บทที่ 6 Double Linked List

บทเรียนย่อย

6.1 Double Linked Lists Operations and Concept

6.2 Double Linked Lists Component

6.3 Double Linked Lists Implementation

วัตถุประสงค์

- นิสิตมีความรู้ และความเข้าใจเกี่ยวกับแนวคิด และองค์ประกอบสำคัญต่าง ๆ ในการ จัดการโครงสร้างข้อมูลในรูปแบบของ Double Linked Lists
- นิสิตสามารถเขียนโปรแกรมเพื่อดำเนินการตามแนวคิดของ Double Linked Lists
- นิสิตสามารถนำแนวคิดของ Double Linked List มาประยุกต์ใช้งานในการพัฒนาโปรแกรม

บทที่ 6 Double Linked List

บทเรียนย่อย

6.1 Double Linked Lists Operations and Concept

6.2 Double Linked Lists Component

6.3 Double Linked Lists Implementation

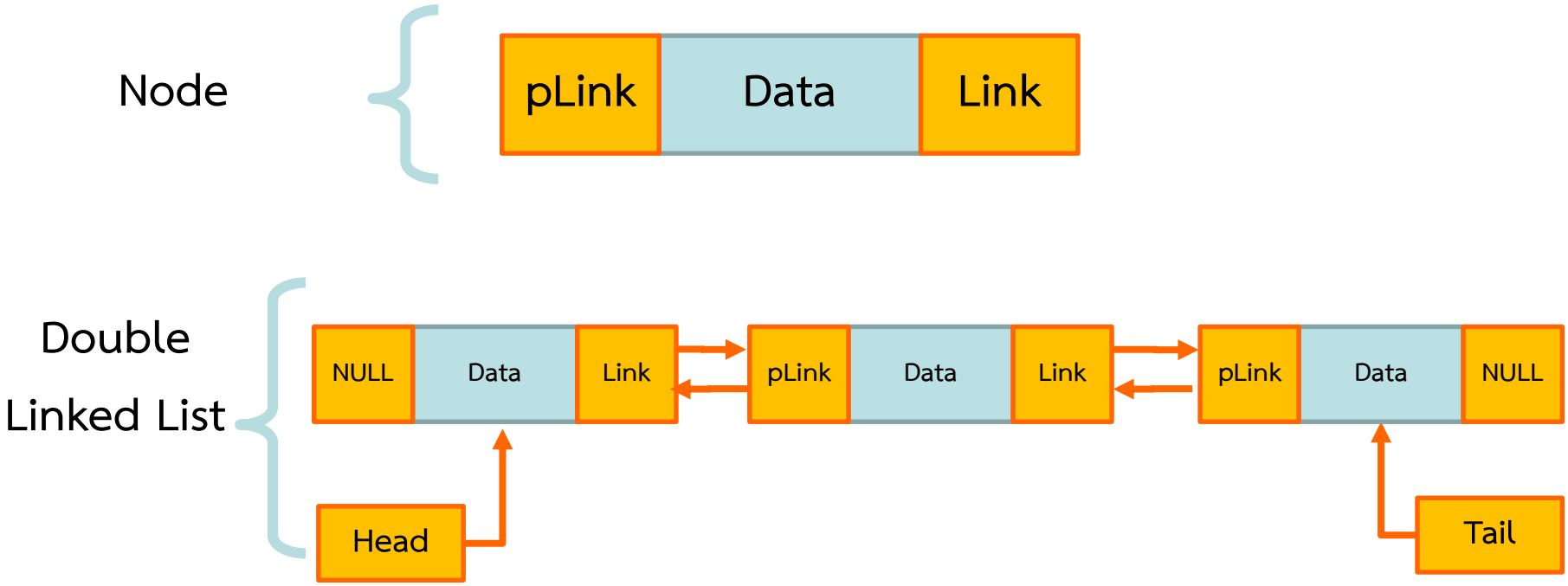
6.1 Double Linked Lists Operations and Concept

