

LAPORAN PRAKTIKUM STRUKTUR DATA
IMPLEMENTASI DOUBLE LINKEDLIST PADA
PEMROGRAMAN JAVA



Oleh :

DERIEL CHAERAHMAN

NIM 2411533007

DOSEN PENGAMPU : DR. WAHYUDI, S.T, M.T
ASISTEN PRAKTIKUM : RAHMAT DWIRIZKI OLDERS

FAKULTAS TEKNOLOGI INFORMASI
DEPARTEMEN INFORMATIKA
UNIVERSITAS ANDALAS

2025

A. Pendahuluan

Praktikum ini dilakukan untuk membuat program yang dapat menyimpan/mengelola data dengan implementasi struktur data pada java yaitu Double LinkedList. Digunakan untuk situasi manipulasi data secara dinamis (fleksibel menambah dan menghapus data dalam struktur datanya) tanpa harus menggeser elemen seperti pada array. Pengaplikasiannya dalam dunia nyata berupa navigasi maju dalam browser, undo/redo dalam aplikasi, playlist lagu/video dan manajemen memori.

1. Double LinkedList (Senarai Berantai)

Merupakan struktur data (linear) berantai di mana setiap elemen (node) menyimpan referensi ke elemen berikutnya dan sebelumnya. Terdapat 3 bagian utama yaitu data(node), referensi ke node berikutnya(next) serta ke node sebelumnya(prev). Double linkedlist punya 2 traversal, yaitu next dan prev.

Node pertama (head) merujuk ke sebelumnya (prev) = null dan node terakhir (tail) merujuk ke berikutnya(next) null sebagai tanda akhir. Cocok untuk operasi penambahan/penghapusan di tengah list, karena node-nodenya tidak disimpan secara berdekatan dalam memori, melainkan saling terhubung lewat pointer.

Kelebihannya yaitu operasi penghapusan lebih efisien dibanding single linked list, kekurangannya memori usage lebih besar karena menggunakan pointer prev.

Method :

- insertAtFront() : Menambah elemen pada awal node
- insertAtEnd() : Menambah elemen pada akhir node
- display() : Menampilkan seluruh isi linkedlist.

B. Tujuan

Tujuan dari dilakukannya praktikum ini adalah :

1. Memahami dan mengaplikasikan Double LinkedList (DLL) dalam program java untuk menyimpan data.
2. Implementasi method Double LinkedList (DLL) pada program java.
3. Dapat mengaplikasikan statement (if, while), class (Scanner), dll. yang dipelajari sebelumnya ke dalam program.
4. Membuat class OOP(Object Oriented Programming) dengan implementasi Double LinkedList.

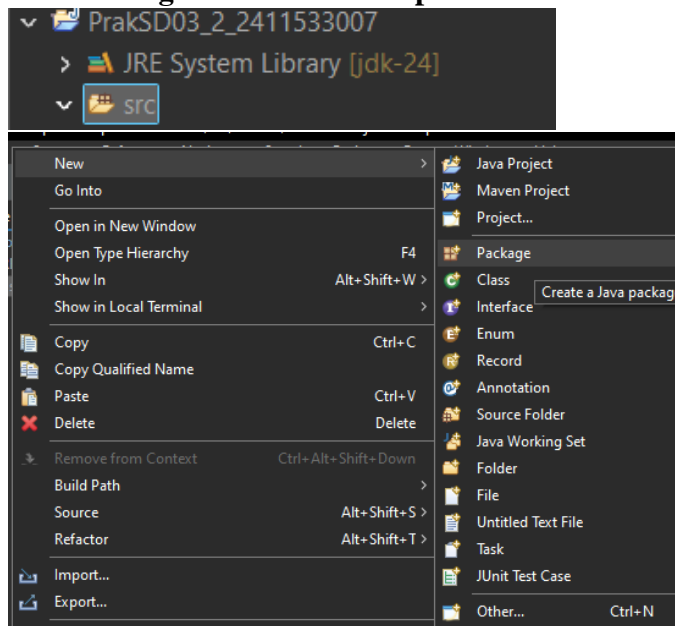
C. Langkah kerja praktikum

a. Alat dan Bahan

1. Perangkat computer atau laptop
2. Jaringan internet
3. IDE (Integreted Development Environment) direkomendasikan Eclipse IDE
4. Java JDK (Java Development Kit)

b. Package pekan 6

1. Buat new package. **Buka java project** yang telah dibuat sebelumnya, lalu **klik kanan** pada folder 'src', setelahnya akan muncul list option, pilih 'new', lalu 'Package' dan beri nama 'pekan6'.



