



DOUBLE LINKED LIST

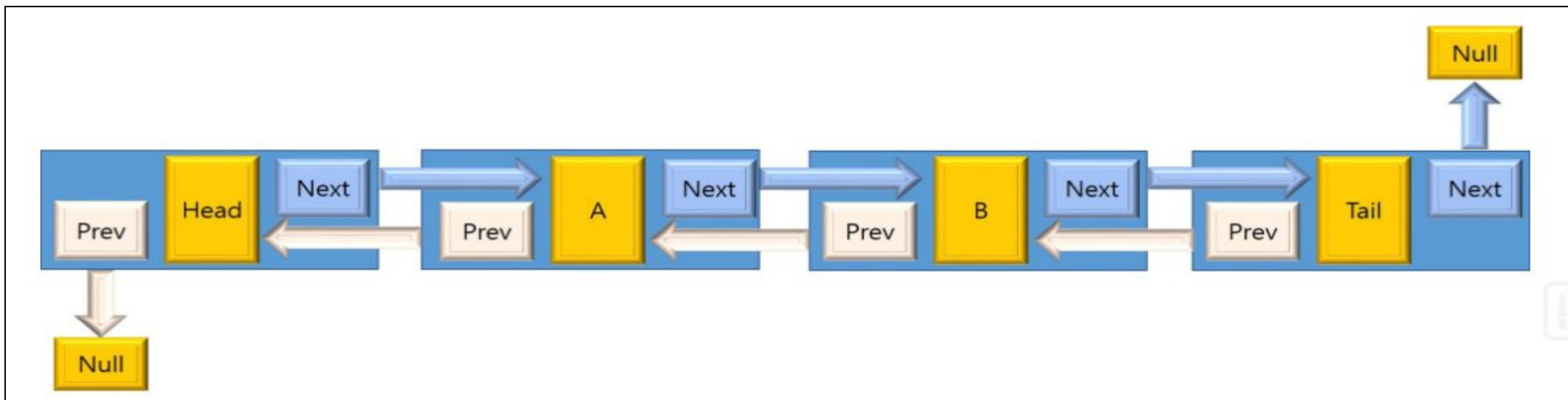
Tim Ajar Algoritma dan Struktur Data
Genap 2023/2024

Double Linked List

- Double Linked Lists memiliki dua buah pointer yaitu pointer **next** dan **prev**.
- Pointer next menunjuk pada node setelahnya dan pointer prev menunjuk pada node sebelumnya.
- Double memiliki arti field pointer-nya dua buah dan dua arah, ke node sebelum dan sesudahnya.
- Sedangkan Linked List artinya adalah node-node yang saling terhubung satu sama lain.

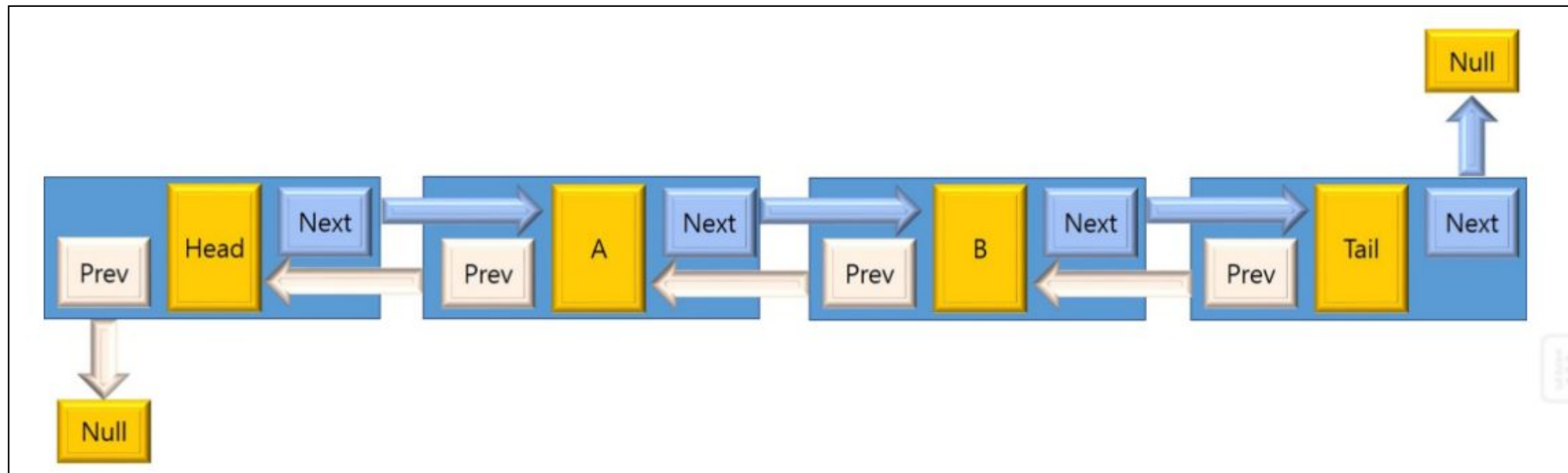
Ilustrasi Double Linked List

- Jika digambarkan double linked lists seperti gambar di bawah ini.
- Data pada awal linked lists memiliki prev sama dengan null, sedangkan data akhir memiliki next sama dengan null.



Ilustrasi Double Linked List

- Berbeda dengan single linked lists yang hanya memiliki satu pointer next saja.
- Double linked lists memiliki **dua pointer** (prev dan next) pada setiap Node.
- Oleh karena itu, atribut yang dimiliki terdiri dari tiga yaitu **Data**, **next node**, dan **previous node**.



Implementasi Double Linked List

Untuk merepresentasikan elemen data, diperlukan Node pada double link list. Implementasi dalam bahasa Java sebagai berikut:

```
public class Node {  
    int data;  
    Node prev, next;  
  
    public Node(int data, Node prev, Node next) {  
        this.data = data;  
        this.prev = prev;  
        this.next = next;  
    }  
}
```

Implementasi Double Linked List

- Untuk melakukan operasi pada double link list dibuat class double link list. Implementasi atribut dan konstruktor class double link list dalam bahasa Java sebagai berikut:

```
public class DoubleLinkLists {  
    Node head;  
    int size;  
    public DoubleLinkLists() {  
        head = null;  
        size = 0;  
    }  
}
```

Operasi Double Linked List

- Double linked lists memiliki perintah-perintah yang diimplementasikan dalam suatu method.
- Perintah tersebut terdiri dari penambahan data, penghapusan data, ataupun pengambilan data.
- Masing-masing perintah tersebut dapat dilakukan pada awal, akhir, ataupun pada indeks yang sudah ditentukan pada double linked lists.

