

Laporan Hasil Praktikum
Algoritma Dan Struktur Data
Jobsheet 13



Disusun Oleh :

Nama : Fadhil Taufiqurrachman
NIM : 244107020090
Kelas : Teknik Informatika 1E

Program Studi Teknik Informatika
Jurusan Teknologi Informasi
Politeknik Negeri Malang 2025

2.1 Percobaan 1 : Pembuatan Double Linked List

2.1.1 Kode Program

Kode program pada class Mahasiswa08 :

```
package Jobsheet13;

public class Mahasiswa08 {
    String nama, nim, kelas;
    double ipk;

    Mahasiswa08(String nama, String nim, String kelas, double ipk) {
        this.nama = nama;
        this.nim = nim;
        this.kelas = kelas;
        this.ipk = ipk;
    }

    public void tampil() {
        System.out.println("Nim : " + nim + ", Nama : " + nama + ",
        Kelas : " + kelas + ", IPK : " + ipk);
    }
}
```

Kode program pada class Node08 :

```
package Jobsheet13;

public class Node08 {
    Mahasiswa08 data;
    Node08 prev;
    Node08 next;

    public Node08(Mahasiswa08 data) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}
```

Kode program pada class DoubleLinkedList08 :

```
package Jobsheet13;

public class DoubleLinkedList08 {
    Node08 head;
    Node08 tail;

    public DoubleLinkedList08() {
        head = null;
        tail = null;
    }

    public boolean isEmpty() {
        return head == null;
    }

    public void addFirst(Mahasiswa08 data) {
        Node08 newNode = new Node08(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
    }

    public void addLast(Mahasiswa08 data) {
        Node08 newNode = new Node08(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }
}
```

```

    public void insertAfter(String keyNim, Mahasiswa08 data) {
        Node08 current = head;
        while(current != null && !current.data.nim.equals(keyNim)) {
            current = current.next;
        }

        if (current == null) {
            System.out.println("Node Dengan NIM " + keyNim + " Tidak Ditemukan.");
            return;
        }
        Node08 newNode = new Node08(data);

        if (current == tail) {
            current.next = newNode;
            newNode.prev = current;
            tail = newNode;
        } else {
            newNode.next = current.next;
            newNode.prev = current;
            current.next.prev = newNode;
            current.next = newNode;
        }
        System.out.println("Node Berhasil Disisipkan Setelah NIM " + keyNim);
    }

    public void print() {
        Node08 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }

    public Node08 search(String nim) {
        Node08 current = head;
        while (current != null) {
            if (current.data.nim.equals(nim)) {
                return current;
            }
            current = current.next;
        }
        return null;
    }
}

```

Kode program pada class DLLMain08 :

```
package Jobsheet13;

import java.util.Scanner;
public class DLLMain08 {
    static Mahasiswa08 inputMahasiswa(Scanner input) {
        System.out.print("Masukkan NIM : ");
        String nim = input.nextLine();
        System.out.print("Masukkan Nama : ");
        String nama = input.nextLine();
        System.out.print("Masukkan Kelas : ");
        String kelas = input.nextLine();
        System.out.print("Masukkan IPK : ");
        double ipk = input.nextDouble();
        input.nextLine();
        return new Mahasiswa08(nama, nim, kelas, ipk);
    }

    public static void main(String[] args) {
        DoubleLinkedList08 list = new DoubleLinkedList08();
        Scanner input = new Scanner(System.in);
        int pilihan;

        do {
            System.out.println("\nMenu Double Linked List Mahasiswa");
            System.out.println("1. Tambah Di Awal");
            System.out.println("2. Tambah Di Akhir");
            System.out.println("3. Hapus Di Awal");
            System.out.println("4. Hapus Di Akhir");
            System.out.println("5. Tampilkan Data");
            System.out.println("7. Cari Mahasiswa Berdasarkan NIM");
            System.out.println("0. Keluar");
            System.out.print("Pilih Menu : ");
            pilihan = input.nextInt();
            input.nextLine();

            switch (pilihan) {
                case 1 -> {
                    Mahasiswa08 mhs = inputMahasiswa(input);
                    list.addFirst(mhs);
                }
            }
        }
    }
}
```

```

        case 2 -> {
            Mahasiswa08 mhsw = inputMahasiswa(input);
            list.addLast(mhsw);
        }
        case 3 -> list.removeFirst();
        case 4 -> list.removeLast();
        case 5 -> list.print();
        case 7 -> {
            System.out.print("Masukkan NIM Yang Dicari : ");
            String nim = input.nextLine();
            Node08 found = list.search(nim);
            if (found != null) {
                System.out.println("Data Ditemukan :");
                found.data.tampil();
            } else {
                System.out.println("Data Tidak Ditemukan.");
            }
        }
        case 0 -> System.out.println("Keluar Dari Program.");
        default -> System.out.println("Pilihan Tidak Valid!");
    }
    while (pilihan != 0);
    input.close();
}
}