c. Program NodeDLL

1. **Buat** terlebih dahulu new **class** dengan klik kanan pada package pekan6 dan beri nama "NodeDLL".

```
1 package pekan6;
2
3 public class NodeDLL {
```

2. Membuat attribut class, terdiri dari integer data (untuk menyimpan nilai), NodeDLL next (sebagai pointer ke node berikutnya) dan NodeDLL prev (sebagai pointer ke node sebelumnya).

```
4 // Mendefinisikan kelas Node
5 int data; //data
6 NodeDLL next; //Pointer ke next node
7 NodeDLL prev; //Pointer ke previous node
```

3. Membuat konstruktor dari class NodeSLL, menggunakan parameter int data yang berfungsi inisialisasi node baru dengan nilai dari parameter data dari konstruktor ini. This.data = data, artinya inisialisasi nilai data(attribut class) menggunakan value dari parameter konstruktor yang terpanggil saat membuat object dari class NodeDLL. this.next/prev = null, artinya set pointer next/prev ke null (node awalnya kosong)

```
9 //konstruktor
10 public NodeDLL(int data) {
11     this.data = data;
12     this.next = null;
13     this.prev = null;
14 }
15 }
```

4. Penjelasan program : Merupakan program OOP. Mendefinisikan sebuah kelas NodeDLL yang merupakan blok pembangun (Building block) untuk struktur data double linkedlist. Setiap node punya 3 komponen utama yaitu data (untuk menyimpan nilai/data node), next (pointer ke node berikutnya) dan prev (pointer ke node sebelumnya). Terdapat konstruktor untuk membuat node baru dengan inisialisasi data node dengan nilai yang diberikan menggunakan method setter, lalu pointer “next/prev” di set ke null (artinya node ini awalnya tidak terhubung ke node lain).

Memungkinkan operasi penyisipan/penghapusan node dari depan/belakang/tengah dan traversal(proses menelusuri node) baik secara maju (next) dan mundur (prev).

d. Program InsertDLL

1. Buat new class dengan klik kanan pada package pekan6 dan beri nama public class “InsertDLL”.

```
1 package pekan6;
2
3 public class InsertDLL {
```

2. Membuat method insertBegin, berfungsi menambahkan node baru di awal DLL. Memiliki parameter NodeDLL head sebagai node pertama DLL, parameter int data berfungsi inisialisasi untuk node baru. Instansi node dari class NodeDLL dengan identifier “new_node” dengan value ‘data’. Membuat objek node baru, lalu mengarahkan pointer ‘next’ node baru ke ‘head’. Jika ‘head’ tidak null, set ‘prev’-nya ke node baru. Mengembalikan node baru sebagai head baru.

```
4 //Menambahkan node di awal DLL
5 static NodeDLL insertBegin(NodeDLL head, int data) {
6     //buat node baru
7     NodeDLL new_node = new NodeDLL(data);
8     //jadikan pointer nextnya head
9     new_node.next = head;
10    //jadikan pointer prev head ke new_node
11    if (head != null) {
12        head.prev = new_node;
13    }
14    return new_node;
15 }
```

3. Membuat method insertEnd berfungsi menambahkan node baru di akhir DLL. Memiliki parameter NodeDLL head sebagai node pertama DLL dan int newData. Membuat objek node baru dengan identifier “new_node” dengan value ‘data’, lalu memeriksa apakah DLL kosong (head sama dengan null) menggunakan if statement jika true membuat node baru sebagai ‘head’, jika false/else telusuri hingga node terakhir, lalu hubungkan newNode sebagai nextnya, set ‘prev’ node baru ke node sebelumnya. Mengembalikan ‘head’.

```

17 //fungsi menambahkan node di akhir
18 public static NodeDLL insertEnd(NodeDLL head, int newData) {
19     //buat node baru
20     NodeDLL newNode = new NodeDLL(newData);
21     //jika null jadikan head
22     if (head == null) {
23         head = newNode;
24     } else {
25         NodeDLL curr = head;
26         while (curr.next != null) {
27             curr = curr.next;
28         }
29         curr.next = newNode;
30         newNode.prev = curr;
31     }
32     return head;
33 }