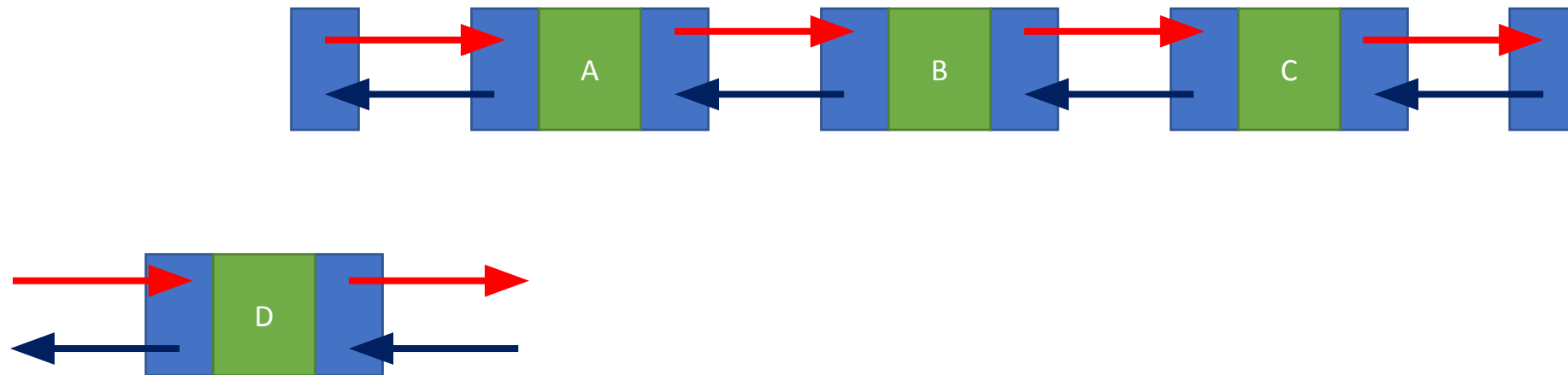
Operasi Double Linked Lists: AddFirst

- Berikut ini adalah contoh penambahan data pada bagian head linked lists.
- Kondisi sebelum dirubah dapat dilihat pada gambar berikut.



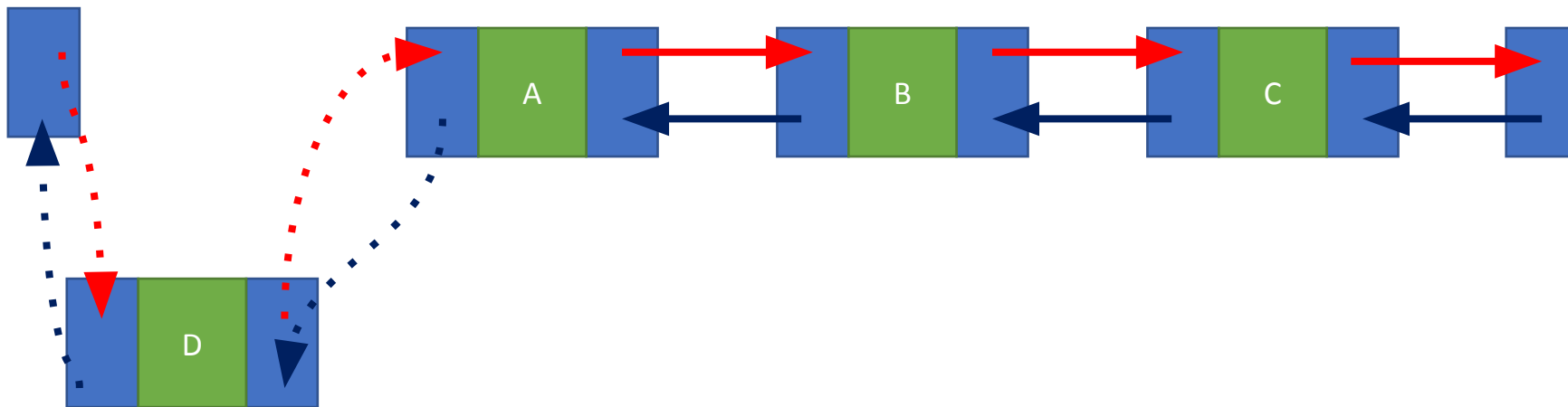
Operasi Double Linked Lists: AddFirst

- “A” adalah data pada elemen head pada awalnya. Kemudian akan ditambahkan data “D” pada bagian sebelum head.



Operasi Double Linked Lists: AddFirst

- Maka bagian header harus dipindahkan dulu ke dalam previous, kemudian Node next berisi bagian di sebelah kanan next.



Operasi Double Linked Lists: AddFirst

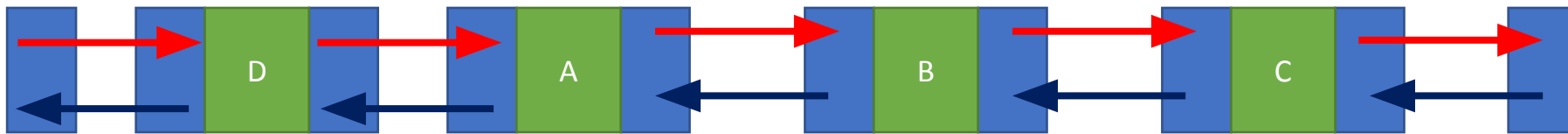
```
public void addFirst(int item) {  
    if (isEmpty()) {  
        head = new Node(null, item, null);  
    } else {  
        Node newNode = new Node(null, item, head);  
        head.prev = newNode;  
        head = newNode;  
    }  
    size++;  
}
```

- Berikut adalah implementasi method isEmpty(), digunakan untuk memastikan kondisi double linked list kosong.

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

Operasi Double Linked Lists: AddFirst

- Maka bagian header harus dipindahkan dulu ke dalam previous, kemudian Node next berisi bagian di sebelah kanan next.

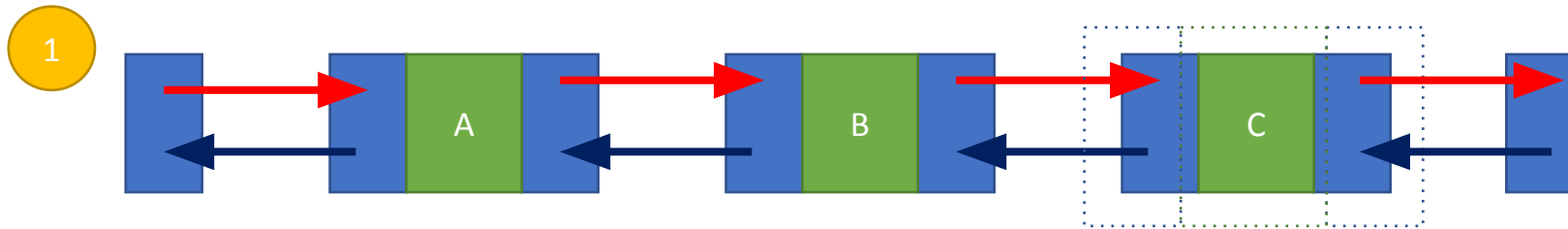


Operasi Double Linked Lists: AddLast

- Menambahkan data pada bagian akhir linked lists.
- Diawali dengan penentuan Node akhir sebagai lokasi yang akan ditambah, kemudian Node baru akan ditambahkan pada posisi lokasi next.
- Setelah itu Node Baru memiliki next yang berisi null.