```

2.1.2 Verifikasi

```

Menu Double Linked List Mahasiswa
1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 1
Masukkan NIM : 20304050
Masukkan Nama : Hermione Granger
Masukkan Kelas : Gryffindor
Masukkan IPK : 4.0

Menu Double Linked List Mahasiswa
1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 5
Nim : 20304050, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0

```

3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 2
Masukkan NIM : 60708090
Masukkan Nama : Harry Potter
Masukkan Kelas : Gryffindor
Masukkan IPK : 3.9

Menu Double Linked List Mahasiswa

1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 1
Masukkan NIM : 11223344
Masukkan Nama : Ron Weasley
Masukkan Kelas : Gryffindor
Masukkan IPK : 3.8

Menu Double Linked List Mahasiswa

1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 5
Nim : 11223344, Nama : Ron Weasley, Kelas : Gryffindor, IPK : 3.8
Nim : 20304050, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0
Nim : 60708090, Nama : Harry Potter, Kelas : Gryffindor, IPK : 3.9

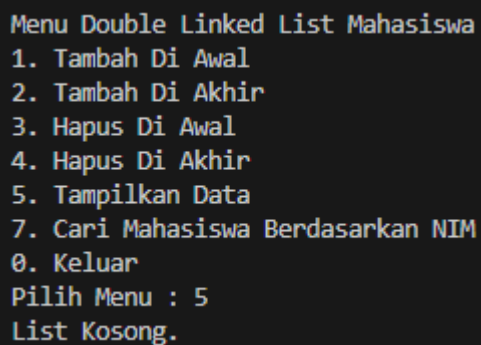
2.1.3 Pertanyaan

1. Berikut adalah perbedaan antara Single Linked List dan Double Linked List.
 - a. Perbedaan yang paling jelas dan mencolok adalah terletak pada struktur node dari linked list. Single linked list hanya memiliki 2 komponen pada setiap node, yakni data dan pointer next. Sedangkan pada Double linked list, setiap node memiliki 3 komponen yang terdiri dari data, pointer next dan pointer prev.
 - b. Karena perbedaan komponen tersebut, pada Single linked list arah traversal hanya bisa dilakukan secara satu arah yakni dari awal ke akhir (Head ke tail). Sedangkan pada Double linked list, selain dari head ke tail, arah traversal juga bisa dilakukan ke arah sebaliknya (Tail ke Head) karena memiliki pointer prev.
 - c. Single linked list lebih sederhana dalam implementasi, sedangkan Double linked list lebih kompleks. Namun meskipun lebih kompleks, Double linked lebih efisien dalam keperluan mengakses node dengan 3 komponen yang dimilikinya.
2. Atribut next dan prev tersebut merupakan komponen pointer untuk suatu node dalam Double linked list (Karena terdapat atribut prev). Yang dimana atribut next akan berisi alamat ke node selanjutnya, sedangkan atribut prev akan berisi alamat ke node sebelumnya.
3. Konstruktor tersebut berfungsi sebagai inisialisasi awal dari atribut head dan tail. Yang dimana jika belum terdapat data yang di inputkan (Belum ada node atau linked list masih kosong), maka head dan tail akan di atur ke null
4. Kode program tersebut adalah struktur dari penambahan node atau data untuk kondisi jika kita menginputkan node untuk pertama kalinya atau ketika linked list masih kosong. Jadi kondisi tersebut akan mengecek apakah linked list kosong, jika benar maka head dan tail akan di arahkan atau diubah ke node baru tersebut.
5. Saat newNode ditambahkan pada awal linked list, node saat ini yang awalnya adalah head akan berubah menjadi node kedua. Oleh karena itu, pointer prev node saat ini perlu menunjuk kembali ke newNode yang akan menjadi head baru.

6. Modifikasi kode program pada method `print()` di class `DoubleLinkedList08`.

```
public void print() {
    if (isEmpty()) {
        System.out.println("List Kosong.");
        return;
    }
    Node08 current = head;
    while (current != null) {
        current.data.tampil();
        current = current.next;
    }
}
```

Hasil Verifikasi :



```
Menu Double Linked List Mahasiswa
1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 5
List Kosong.
```

7. Statement `current.next.prev = newNode;` tersebut berfungsi untuk mengatur pointer `prev` dari node setelah `current` agar menunjuk ke `newNode`. Karena data atau node yang akan disisipkan berada pada setelah node `current`.