```

4. **Method insertAtPosition**, berfungsi menyisipkan node baru pada posisi tertentu. Memiliki parameter 'headNode' yaitu Node pertama pada DDL, 'int pos' yaitu posisi penyisipan, dan 'int new_data' yaitu value untuk node baru. Membuat objek node baru dengan identifier "new_node" dengan value 'new_data', lalu memeriksa jika "pos = 1", new_node pada posisi head. Telusuri DLL hingga posisi tujuan "pos-1", jika posisi tidak valid, menampilkan pesan/sysout dan mengembalikan head. Hubungkan 'curr' dengan 'new_node' dan 'new_node' dengan node sesudahnya, jika node sesudahnya ada, perbarui pointer balik ke node baru, lalu kembalikan head.

```

35 //fungsi menambahkan node di posisi tertentu
36 public static NodeDLL insertAtPosition (NodeDLL head, int pos, int new_data) {
37     //buat node baru
38     NodeDLL new_node = new NodeDLL(new_data);
39     if (pos == 1) {
40         new_node.next = head;
41         if (head != null) {
42             head.prev = new_node;
43         }
44         head = new_node;
45         return head;
46     }
47     NodeDLL curr = head;
48     for (int i = 1; i < pos - 1 && curr != null; i++) {
49         curr = curr.next;
50         if (curr == null) {
51             System.out.println("Posisi tidak ada");
52             return head;
53         }
54         new_node.prev = curr;
55         new_node.next = curr.next;
56         curr.next = new_node;
57         if (new_node.next != null) {
58             new_node.next.prev = new_node;
59         }
60         return head;
61     }

```

5. Membuat **method printList**, berfungsi untuk mencetak seluruh isi DDL. Memiliki parameter NodeDLL (dari class sebelumnya) head (attribut dari class NodeDLL) yang berfungsi sebagai node pertama DLL. Menelusuri DDL dari head, lalu menampilkan/sysout nilai dari setiap node dengan tambahan string/symbol "<->" sebagai penghubung antar kode.

```

63 public static void printList (NodeDLL head) {
64     NodeDLL curr = head;
65     while (curr != null) {
66         System.out.print(curr.data + " <-> ");
67         curr = curr.next;
68     }
69     System.out.println();
70 }

```

6. **Main method**, Membuat object NodeDLL dengan identifier “head”, terdiri dari node awalnya yaitu 2,3, dan 5 yang setiap node nya dihubungkan oleh pointer next/prev antar node. Menampilkan/sysout iss DLL dari head.

```

72 public static void main (String[] args) {
73     System.out.println("Nama : Deriel Chaerahan");
74     System.out.println("NIM : 2411533007");
75     System.out.println();
76     //membuat dll 2 <-> 3 <-> 5
77     NodeDLL head = new NodeDLL(2);
78     head.next = new NodeDLL(3);
79     head.next.prev = head;
80     head.next.next = new NodeDLL(5);
81     head.next.next.prev = head.next;
82
83     //cetak DLL awal
84     System.out.print("DLL awal: ");
85     printList(head);
86     System.out.println();

```

Menambahkan node baru dengan nilai integer “1” diposisi depan/’head’ menggunakan method insertBegin, lalu tampilkan dengan method ‘printList’.

```

88     //tambah 1 di awal
89     head = insertBegin(head, 1);
90     System.out.println("Simpul 1 ditambah di awal: ");
91     printList(head);
92     System.out.println();

```

Menambahkan node baru dengan nilai integer “6” dibelakang menggunakan method insertEnd, lalu ditampilkan dengan method ‘printList’.

```

94     //tambah 6 di akhir
95     System.out.println("Simpul 6 ditambah di akhir: ");
96     int data = 6;
97     head = insertEnd(head, data);
98     printList(head);
99     System.out.println();

```

Menambahkan node baru dengan nilai integer “4” disimpan pada variabel “data2”, diletakan pada posisi ke-4 (variabel pos=4) menggunakan method insertAtPosition. Menampilkan DLL menggunakan method ‘printList’.

```

101     //menambah node 4 di posisi 4
102     System.out.println("Tambah node 4 di posisi 4: ");
103     int data2 = 4;
104     int pos = 4;
105     head = insertAtPosition(head, pos, data2);
106     printList(head);
107 }
108 }