Double Linked Lists คือ การนำโครงสร้างที่รวบรวมกลุ่มของข้อมูลหรือที่เรียกว่า Linked Lists มาปรับปรุงและเพิ่มกระบวนการให้มีประสิทธิภาพในการนำไปใช้งานมากขึ้น โดยทำการเพิ่มตัวชี้ของ Node ให้สามารถระบุถึง Node ตัวก่อนหน้าได้

โดยที่ Node คือ เป็นส่วนของข้อมูล โดยใน node นั้นจะประกอบไปด้วย

1. Data หรือ Information เป็นข้อมูลที่อยู่ใน node
2. Link ซึ่งทำหน้าที่เป็นตัวเชื่อม node โดยการเก็บข้อมูลที่อยู่ของ node ตัวถัดไป
3. Previous Link ซึ่งทำหน้าที่เป็นตัวเชื่อม node โดยการเก็บข้อมูลที่อยู่ของ node ตัวก่อนหน้า

6.1 Double Linked Lists Operations and Concept [2]



หมายเหตุ

= ข้อมูล

= ตัวชี้ (Pointer)

บทที่ 6 Double Linked List

บทเรียนย่อย

6.1 Double Linked Lists Operations and Concept

6.2 Double Linked Lists Component

6.3 Double Linked Lists Implementation

6.2 Double Linked Lists Component

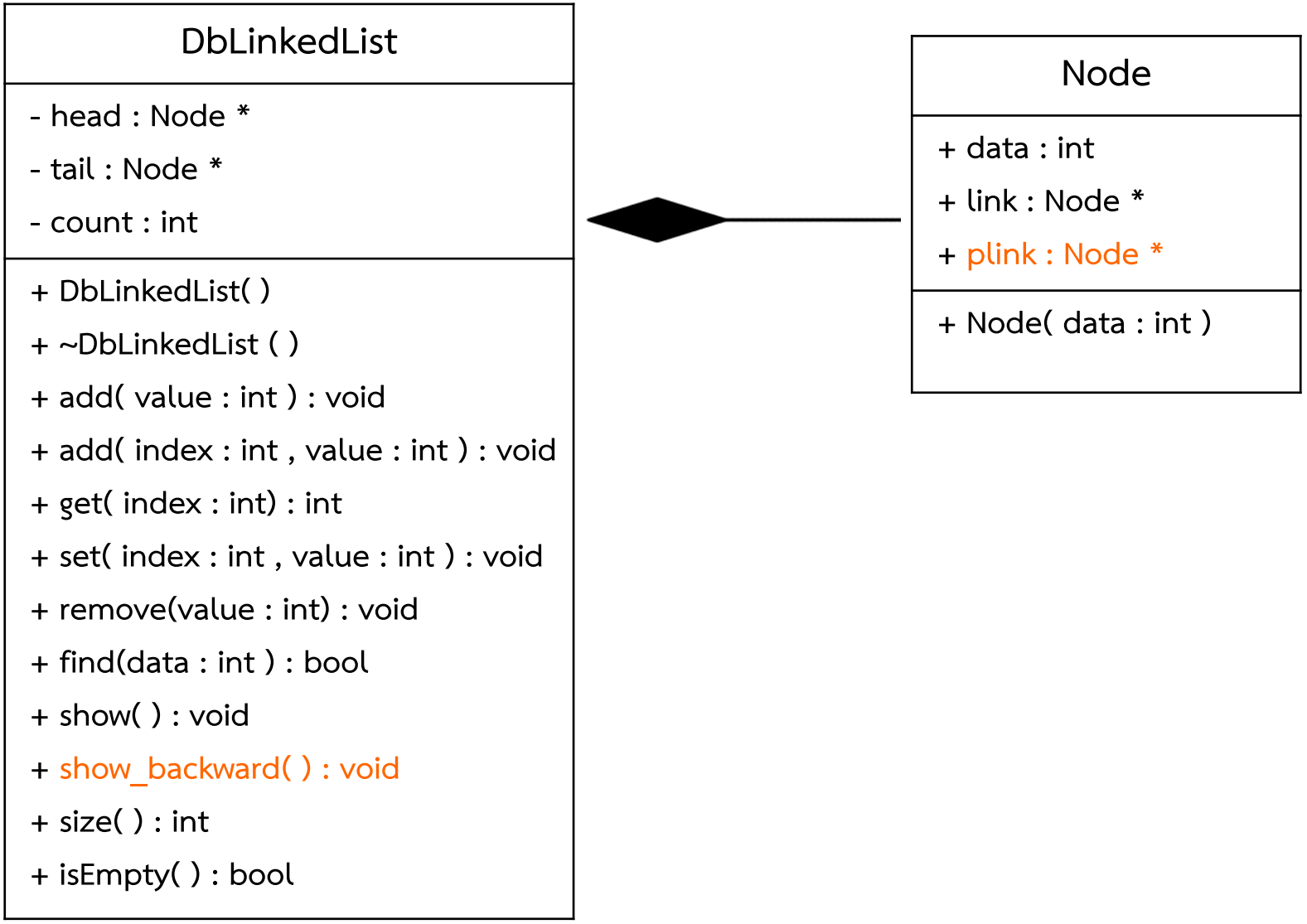
องค์ประกอบของ Double Linked Lists จะประกอบด้วย คุณสมบัติ (Property) และกระบวนการทำงาน (Method) เนื่องจากมีการสร้างขึ้นให้อยู่ในรูปแบบของคลาส (Class) ซึ่งมีรายละเอียดดังนี้