8. Menambahkan menu pilihan dan switch case pada class `DLLMain08`.

```
.....
System.out.println("6. Sisipkan Setelah NIM Tertentu");
.....
        case 6 -> {
            System.out.print("Masukkan NIM Untuk Menyisipkan
Setelah NIM : ");
            String nimCari = input.nextLine();
            Mahasiswa08 mhsa = inputMahasiswa(input);
            list.insertAfter(nimCari, mhsa);
        }
```

Hasil Verifikasi :

```
Pilih Menu : 5
Nim : 20304050, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0
Nim : 60708090, Nama : Harry Potter, Kelas : Gryffindor, IPK : 3.9

Menu Double Linked List Mahasiswa
1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 6
Masukkan NIM Untuk Menyisipkan Setelah NIM : 20304050
Masukkan NIM : 11223344
Masukkan Nama : Ron Weasley
Masukkan Kelas : Gryffindor
Masukkan IPK : 3.8
Node Berhasil Disisipkan Setelah NIM 20304050

Menu Double Linked List Mahasiswa
1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 5
Nim : 20304050, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0
Nim : 11223344, Nama : Ron Weasley, Kelas : Gryffindor, IPK : 3.8
Nim : 60708090, Nama : Harry Potter, Kelas : Gryffindor, IPK : 3.9
```

2.2 Percobaan 2 : Modifikasi Elemen Pada Double Linked List

2.2.1 Kode Program

Menambahkan kode program pada class DoubleLinkedList08 :

```
public void removeFirst() {
    if (isEmpty()) {
        System.out.println("List Kosong, Tidak Bisa Dihapus.");
        return;
    }

    if (head == tail) {
        head = tail = null;
    } else {
        head = head.next;
        head.prev = null;
    }
}

public void removeLast() {
    if (isEmpty()) {
        System.out.println("List Kosong, Tidak Bisa Dihapus.");
        return;
    }

    if (head == tail) {
        head = tail = null;
    } else {
        tail = tail.prev;
        tail.next = null;
    }
}
```

2.2.2 Verifikasi

```
Pilih Menu : 5
Nim : 11223344, Nama : Ron Weasley, Kelas : Gryffindor, IPK : 3.8
Nim : 20304050, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0
Nim : 60708090, Nama : Harry Potter, Kelas : Gryffindor, IPK : 3.9

Menu Double Linked List Mahasiswa
1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 3

Menu Double Linked List Mahasiswa
1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 4

Menu Double Linked List Mahasiswa
1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 5
Nim : 20304050, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0
```

2.2.3 Pertanyaan

1. Kedua statement tersebut saling berhubungan dalam menghapus node pertama.
 - `head = head.next;` Statement ini akan menggeser pointer head agar diganti ke node yang semula merupakan node kedua. Dengan demikian, node pertama yang asli sudah tidak lagi dianggap sebagai awal dari list.
 - `head.prev = null;` Kemudian setelah head digeser ke node kedua (yang saat ini menjadi head baru), pointer prev dari head baru ini harus diatur menjadi null. Ini karena head baru yang sekarang adalah node paling depan, dan tidak ada node sebelumnya. Sehingga statement ini akan memutuskan hubungan ke node yang akan dihapus.
2. Modifikasi kode program pada method `removeFirst()` dan `removeLast()` di class `DoubleLinkedList08`.

```
public void removeFirst() {
    if (isEmpty()) {
        System.out.println("List Kosong, Tidak Bisa Dihapus.");
        return;
    }
    System.out.println("Data Sudah Berhasil Dihapus. Data Yang
Terhapus Adalah");
    head.data.tampil();
    if (head == tail) {
        head = tail = null;
    } else {
        head = head.next;
        head.prev = null;
    }
}

public void removeLast() {
    if (isEmpty()) {
        System.out.println("List Kosong, Tidak Bisa Dihapus.");
        return;
    }
    System.out.println("Data Sudah Berhasil Dihapus. Data Yang
Terhapus Adalah");
    tail.data.tampil();
    if (head == tail) {
        head = tail = null;
    } else {
        tail = tail.prev;
        tail.next = null;
    }
}
```

Hasil Verifikasi :

```
Pilih Menu : 5
Nim : 11223344, Nama : Ron Weasley, Kelas : Gryffindor, IPK : 3.8
Nim : 20304050, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0
Nim : 60708090, Nama : Harry Potter, Kelas : Gryffindor, IPK : 3.9
```

