HASIL IMPLEMENTASI SINGLE LINKED LIST DAN DOUBLE LINKED LIST

Abdullah Azzam Rabbani

10240038

Matkul: Struktur Data

1. Pendahuluan

Linked List adalah struktur data dinamis yang terdiri dari node-node yang saling terhubung. Laporan ini membandingkan implementasi **Single Linked List** (SLL) dan **Double Linked List** (DLL) melalui hasil running program yang mencakup operasi dasar seperti penambahan, pencarian, dan penghapusan data.

2. Implementasikan Program

2.1. Single Linked List

- **Operasi yang Tersedia:**
  - **Tambah data di awal (add).**
  - **Cari data (search).**
  - **Hapus data (remove).**
  - **Tampilkan list (show).**

- **Kode Program**

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

    def getdata(self):
        return self.data

    def getnext(self):
        return self.next

    def setdata(self, newdata):
        self.data = newdata

    def setnext(self, newnext):
```

```
        self.next = newnext


class OrderedList:
    def __init__(self):
        self.head = None

    def show(self):
        current = self.head
        print("Head -> ", end="")
        while current != None:
            print(current.getdata(), end=" -> ")
            current = current.getnext()
        print("None")

    def isempty(self):
        return self.head == None

    def add(self, item):
        temp = Node(item)
        temp.setnext(self.head)
        self.head = temp

    def search(self):
        current = self.head
        s = int(input("\nMasukkan data yang akan dicari: "))
        found = False
        while current != None and not found:
            if current.getdata() == s:
                found = True
            current = current.getnext()
        print(">> Data ditemukan" if found else ">> Data tidak ditemukan")

    def remove(self):
        if self.head is None:
            print("\n>> Linked List kosong")
            return

        s = int(input("\nMasukkan data yang akan dihapus: "))
        current = self.head
        previous = None
        found = False

        while not found and current != None:
            if current.getdata() == s:
                found = True
            else:
                previous = current
                current = current.getnext()

        if found:
            if previous == None:
                self.head = current.getnext()
            else:
                previous.setnext(current.getnext())
            print(">> Data berhasil dihapus")
```

```
        else:
            print(">> Data tidak ditemukan")


# Fungsi untuk menjalankan operasi
def main():
    ol = OrderedList()

    while True:
        print("\n===== MENU =====")
        print("1. Tambah Data")
        print("2. Cari Data")
        print("3. Hapus Data")
        print("4. Tampilkan List")
        print("5. Keluar")

        pilihan = input("Pilih menu (1-5): ")

        if pilihan == "1":
            data = int(input("Masukkan data yang akan ditambahkan: "))
            ol.add(data)
            print(f">> {data} berhasil ditambahkan")
        elif pilihan == "2":
            if ol.isempty():
                print("\n>> Linked List kosong")
            else:
                ol.search()
        elif pilihan == "3":
            ol.remove()
        elif pilihan == "4":
            if ol.isempty():
                print("\nHead -> None")
            else:
                ol.show()
        elif pilihan == "5":
            print("\nProgram selesai.")
            break
        else:
            print("\nPilihan tidak valid!")


if __name__ == "__main__":
    main()
```

- **Output Program**

```
===== MENU =====
1. Tambah Data
2. Cari Data
3. Hapus Data
4. Tampilkan List
5. Keluar
Pilih menu (1-5): 1
Masukkan data yang akan ditambahkan: 30
>> 30 berhasil ditambahkan

===== MENU =====
1. Tambah Data
2. Cari Data
3. Hapus Data
4. Tampilkan List
5. Keluar
Pilih menu (1-5): 4
Head -> 30 -> 30 -> 29 -> None
```

Gambar 1: Output Single Linked List

2.2. Double Linked List

- **Operasi yang Tersedia:**
  - **Tambah data di awal (prepend).**
  - **Tambah data di akhir (append).**
  - **Tampilkan list (print_list).**

- **Kode Program**

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

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

    def append(self, data):
        if self.head is None:
            new_node = Node(data)
```

```
            new_node.prev = None
            self.head = new_node
        else:
            new_node = Node(data)
            cur = self.head
            while cur.next:  # Perbaikan sintaks: tambahkan colon (:)
                cur = cur.next
            cur.next = new_node
            new_node.prev = cur
            new_node.next = None

    def prepend(self, data):
        if self.head is None:
            new_node = Node(data)  # Perbaikan: ganti 'node' -> 'Node'
            new_node.next = self.head
            self.head = new_node
        else:
            new_node = Node(data)
            self.head.prev = new_node
            new_node.next = self.head
            self.head = new_node
            new_node.prev = None

    def print_list(self):
        cur = self.head
        while cur:
            print(cur.data)
            cur = cur.next


def main():
    dll = DoubleLinkedList()

    while True:
        print("\n=== MENU DOUBLE LINKED LIST ===")
        print("1. Tambah Data di Awal")
        print("2. Tambah Data di Akhir")
        print("3. Tampilkan List")
        print("4. Keluar")

        pilihan = input("Pilih menu (1-4): ")

        if pilihan == "1":
            data = int(input("Masukkan data: "))
            dll.prepend(data)
            print(f"Data {data} ditambahkan di awal!")

        elif pilihan == "2":
            data = int(input("Masukkan data: "))
            dll.append(data)
            print(f"Data {data} ditambahkan di akhir!")

        elif pilihan == "3":
            print("\nIsi Linked List:")
            dll.print_list()
```

```
        elif pilihan == "4":
            print("\nProgram selesai.")
            break

        else:
            print("\nPilihan tidak valid!")


if __name__ == "__main__":
    main()
```

- **Output Program**

```
=== MENU DOUBLE LINKED LIST ===
1. Tambah Data di Awal
2. Tambah Data di Akhir
3. Tampilkan List
4. Keluar
Pilih menu (1-4): 2
Masukkan data: 90
Data 90 ditambahkan di akhir!

=== MENU DOUBLE LINKED LIST ===
1. Tambah Data di Awal
2. Tambah Data di Akhir
3. Tampilkan List
4. Keluar
Pilih menu (1-4): 1
Masukkan data: 19
Data 19 ditambahkan di awal!

=== MENU DOUBLE LINKED LIST ===
1. Tambah Data di Awal
2. Tambah Data di Akhir
3. Tampilkan List
4. Keluar
Pilih menu (1-4): 3

Isi Linked List:
19
30
90
```

Gambar 2: Output Double Linked List

3. Kesimpulan

1. Single Linked List lebih efisien dalam penggunaan memori tetapi terbatas pada operasi satu arah.

2. Double Linked List memungkinkan traversal dua arah dengan kompleksitas untuk operasi di awal/akhir, namun memerlukan memori lebih besar.
3. Pemilihan struktur data tergantung pada kebutuhan:
    - Gunakan SLL untuk operasi sederhana seperti stack.
    - Gunakan DLL untuk aplikasi yang memerlukan navigasi maju/mundur.