คุณสมบัติ (Property)	กระบวนการทำงาน (method)
class node	add
head	get
tail	set
count	remove
	show

6.2 Double Linked Lists Component [2]

คุณสมบัติ (Property)	กระบวนการทำงาน (method)
	size
	find
	isEmpty
	show_backward

Double Linked Lists Class



รายละเอียดคุณสมบัติของ Double Linked Lists Class

คุณสมบัติ (Property)	รายละเอียด
Node	เป็น Class ที่เก็บข้อมูล เก็บที่อยู่ของตัวถัดไป และที่อยู่ของตัวก่อนหน้า
count	ตัวแปรสำหรับการใช้นับจำนวนข้อมูลที่เก็บไว้ทั้งหมด
head	ตัวแปรสำหรับเก็บที่อยู่ของข้อมูลตัวแรก
tail	ตัวแปรสำหรับเก็บที่อยู่ของข้อมูลตัวสุดท้าย

รายละเอียดกระบวนการทำงานของ Double Linked Lists Class

กระบวนการทำงาน (method)	รายละเอียดการทำงาน
DbLinkedList()	Constructor สำหรับกำหนด ค่า head = NULL , tail = NULL และ count = 0
~DbLinkedList( )	Deconstructor สำหรับลบข้อมูลที่กำหนดขึ้นออกจากหน่วยความจำ
add(int value)	เพิ่มข้อมูลเข้า List ในพื้นที่ว่างแบบต่อท้ายตามลำดับ
add(int index , int value)	เพิ่มข้อมูลเข้า List แบบแทรกด้านหน้าในตำแหน่งที่ระบุ

รายละเอียดกระบวนการทำงานของ Double Linked Lists Class [2]

กระบวนการทำงาน (method)	รายละเอียดการทำงาน
int get(int index)	คืนค่าข้อมูลในตำแหน่งที่กำหนด จาก List
set(int index , int value)	ปรับเปลี่ยนหรือแก้ไขค่า ในตำแหน่งที่กำหนด
remove(int value)	ลบข้อมูลออกจาก List ตามค่าที่กำหนด
bool find(int data)	ค้นหาค่าตามข้อมูลที่ระบุ