Menu Double Linked List Mahasiswa

1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar

Pilih Menu : 3

Data Sudah Berhasil Dihapus. Data Yang Terhapus Adalah

Nim : 11223344, Nama : Ron Weasley, Kelas : Gryffindor, IPK : 3.8

Menu Double Linked List Mahasiswa

1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar

Pilih Menu : 4

Data Sudah Berhasil Dihapus. Data Yang Terhapus Adalah

Nim : 60708090, Nama : Harry Potter, Kelas : Gryffindor, IPK : 3.9

Menu Double Linked List Mahasiswa

1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
0. Keluar

Pilih Menu : 5

Nim : 20304050, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0

2.3 Latihan Praktikum

2.3.1 Kode Program

Modifikasi kode program pada class DoubleLinkedList08 :

```
package Jobsheet13;

public class DoubleLinkedList08 {
    Node08 head;
    Node08 tail;
    int size;

    public DoubleLinkedList08() {
        head = null;
        tail = null;
        size = 0;
    }

    public boolean isEmpty() {
        return head == null;
    }

    public void addFirst(Mahasiswa08 data) {
        Node08 newNode = new Node08(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    public void addLast(Mahasiswa08 data) {
        Node08 newNode = new Node08(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
        size++;
    }
}
```

```

    public void insertAfter(String keyNim, Mahasiswa08 data) {
        Node08 current = head;
        while(current != null && !current.data.nim.equals(keyNim)) {
            current = current.next;
        }

        if (current == null) {
            System.out.println("Node Dengan NIM " + keyNim + " Tidak Ditemukan.");
            return;
        }
        Node08 newNode = new Node08(data);

        if (current == tail) {
            current.next = newNode;
            newNode.prev = current;
            tail = newNode;
        } else {
            newNode.next = current.next;
            newNode.prev = current;
            current.next.prev = newNode;
            current.next = newNode;
        }
        size++;
        System.out.println("Node Berhasil Disisipkan Setelah NIM " + keyNim);
    }

    public void print() {
        if (isEmpty()) {
            System.out.println("List Kosong.");
            return;
        }
        Node08 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }

    public Node08 search(String nim) {
        Node08 current = head;
        while (current != null) {
            if (current.data.nim.equals(nim)) {
                return current;
            }
            current = current.next;
        }
        return null;
    }
}

```



```

// Percobaan 2
public void removeFirst() {
    if (isEmpty()) {
        System.out.println("List Kosong, Tidak Bisa Dihapus.");
        return;
    }
    System.out.println("Data Sudah Berhasil Dihapus. Data Yang
Terhapus Adalah");
    head.data.tampil();
    if (head == tail) {
        head = tail = null;
    } else {
        head = head.next;
        head.prev = null;
    }
    size--;
}

public void removeLast() {
    if (isEmpty()) {
        System.out.println("List Kosong, Tidak Bisa Dihapus.");
        return;
    }
    System.out.println("Data Sudah Berhasil Dihapus. Data Yang
Terhapus Adalah");
    tail.data.tampil();
    if (head == tail) {
        head = tail = null;
    } else {
        tail = tail.prev;
        tail.next = null;
    }
    size--;
}

```

```

// Tugas Praktikum
public void add(int index, Mahasiswa08 data) {
    if (index < 0) {
        System.out.println("Index Tidak Valid!");
        return;
    }
    if (index > 0 && isEmpty()) {
        System.out.println("List Masih Kosong, Otomatis Menambahkan
Di Awal.");
        addFirst(data);
        return;
    }
    if (index == 0) {
        addFirst(data);
        return;
    }

    Node08 newNode = new Node08(data);
    Node08 current = head;
    int i = 0;
    while (current != null && i < index) {
        current = current.next;
        i++;
    }
    if (current == null) {
        System.out.println("Index Melebihi Ukuran List, Otomatis
Menambahkan Di Akhir.");
        addLast(data);
    } else {
        newNode.next = current.next;
        newNode.prev = current;
        if (current.next != null) {
            current.next.prev = newNode;
        } else {
            tail = newNode;
        }
        current.next = newNode;
    }
    size++;
}

```

```

public void removeAfter(String keyNim) {
    Node08 current = head;
    while (current != null) {
        if (current.data.nim.equals(keyNim)) {
            break;
        }
        current = current.next;
    }
    if (current == null) {
        System.out.println("Node Dengan NIM " + keyNim + " Tidak
Ditemukan.");
        return;
    }
    if (current.next == null) {
        System.out.println("Tidak Ada Node Setelah NIM " + keyNim +
", Tidak Ada Data Yang Dapat Dihapus.");
        return;
    }

    if (current.next == tail) {
        removeLast();
    } else if (current == head) {
        removeFirst();
    } else {
        System.out.println("Data Sudah Berhasil Dihapus. Data Yang
Terhapus Adalah");
        current.next.data.tampil();
        current.prev.next = current.next;
        current.next.prev = current.prev;
    }
    size--;
}