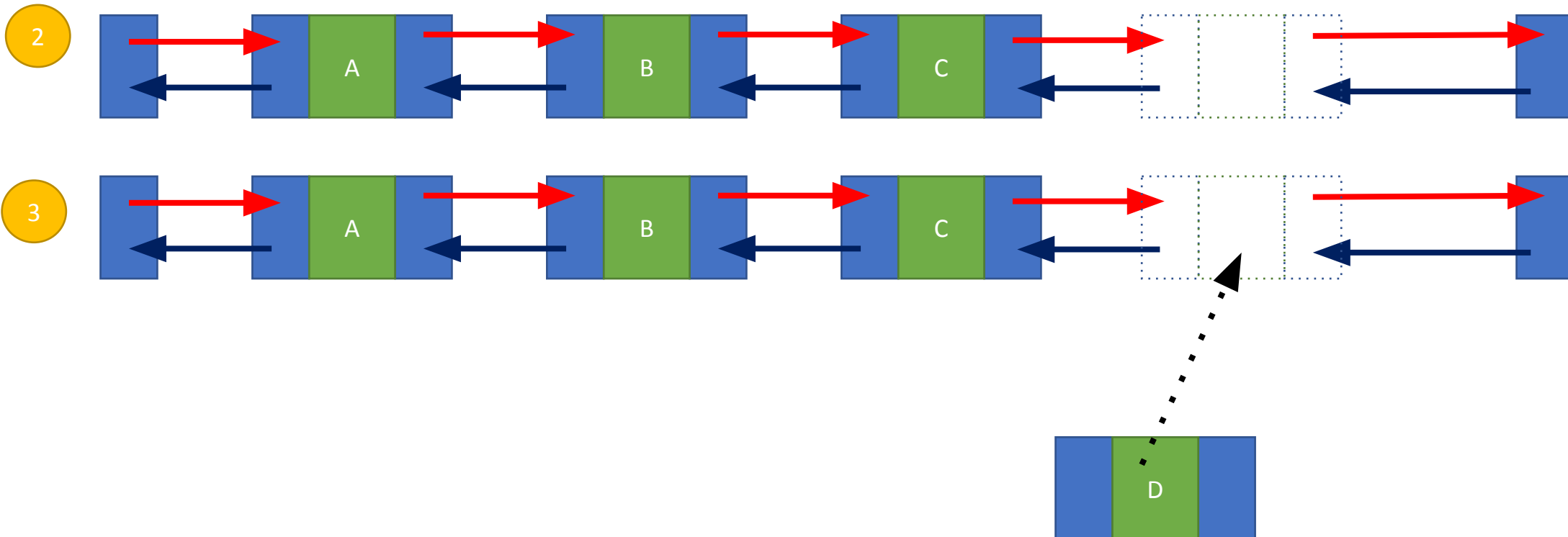
Operasi Double Linked Lists: AddLast

Menambahkan data pada bagian akhir linked lists **diawali dengan penentuan Node akhir sebagai lokasi yang akan ditambah**



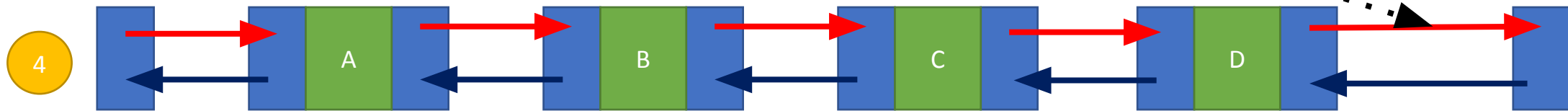
Operasi Double Linked Lists: AddLast

kemudian **Node** baru akan ditambahkan pada posisi lokasi next

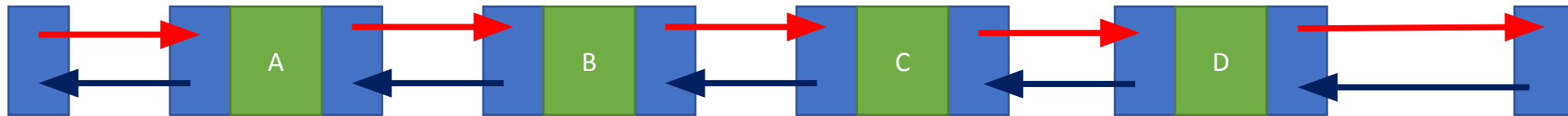


Operasi Double Linked Lists: AddLast

Node Baru memiliki next yang berisi null



Operasi Double Linked Lists: AddLast



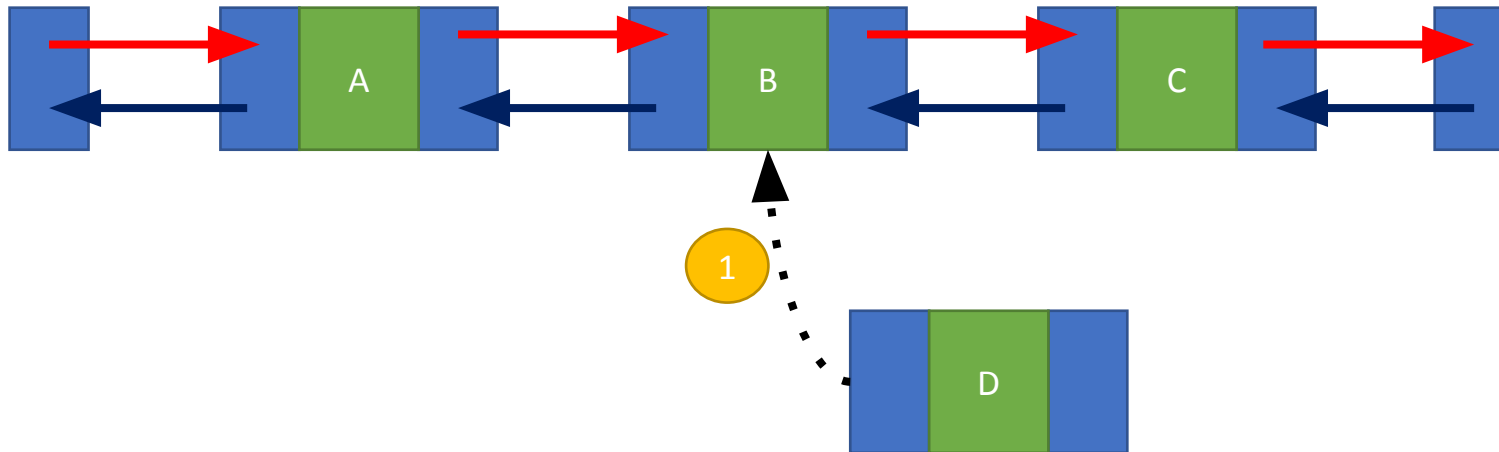
```
public void addLast(int item) {  
    if (isEmpty()) {  
        addFirst(item);  
    } else {  
        Node current = head;  
        while (current.next != null) {  
            current = current.next;  
        }  
        Node newNode = new Node(current, item, null);  
        current.next = newNode;  
        size++;  
    }  
}
```

Operasi Double Linked Lists: Add

- Operasi untuk menambah node berdasarkan indeks.
- Indeks yang akan ditambah dapat disisipkan di awal ataupun di akhir dari double linked lists.
- Terdapat **empat langkah utama** dalam penambahan data yaitu:
 - memposisikan location prev indeks data yang akan dimasukkan sebagai Node baru bagian prev,
 - location terletak pada posisi New Node bagian next,
 - Node baru terletak pada location bagian prev.next, dan
 - Node baru terletak pada bagian location bagian prev.