รายละเอียดกระบวนการทำงานของ Double Linked Lists Class [3]

กระบวนการทำงาน (method)	รายละเอียดการทำงาน
show( )	แสดงผลข้อมูลที่มีใน List ทั้งหมด ผ่านทางหน้าจอ
show_backward( )	แสดงผลข้อมูลที่มีใน List ทั้งหมดแบบย้อนกลับ ผ่านทางหน้าจอ
int size( )	คืนค่าจำนวนข้อมูลที่มีอยู่ใน List
bool isEmpty()	ตรวจสอบค่าใน List ว่ามีข้อมูลอยู่หรือไม่

บทที่ 6 Double Linked List

บทเรียนย่อย

6.1 Double Linked Lists Operations and Concept

6.2 Double Linked Lists Component

6.3 Double Linked Lists Implementation

6.3 Double Linked Lists Implementation

การสร้างคลาส Double Linked List ภาษา C++

```
class DbLinkedList {
    private :
     class Node {
                ...
        };
        Node *head;
        Node *tail;
    public :
        DbLinkedList();
        ~DbLinkedList( );
                ...
};
```

6.3 Double Linked Lists Implementation [2]

การสร้างคลาส Double Linked List ภาษา C++

```
class DbLinkedList {
        ...
    add( int  value );
    add( int index , int  value );
    int get( int index );
    set( int index , int  value );
    remove( int value );
        ...
};
```

6.3 Double Linked Lists Implementation [3]

การสร้างคลาส Double Linked List ภาษา C++

```
class DbLinkedList {
        ...
    bool find(int data);
    show( );
    show_backward( );
    int size( );
    bool isEmpty( );
};
```

การดำเนินการใน Constructor และ Deconstructor

```
DbLinkedList :: DbLinkedList(){
    this->head = NULL;
    this->tail = NULL;
    this->count = 0;
}
```

การดำเนินการใน Constructor และ Deconstructor

```
DbLinkedList :: ~ DbLinkedList( ){
    for(int i = 1; i < this->count; i++){
        node * tmp = this->head;
        this->head = this->head->link;
        delete tmp;
        tmp = NULL;
    }
    this->head = NULL;
    this->tail = NULL;
}
```

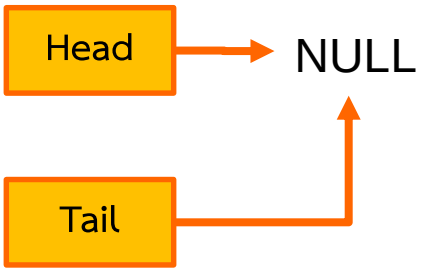
การเรียกใช้งานคลาส Double Linked List

```
int main(void){
    DbLinkedList *obj_DBlinkedlist = new DbLinkedList();
    obj_DBlinkedlist->add(5);
    obj_DBlinkedlist->add(8);
    obj_DBlinkedlist->show();

    LinkedList obj_DBlinkedlist;
    obj_DBlinkedlist.add(5);
    obj_DBlinkedlist.add(8);
    obj_DBlinkedlist.show();
}
```

แนวคิดการเพิ่มข้อมูล

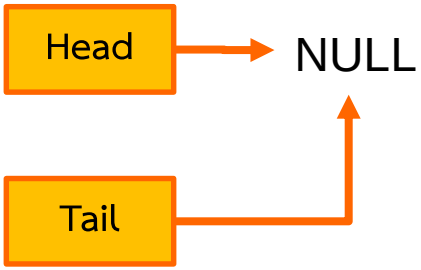
DbLinkedList obj_DBlinkedlist;



count = 0

แนวคิดการเพิ่มข้อมูล [2]