```

- Output program : Program ini menampilkan cara memanipulasi DLL dengan menggunakan method yang dibuat sebelumnya, yaitu untuk menambahkan node di awal, di akhir dan di posisi yang tertentu. Mengabungkan konsep OOP dari class NodeDLL sebelumnya dibuat, object dan method untuk mengelola DLL, yaitu memiliki 2 pointer antar nodenya.

```

<terminated> InsertDLL [Java Application] C:\Program Files\Java
Nama : Deriel Chaerahman
NIM : 2411533007

DLL awal: 2 <-> 3 <-> 5 <->

Simpul 1 ditambah di awal:
1 <-> 2 <-> 3 <-> 5 <->

Simpul 6 ditambah di akhir:
1 <-> 2 <-> 3 <-> 5 <-> 6 <->

Tambah node 4 di posisi 4:
1 <-> 2 <-> 3 <-> 4 <-> 5 <-> 6 <->

```

e. Program PencarianDLL

- Buat new class dengan klik kanan pada package pekan 6 dan beri nama “PencarianDLL”.

```

1 package pekan6;
2
3 public class PenelusuranDLL {

```

- Membuat **method forwardTraversal**, berfungsi menelusuri DLL dari head ke tail. Inisialisasi variabel NodeDLL ‘curr’ dengan ‘head’, lalu lakukan while loop hingga node terakhir(null), lalu tampilkan/sysout node saat ini, pindah ke node selanjutnya hingga loop berakhir/mencapai akhir node yaitu pointer prev merujuk ke null.

```

4 //fungsi penelusuran maju
5 static void forwardTraversal (NodeDLL head) {
6 //memulai penelusuran dari head
7 NodeDLL curr = head;
8 //lanjutkan sampai akhir
9 while (curr != null) {
10 //print data
11 System.out.print((curr.data + " <-> "));
12 //pindah ke node berikutnya
13 curr = curr.next;
14 }
15 //print spasi
16 System.out.println();
17 }

```

- Membuat **method backwardTraversal**, berfungsi menelusuri DLL dari tail ke head. Inialisasi variabel NodeDLL ‘curr’ dengan ‘tail’. While loop selama ‘curr’ tidak merujuk ke null, yang mencetak/sysout node yang dilalui(dari tail ke head), lalu lanjut pindah ke node sebelumnya/prev. Cetak spasi dengan sysout kosong.

```

19 //fungsi penelusuran mundur
20 static void backwardTraversal (NodeDLL tail) {
21     //mulai dari akhir
22     NodeDLL curr = tail;
23     //lanjut sampai head
24     while (curr != null) {
25         //cetak data
26         System.out.print(curr.data + " <-> ");
27         //pindah ke node sebelumnya
28         curr = curr.prev;
29     }
30     //cetak spasi
31     System.out.println();
32 }

```

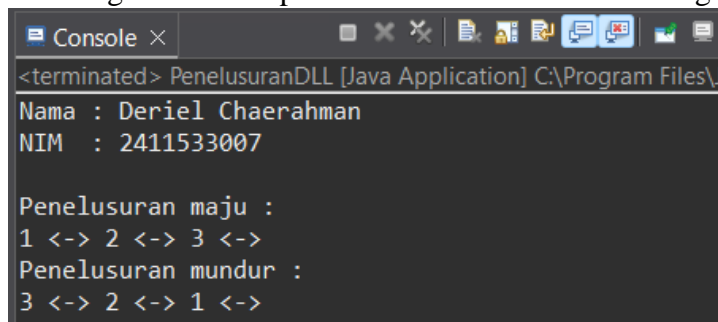
4. **Main method**, Inisialisasi DLL dengan 3 node awal bernilai 1, 2 dan 3. Lalu hubungkan tiap node dengan next/prev. Menggunakan **method forwardTraversal** untuk menampilkan/sysout node dari head hingga tail. Lalu menggunakan **method backwardTraversal** untuk menampilkan/sysout node dari headd hingga tail.

```

35 public static void main (String[] args) {
36     System.out.println("Nama : Deriel Chaerahman");
37     System.out.println("NIM : 2411533007");
38     System.out.println();
39     //cetak DLL
40     NodeDLL head = new NodeDLL(1);
41     NodeDLL second = new NodeDLL(2);
42     NodeDLL third = new NodeDLL(3);
43
44     head.next = second;
45     second.prev = head;
46     second.next = third;
47     third.prev = second;
48
49     System.out.println("Penelusuran maju : ");
50     forwardTraversal(head);
51
52     System.out.println("Penelusuran mundur : ");
53     backwardTraversal(third);
54 }
55 }