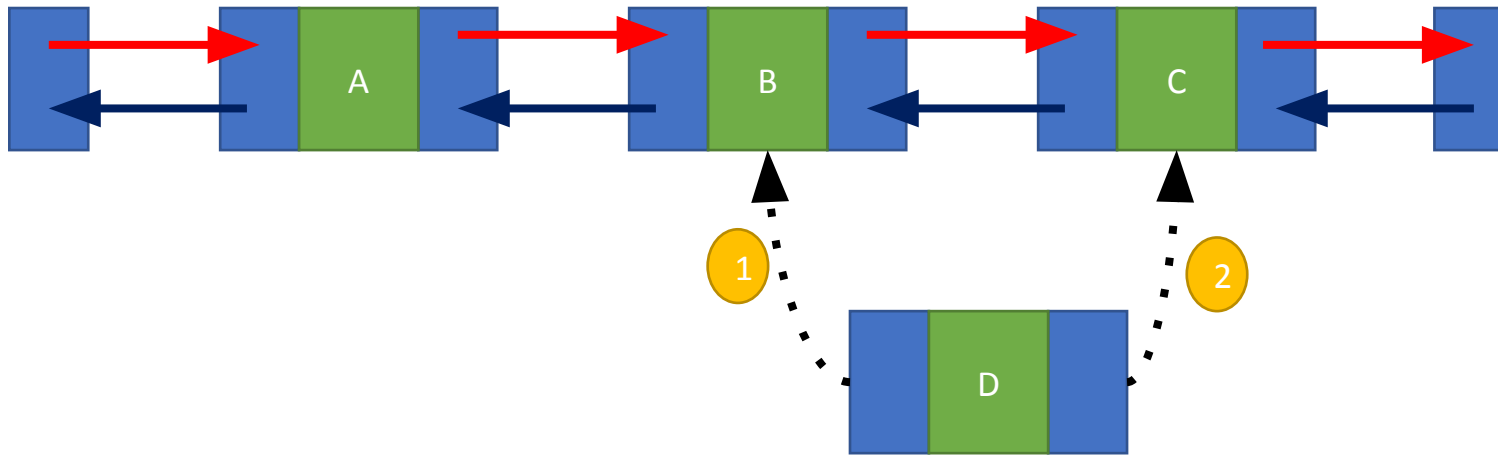
Operasi Double Linked Lists: Add

- 1 memposisikan location prev indeks data yang akan dimasukkan sebagai Node baru bagian prev,



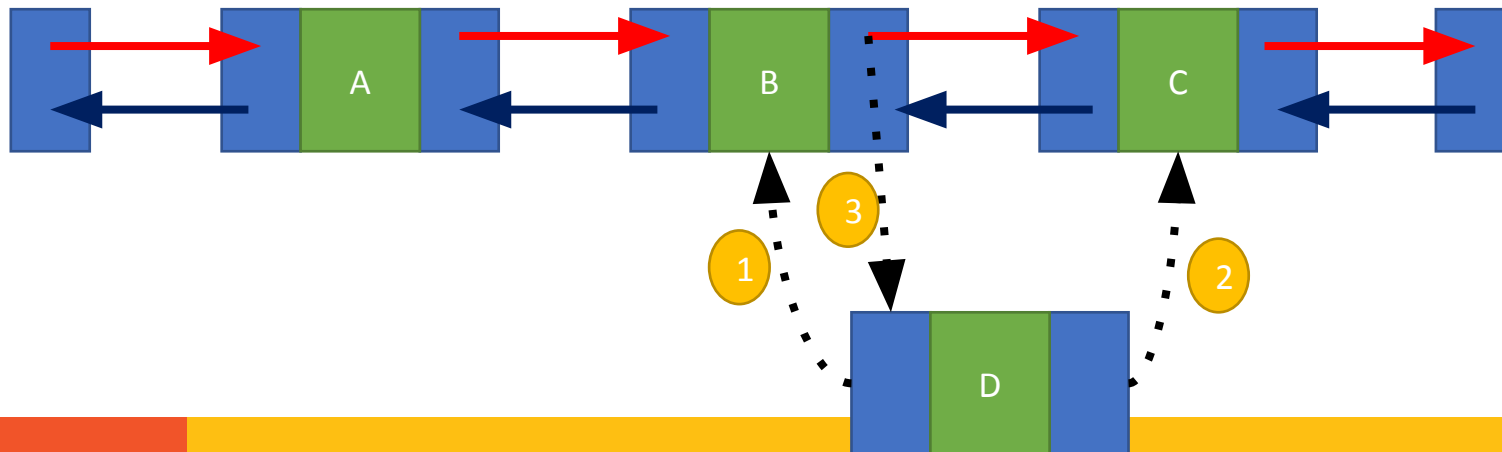
Operasi Double Linked Lists: Add

- 1 memposisikan location prev indeks data yang akan dimasukkan sebagai Node baru bagian prev,
- 2 location terletak pada posisi New Node bagian next,



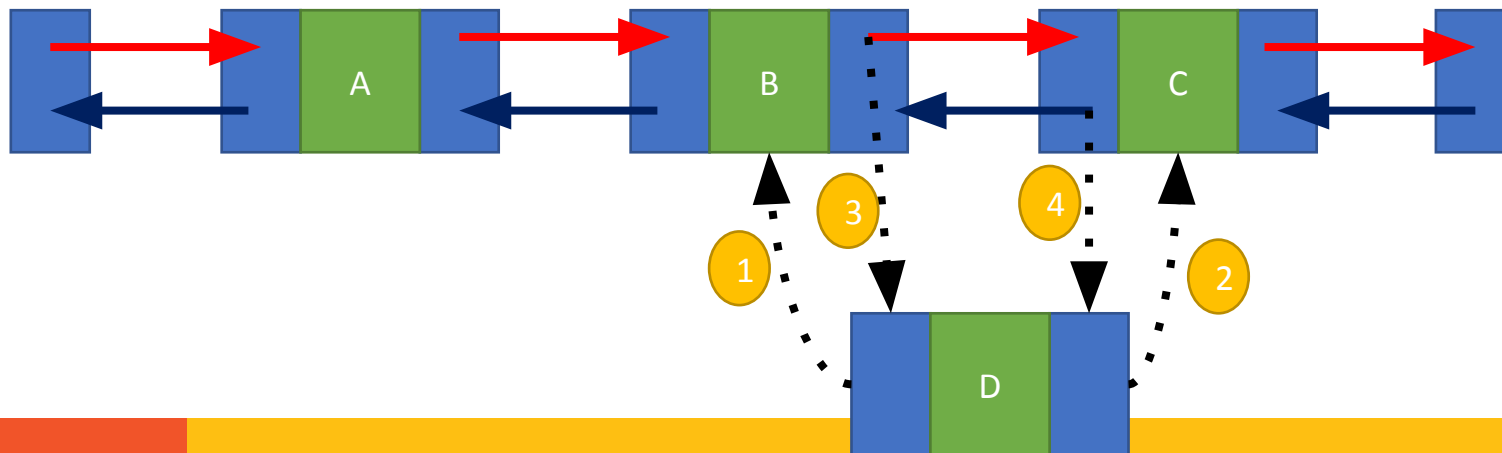
Operasi Double Linked Lists: Add

- 1 memposisikan location prev indeks data yang akan dimasukkan sebagai Node baru bagian prev,
- 2 location terletak pada posisi New Node bagian next,
- 3 Node baru terletak pada location bagian prev.next,