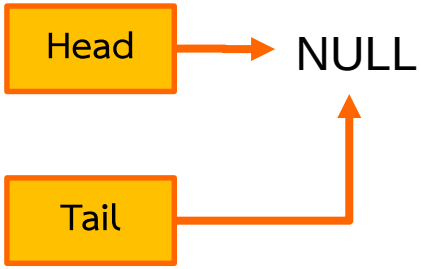
obj_DBlinkedlist.add(5);



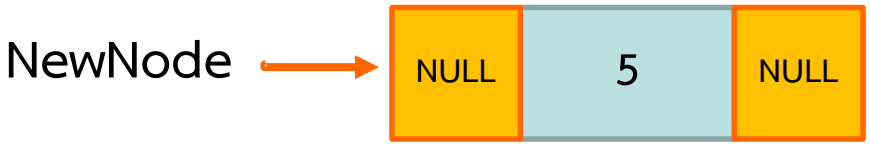
count = 0

แนวคิดการเพิ่มข้อมูล [3]

obj_DBlinkedlist.add(5);



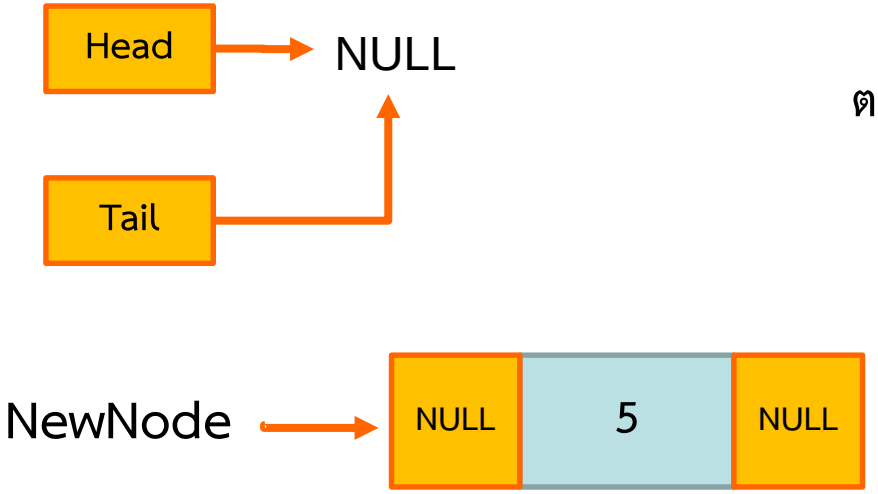
สร้าง new object จาก class node โดยกำหนดค่า data มีค่าเท่ากับ 5 และ link , plink มีค่าเป็น NULL



count = 0

แนวคิดการเพิ่มข้อมูล [4]

obj_DBlinkedlist.add(5);



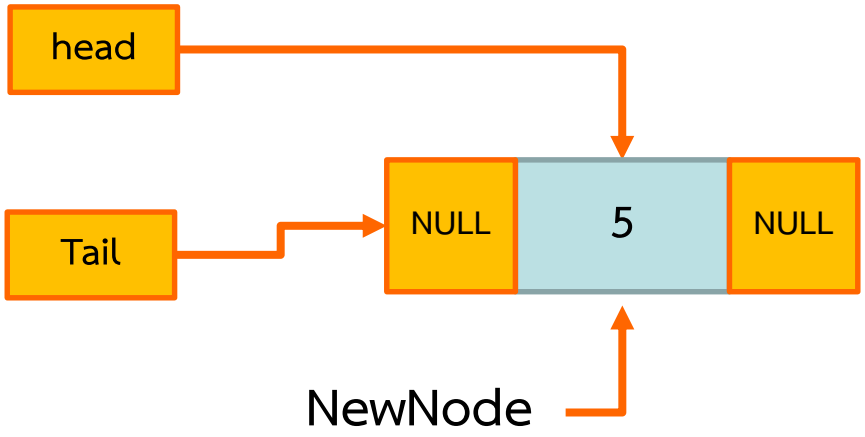
ตรวจสอบค่า head มีค่าเป็น null หรือไม่

ถ้าเป็น ทำการกำหนดค่า head และ tail ให้ เท่ากับ NewNode

ถ้าไม่เป็น ทำการนำ link ของ node ตัวสุดท้าย มากำหนดให้เท่ากับค่าของ NewNode จากนั้นนำ plink ของ NewNode มากำหนดให้เท่ากับ node ตัวก่อนหน้า และทำการเพิ่มค่า count