```

5. Penjelasan program : Membuat method bertipe static untuk menampilkan isi (node) dari DDL, yaitu dapat menampilkan isi dari DDL dari depan ke belakang dan menampilkan isi dari DDL dari belakang ke depan.



```

<terminated> PenelusuranDLL [Java Application] C:\Program Files\J
Nama : Deriel Chaerahman
NIM : 2411533007

Penelusuran maju :
1 <-> 2 <-> 3 <->
Penelusuran mundur :
3 <-> 2 <-> 1 <->

```


f. Program HapusDLL

1. Buat new **class** dengan klik kanan pada package pekan6 dan beri nama “HapusDLL”.

```
1 package pekan6;
2
3 public class HapusDLL {
```

2. Membuat **method delHead**, berfungsi untuk menghapus node awal (head) dari DLL. Memiliki parameter NodeDLL head merujuk node pertama DLL. Memeriksa jika head kosong maka kembalikan null. Buat variabel “tempt” insialisasi dengan head, buat ‘head’ pointer ke next node, jika ‘head’ baru ada, set “head.prev” ke null. Kembalikan ‘head’ baru.

```
4 //fungsi menghapus node awal
5 public static NodeDLL delHead(NodeDLL head) {
6     if (head == null) {
7         return null;
8     }
9     NodeDLL temp = head;
10    head = head.next;
11    if (head != null) {
12        head.prev = null;
13    }
14    return head;
15 }
```

3. Membuat **method delLast**, berfungsi untuk menghapus node akhir (tail) dari DLL. Memiliki parameter NodeDLL head merujuk node pertama DLL. Memeriksa jika head/dan node selanjutnya kosong maka kembalikan null. Buat variabel ‘curr’ inisialisasi dengan head, lalu telusuri(while loop) hingga node terakhir (pointer prev merujuk ke null). Update pointer node sebelumnya dengan “curr.prev.next = null”. Kembalikan head.

```
17 //fungsi menghapus diakhir
18 public static NodeDLL delLast(NodeDLL head) {
19     if (head == null) {
20         return null;
21     }
22     if (head.next == null) {
23         return null;
24     }
25     NodeDLL curr = head;
26     while (curr.next != null) {
27         curr = curr.next;
28     }
29     //update pointer previous node
30     if (curr.prev != null) {
31         curr.prev.next = null;
32     }
33     return head;
34 }
```

4. Membuat **method delPos**, berfungsi untuk menghapus node pada posisi tertentu. Memiliki parameter head dan pos. Memeriksa jika head kosong maka kembalikan ‘head’. Buat variabel ‘curr’ inisialisasi dengan head, lalu telusuri (for loop) sampai ke (curr.next) node yang ingin dihapus. Jika posisi tidak

ditemukan, maka kembalikan 'head'. Update pointer curr.prev jika tidak kosong menjadi hubungkan dengan node sebelum dan setelah node dihapus. Jika node yang dihapus head, maka update head. Kembalikan head yang telah di update/'head' baru.

```

36 //fungsi menghapus node posisi tertentu
37● public static NodeDLL delPos(NodeDLL head, int pos) {
38     //jika DLL kosong
39     if (head == null) {
40         return head;
41     }
42     NodeDLL curr = head;
43     //Telusuri sampai ke node yang akan dihapus
44     for (int i = 1; curr != null && i < pos; i++) {
45         curr = curr.next;
46     }
47     //jika posisi tidak ditemukan
48     if (curr == null) {
49         return head;
50     }
51     //Update pointer
52     if (curr.prev != null) {
53         curr.prev.next = curr.next;
54     }
55     if (curr.next != null) {
56         curr.next.prev = curr.prev;
57     }
58     //jika yang dihapus head
59     if (head == curr) {
60         head = curr.next;
61     }
62     return head;
63 }