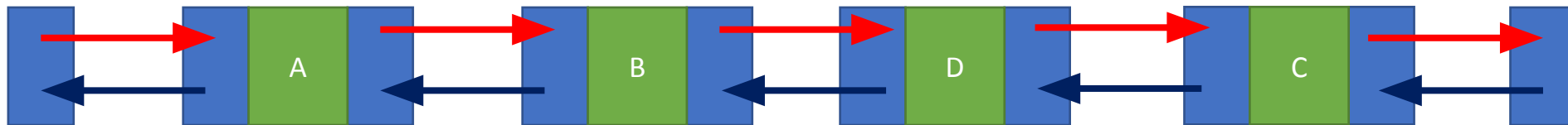


Operasi Double Linked Lists: Add

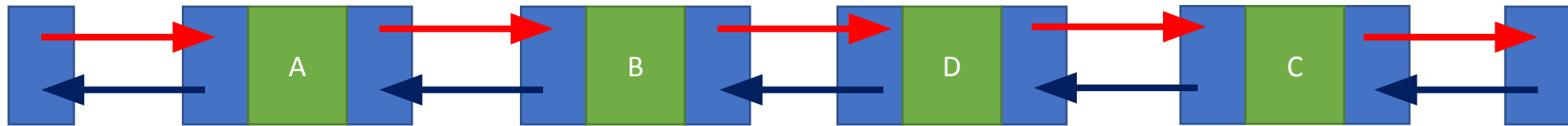
- 1 memposisikan location prev indeks data yang akan dimasukkan sebagai Node baru bagian prev,
- 2 location terletak pada posisi New Node bagian next,
- 3 Node baru terletak pada location bagian prev.next, dan
- 4 Node baru terletak pada bagian location bagian prev.



Operasi Double Linked Lists: Add



Operasi Double Linked Lists: Add



```
public void add(int item, int index) throws Exception {
    if (isEmpty()) {
        addFirst(item);
    } else if (index < 0 || index > size) {
        throw new Exception("Nilai indeks di luar batas");
    } else {
        Node current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }
        if (current.prev == null) {
            Node newNode = new Node(null, item, current);
            current.prev = newNode;
            head = newNode;
        } else {
            Node newNode = new Node(current.prev, item, current);
            newNode.prev = current.prev;
            newNode.next = current;
            current.prev.next = newNode;
            current.prev = newNode;
        }
    }
    size++;
}
```

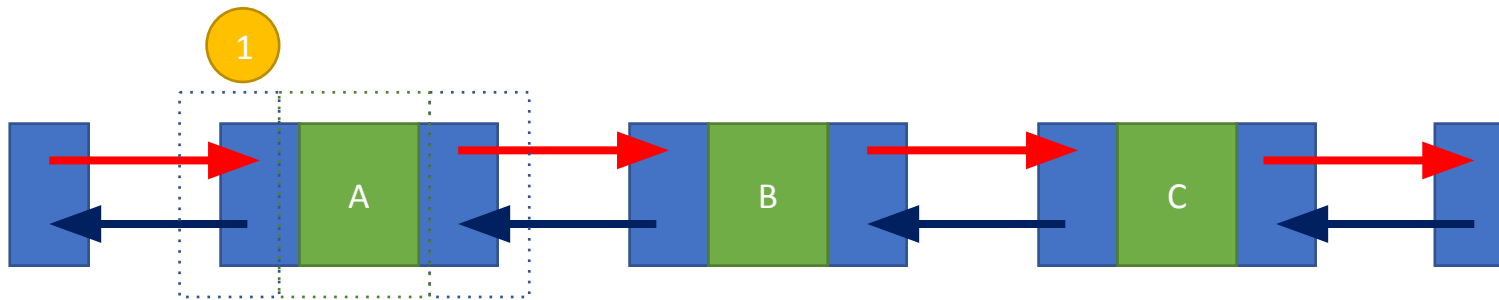



Operasi Double Linked Lists: RemoveFirst

- Menghapus data pada bagian awal dilakukan dengan pencarian lokasi awal double linked list,
- kemudian melakukan removing, dan
- menjadikan data pada bagian next menjadi bagian head.

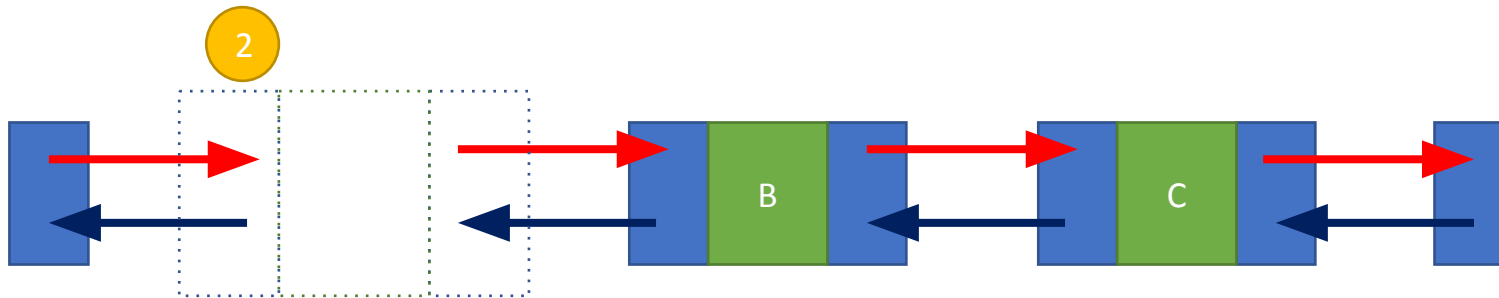
Operasi Double Linked Lists: RemoveFirst

- 1 Menghapus data pada bagian awal dilakukan dengan **pencarian lokasi awal double linked list**,



