanggan: {self.size}")
```

Cukup menampilkan nilai dari self.size

h. Contoh Penggunaan

```
antrian = CircularQueue()
antrian.tambah_pelanggan("Andi")
antrian.tambah_pelanggan("Budi")
antrian.tambah_pelanggan("Citra")
antrian.tampilkan_antrian()
antrian.layani_pelanggan()
antrian.tampilkan_antrian()
antrian.cari_pelanggan("Citra")
antrian.jumlah_pelanggan()
```

ini menunjukkan bagaimana fitur-fitur tadi digunakan