```

5. Membuat **method printList**, berfungsi untuk mencetak seluruh isi DLL. Memiliki parameter NodeDLL (dari class sebelumnya) head (attribut dari class NodeDLL) yang berfungsi sebagai node pertama DLL. Menelusuri DLL dari head, lalu menampilkan/sysout nilai dari setiap node dengan tambahan string/symbol "<->" sebagai penghubung antar kode.

```

65 //fungsi mencetak DLL
66● public static void printList (NodeDLL head) {
67     NodeDLL curr = head;
68     while (curr != null) {
69         System.out.print(curr.data + " ");
70         curr = curr.next;
71     }
72     System.out.println();
73 }

```

6. **Main method**, instasi objek node baru dengan nilai awal node yaitu 1, 2, 3, 4, dan 5. Hubungkan antar node sesuai dengan next dan prev nya.

```

76● public static void main (String[] args) {
77     System.out.println("Nama : Deriel Chaerahman");
78     System.out.println("NIM : 2411533007");
79     System.out.println();
80     //buat sebuah DLL
81     NodeDLL head = new NodeDLL(1);
82     head.next = new NodeDLL(2);
83     head.next.prev = head;
84     head.next.next = new NodeDLL(3);
85     head.next.next.prev = head.next;
86     head.next.next.next = new NodeDLL(4);
87     head.next.next.next.prev = head.next.next;
88     head.next.next.next.next = new NodeDLL(5);
89     head.next.next.next.next.prev = head.next.next.next;

```

Tampilkan isi dari DDL menggunakan method `printList`.

```
91     System.out.print("DLL awal: ");
92     printList(head);
```

Gunakan method `delHead` untuk menghapus node awal, lalu tampilkan menggunakan method `printList`.

```
94     System.out.print("Setelah head dihapus: ");
95     head = delHead(head);
96     printList(head);
```

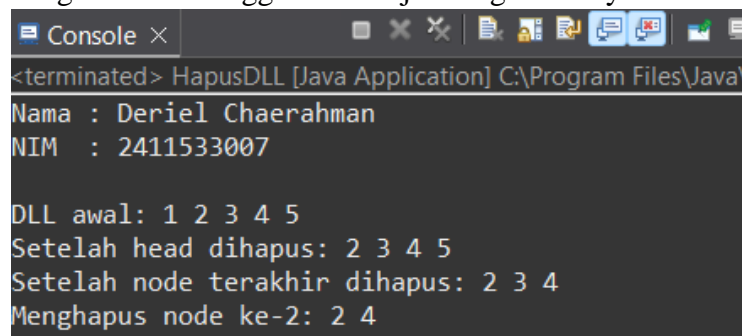
Gunakan method `delLast` untuk menghapus node akhir, lalu tampilkan menggunakan method `printList`.

```
98     System.out.print("Setelah node terakhir dihapus: ");
99     head = delLast(head);
100    printList(head);
```

Gunakan method `delPos` untuk menghapus node pada posisi ke-2, lalu tampilkan menggunakan `printList`.

```
102    System.out.print("Menghapus node ke-2: ");
103    head = delPos(head, 2);
104
105    printList(head);
106    }
107 }
```

7. Output program : Program dari class ini berfungsi untuk membuat method yang menghapus node pada posisi awal, akhir dan posisi tertentu, dengan memeriksa jika head kosong, jika tidak kosong maka telusuri hingga akhir DLL, pentingnya mengupdate pointer `next/prev` agar program DLL berjalan dengan baik sehingga tidak terjadi bug/memory leak.



```
Console x
<terminated> HapusDLL [Java Application] C:\Program Files\Java\
Nama : Deriel Chaerahan
NIM : 2411533007

DLL awal: 1 2 3 4 5
Setelah head dihapus: 2 3 4 5
Setelah node terakhir dihapus: 2 3 4
Menghapus node ke-2: 2 4
```

D. Kesimpulan

Setelah melakukan praktikum ini dapat memahami dan mengimplementasikan Double LinkedList (Senarai Berantai Ganda) yang memiliki 2 pointer yaitu `next` dan `prev`. Membuat program dengan konsep OOP seperti `NodeDLL`, membuat method dengan implementasi DDL untuk memanipulasi node. Dapat diterapkan seperti Mempelajari cara menambahkan/menghapus node secara lebih efisien pada DDL dengan memanipulasi node serta update arah antar node nya pointer '`next/prev`'. Return `NodeDLL` untuk mengupdate pointer `head/tail`.