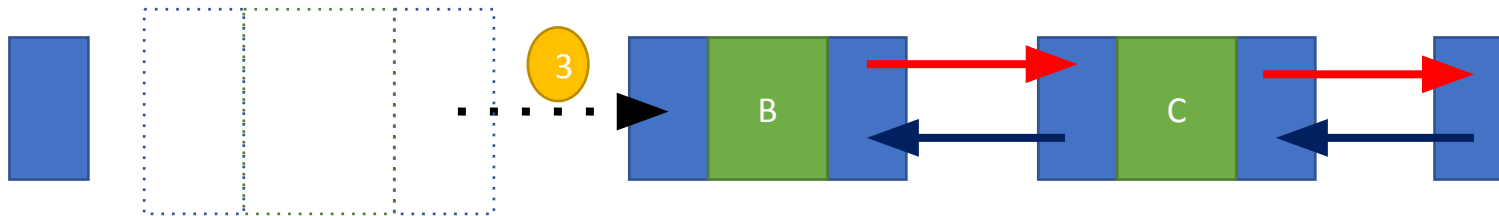
Operasi Double Linked Lists: RemoveFirst

- 1 Menghapus data pada bagian awal dilakukan dengan pencarian lokasi awal double linked list,
- 2 kemudian melakukan removing

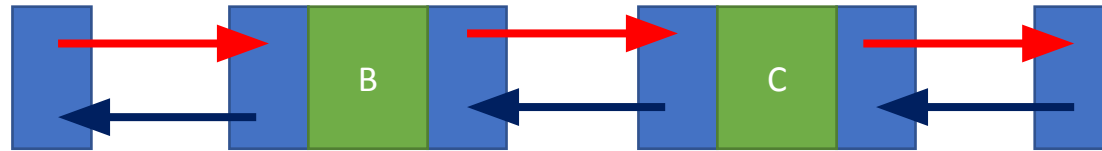


Operasi Double Linked Lists: RemoveFirst

- 1 Menghapus data pada bagian awal dilakukan dengan pencarian lokasi awal double linked list,
- 2 kemudian melakukan removing
- 3 **menjadikan data pada bagian next menjadi bagian head.**



Operasi Double Linked Lists: RemoveFirst



```
public void removeFirst() throws Exception {  
    if (isEmpty()) {  
        throw new Exception("Linked List masih kosong, tidak dapat dihapus!");  
    } else if (size == 1) {  
        removeLast();  
    } else {  
        head = head.next;  
        head.prev = null;  
        size--;  
    }  
}
```

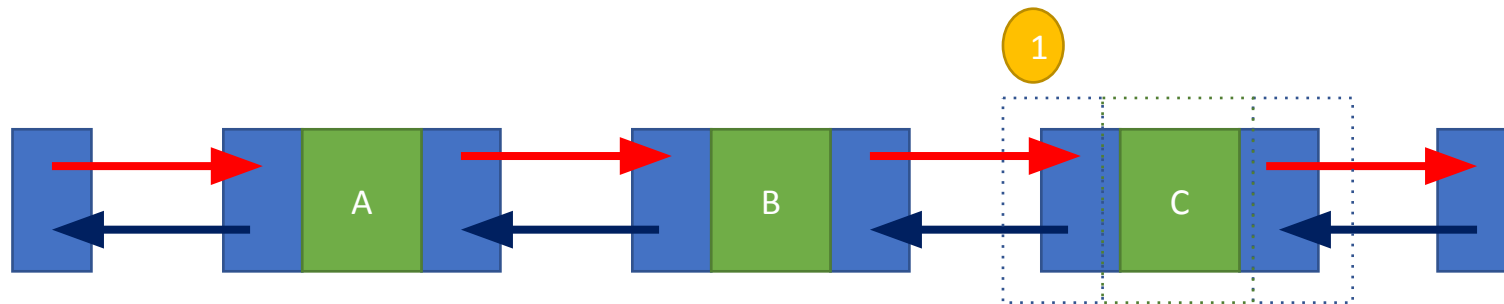


Operasi Double Linked Lists: RemoveLast

- Menghapus data pada bagian akhir elemen diawali dengan memastikan posisi yang diinginkan berada di bagian akhir,
- kemudian menghapus item yang berada di posisi tersebut.

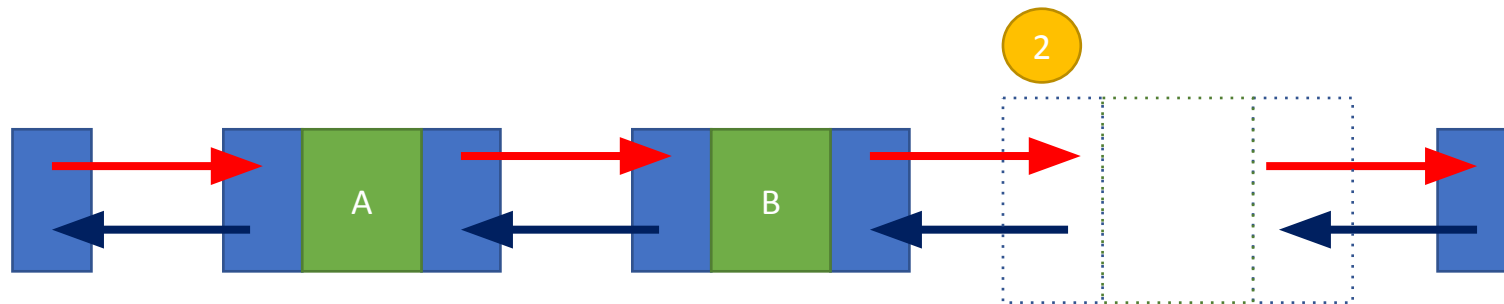
Operasi Double Linked Lists: RemoveLast

- 1 Menghapus data pada bagian akhir elemen diawali dengan memastikan posisi yang diinginkan berada di bagian akhir,

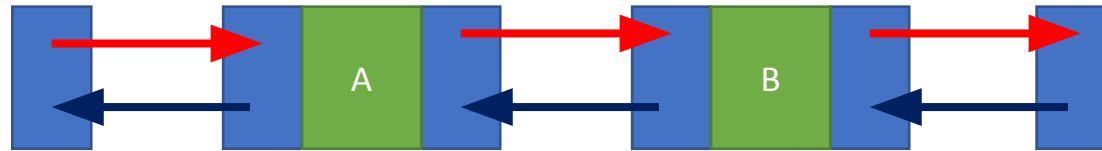


Operasi Double Linked Lists: RemoveLast

- 1 Menghapus data pada bagian akhir elemen diawali dengan memastikan posisi yang diinginkan berada di bagian akhir,
- 2 kemudian menghapus item yang berada di posisi tersebut.