```

```

public void remove(int index) {
    if (index < 0) {
        System.out.println("Index Tidak Valid!");
        return;
    }
    if (head == tail && index > 0) {
        System.out.println("List Hanya Memiliki Satu Elemen, Tidak
Ada Data Yang Dapat Dihapus.");
        return;
    }
    if (index == 0) {
        removeFirst();
        return;
    }

    Node08 current = head;
    int i = 0;
    while (current != null && i < index) {
        current = current.next;
        i++;
    }
    if (current == null) {
        System.out.println("Index Melebihi Ukuran List, Tidak Ada
Data Yang Dapat Dihapus.");
        return;
    }
    System.out.println("Data Sudah Berhasil Dihapus. Data Yang
Terhapus Adalah");
    current.data.tampil();
    if (current == tail) {
        removeLast();
    } else {
        current.prev.next = current.next;
        if (current.next != null) {
            current.next.prev = current.prev;
        }
    }
    size--;
}

```

```

    public Mahasiswa08 getFirst() {
        if (isEmpty()) {
            System.out.println("List Masih Kosong, Tidak Ada Data Yang
Dapat Ditampilkan.");
            return null;
        }
        return head.data;
    }

    public Mahasiswa08 getLast() {
        if (isEmpty()) {
            System.out.println("List Masih Kosong, Tidak Ada Data Yang
Dapat Ditampilkan.");
            return null;
        }
        return tail.data;
    }

    public Mahasiswa08 getIndex(int index) {
        if (index < 0) {
            System.out.println("Index Tidak Valid!");
            return null;
        }
        Node08 current = head;
        int i = 0;
        while (current != null && i < index) {
            current = current.next;
            i++;
        }
        if (current == null) {
            System.out.println("Index Melebihi Ukuran List, Tidak Ada
Data Yang Dapat Ditampilkan.");
            return null;
        }
        return current.data;
    }

    public void size() {
        if (isEmpty()) {
            System.out.println("List Masih Kosong, Jumlah Data Adalah
0.");
        } else {
            System.out.println("Jumlah Data Saat Ini Adalah " + size);
        }
    }
}

```

2.3.2 Hasil Output

```
Pilih Menu : 5
Nim : 102030, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0

Menu Double Linked List Mahasiswa
1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
8. Tambah Data Di Indeks Tertentu
9. Hapus Setelah NIM Tertentu
10. Hapus Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar
Pilih Menu : 8
Masukkan Indeks Untuk Menambahkan Data : 0
Masukkan NIM : 405060
Masukkan Nama : Draco Malfoy
Masukkan Kelas : Slytherin
Masukkan IPK : 3.9
```

```
Pilih Menu : 5
Nim : 405060, Nama : Draco Malfoy, Kelas : Slytherin, IPK : 3.9
Nim : 102030, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0

Menu Double Linked List Mahasiswa
1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
8. Tambah Data Di Indeks Tertentu
9. Hapus Setelah NIM Tertentu
10. Hapus Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar
Pilih Menu : 11
Data Pertama :
Nim : 405060, Nama : Draco Malfoy, Kelas : Slytherin, IPK : 3.9
```

Pilih Menu : 12
Data Terakhir :
Nim : 102030, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0

Menu Double Linked List Mahasiswa

1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
8. Tambah Data Di Indeks Tertentu
9. Hapus Setelah NIM Tertentu
10. Hapus Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar

Pilih Menu : 14
Jumlah Data Saat Ini Adalah 2

Pilih Menu : 9
Masukkan NIM Untuk Menghapus Data Setelah NIM : 405060
Data Sudah Berhasil Dihapus. Data Yang Terhapus Adalah
Nim : 102030, Nama : Hermione Granger, Kelas : Gryffindor, IPK : 4.0

Menu Double Linked List Mahasiswa

1. Tambah Di Awal
2. Tambah Di Akhir
3. Hapus Di Awal
4. Hapus Di Akhir
5. Tampilkan Data
6. Sisipkan Setelah NIM Tertentu
7. Cari Mahasiswa Berdasarkan NIM
8. Tambah Data Di Indeks Tertentu
9. Hapus Setelah NIM Tertentu
10. Hapus Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar

Pilih Menu : 5
Nim : 405060, Nama : Draco Malfoy, Kelas : Slytherin, IPK : 3.9