Operasi Double Linked Lists: RemoveLast



```
public void removeLast() throws Exception {  
    if (isEmpty()) {  
        throw new Exception("Linked List masih kosong, tidak dapat dihapus!");  
    } else if (head.next == null) {  
        head = null;  
        size--;  
        return;  
    }  
    Node current = head;  
    while (current.next.next != null) {  
        current = current.next;  
    }  
    current.next = null;  
    size--;  
}
```

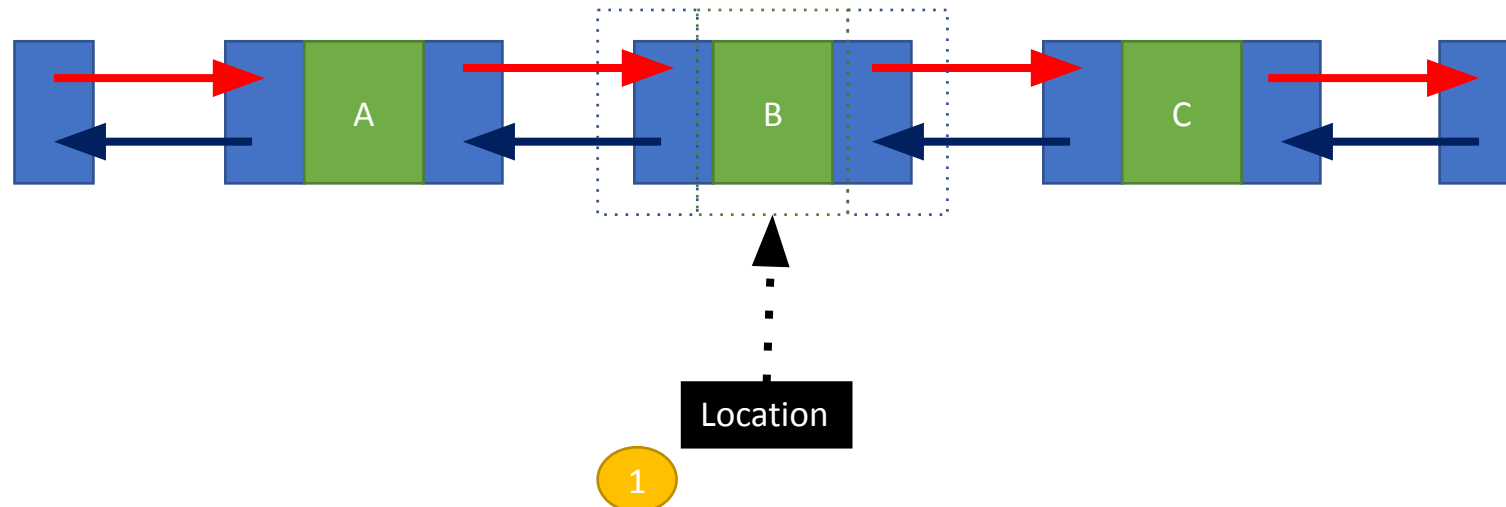
Operasi Double Linked Lists: Remove

- Operasi untuk menghapus node **berdasarkan indeks**.
- Indeks yang akan dihapus dapat disisipkan di awal ataupun di akhir dari double linked list.
- Perintah ini akan merubah posisi Node pada bagian next menjadi Node pada bagian next next.

Operasi Double Linked Lists: Remove

Operasi untuk menghapus node berdasarkan indeks.

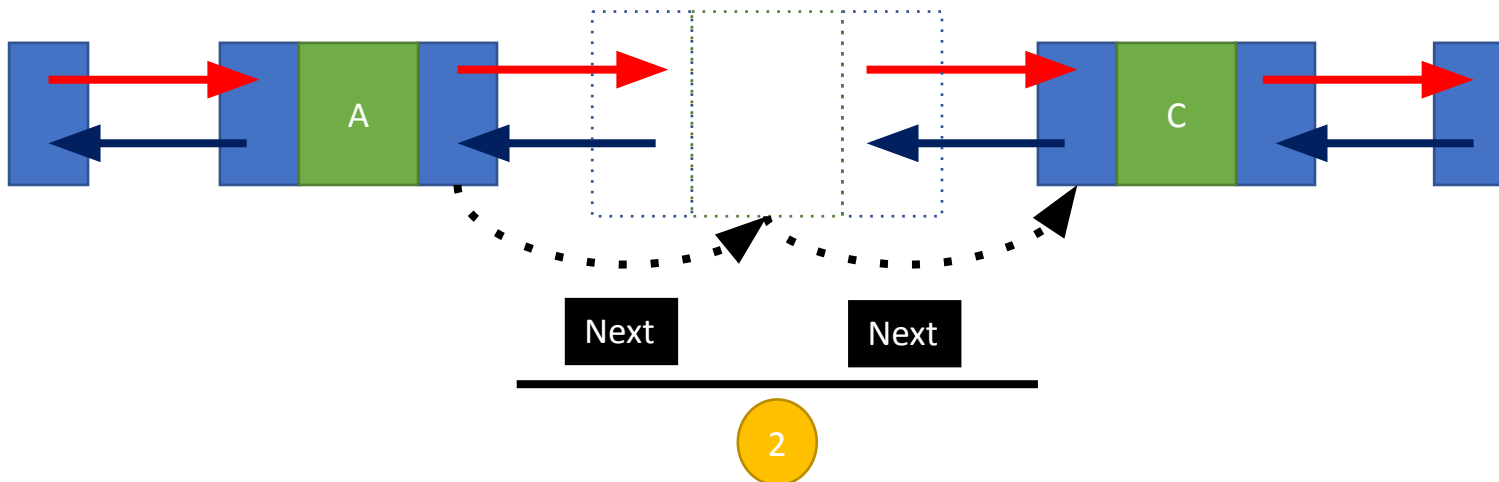
- 1 Indeks yang akan dihapus dapat disisipkan di awal ataupun di akhir dari double linked list (location).



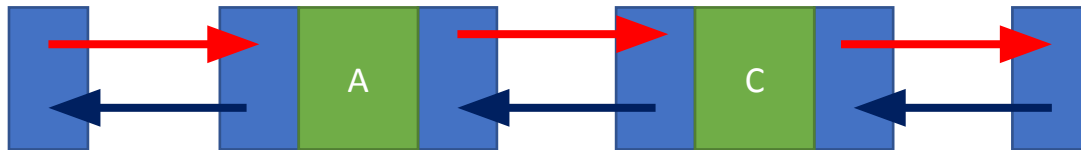
Operasi Double Linked Lists: Remove

Operasi untuk menghapus node berdasarkan indeks.

- 1 Indeks yang akan dihapus dapat disisipkan di awal ataupun di akhir dari double linked list (location).
- 2 Perintah ini akan **merubah posisi Node pada bagian next menjadi Node pada bagian next next**.



Operasi Double Linked Lists: Remove



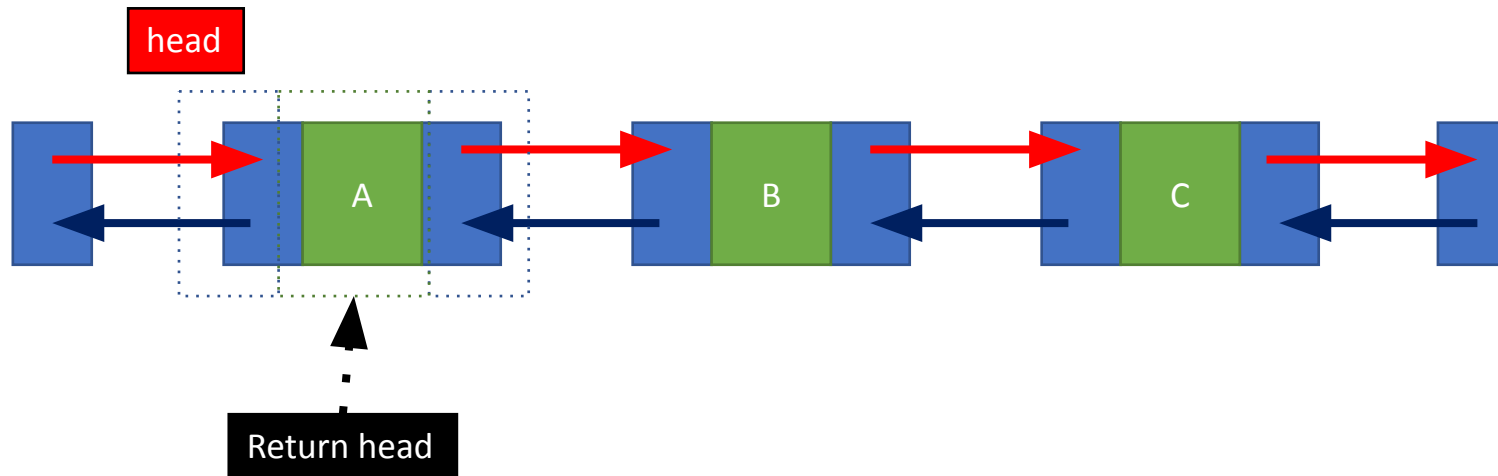


Operasi Double Linked Lists: **getFirst** / **getLast**

- Fungsi **getFirst** digunakan untuk **mengambil data di elemen paling depan (head)**.
- Prosedur **getFirst** pada double linked lists adalah dengan cara **mengembalikan nilai data pada head** untuk dapat ditampilkan.
- Berbeda dengan pengambilan data pada akhir elemen (**getLast**) digunakan untuk mengambil data pada double linked lists yang mana posisi data adalah **pada indeks terakhir atau data paling belakang**.

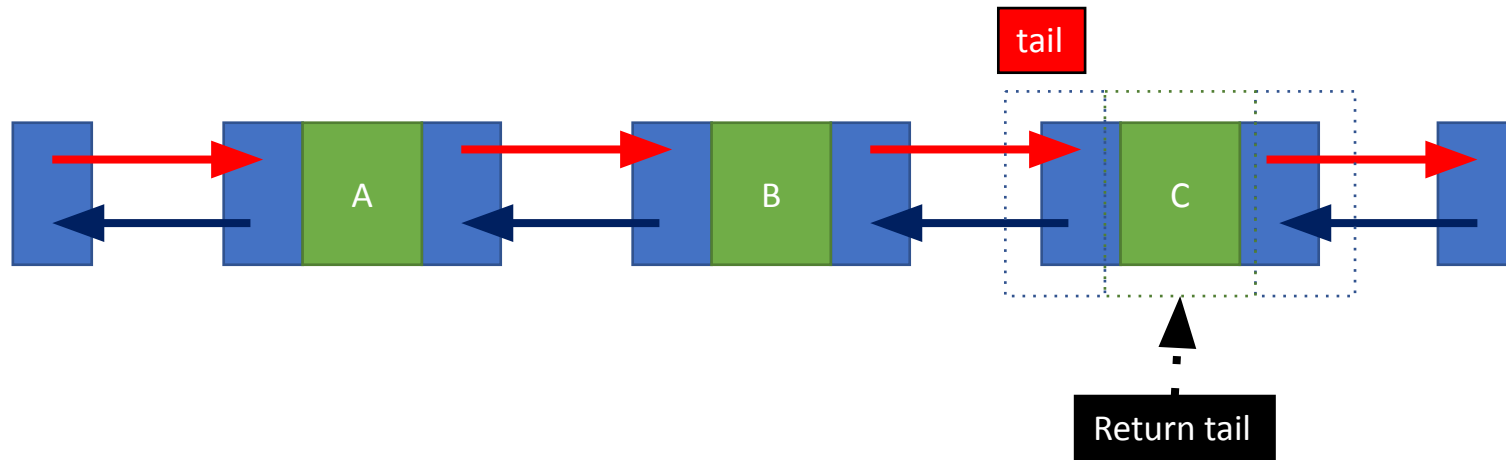
Operasi Double Linked Lists: **getFirst / getLast**

- Fungsi **getFirst** digunakan untuk mengambil data di elemen **paling depan (head)**.
- Prosedur **getFirst** pada double linked lists adalah **dengan cara mengembalikan nilai data pada head** untuk dapat ditampilkan.



Operasi Double Linked Lists: getFirst / getLast

Pengambilan data pada akhir elemen (getLast) digunakan untuk **mengambil data pada double linked lists yang mana posisi data adalah pada indeks terakhir atau data paling belakang.**



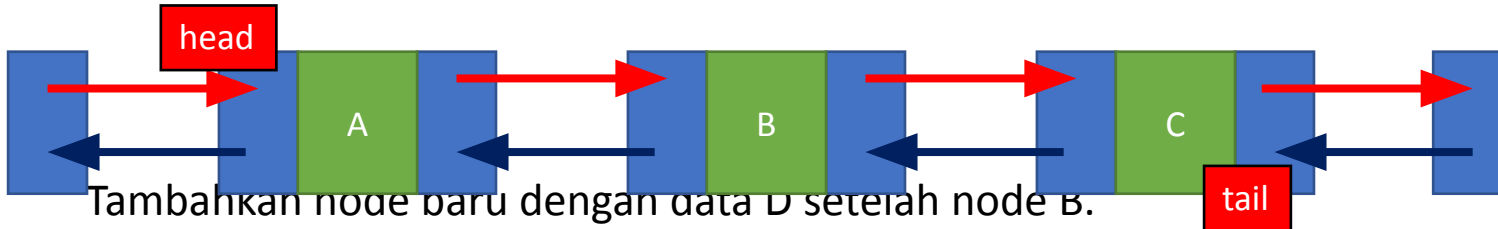
Operasi Double Linked Lists: get

- Fungsi `get(index)` digunakan jika ingin mengambil data yang dipilih pada indeks tertentu.
- Prosedur pengambilan data pada indeks tertentu adalah sebagai berikut:

```
public int get(int index) throws Exception {  
    if (isEmpty() || index >= size) {  
        throw new Exception("Nilai indeks di luar batas.");  
    }  
    Node tmp = head;  
    for (int i = 0; i < index; i++) {  
        tmp = tmp.next;  
    }  
    return tmp.data;  
}
```

Latihan

Jelaskan Langkah-langkah dari 3 node berikut dengan kondisi awal double linked list kosong secara berkelanjutan!



1. Tambahkan node baru dengan data D setelah node B.
2. Tambahkan node baru dengan data E setelah node C.
3. Tambahkan node dengan data F setelah node D.
4. Tambahkan node dengan data G pada indeks ke-3.
5. Tambahkan node dengan data H pada posisi sebelum head (sebagai head baru).
6. Tambahkan node dengan data I pada posisi setelah head baru.
7. Hapus node depan
8. Hapus node belakang
9. Hapus node yg memiliki data A.
10. Hapus node pada indeks ke-5

*Tampilkan semua data dari seluruh node pada linked list untuk setiap penambahan/penghapusan