แนวคิดการเพิ่มข้อมูล [5]

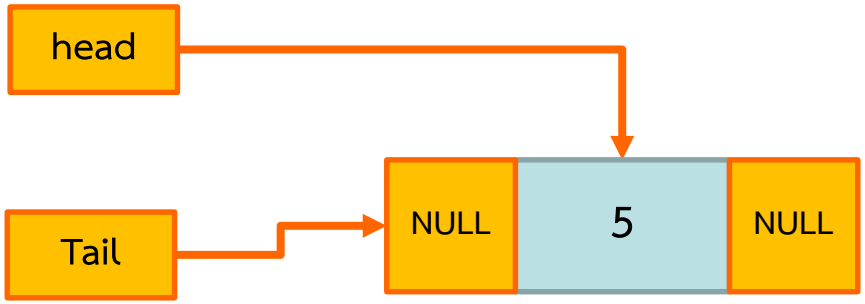
obj_DBlinkedlist.add(5);



count = 1

แนวคิดการเพิ่มข้อมูล [6]

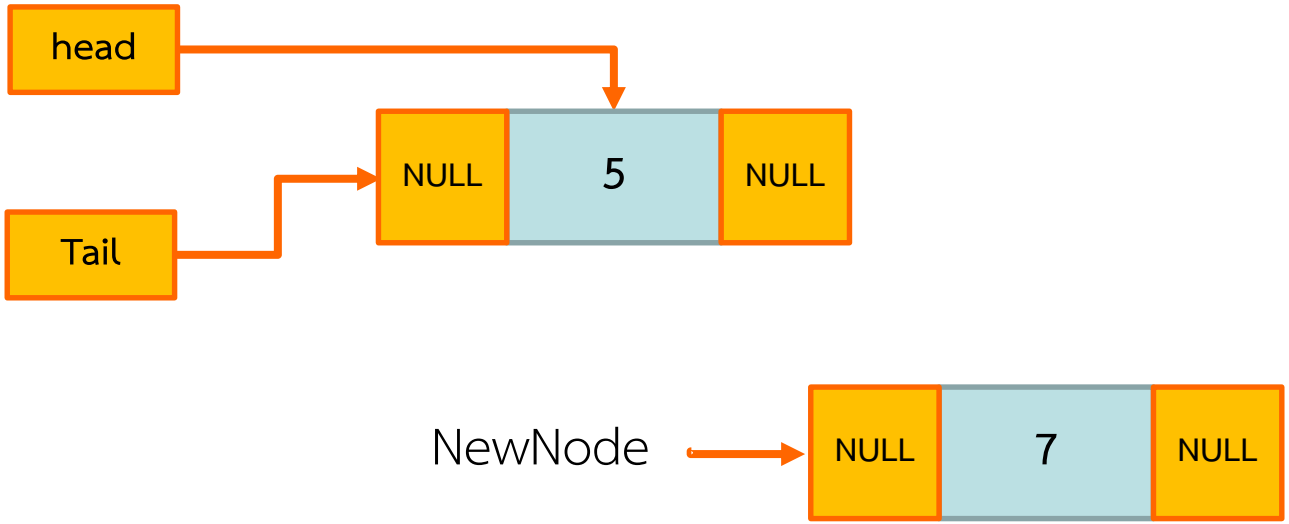
obj_DBlinkedlist.add(7);



count = 1

แนวคิดการเพิ่มข้อมูล [7]

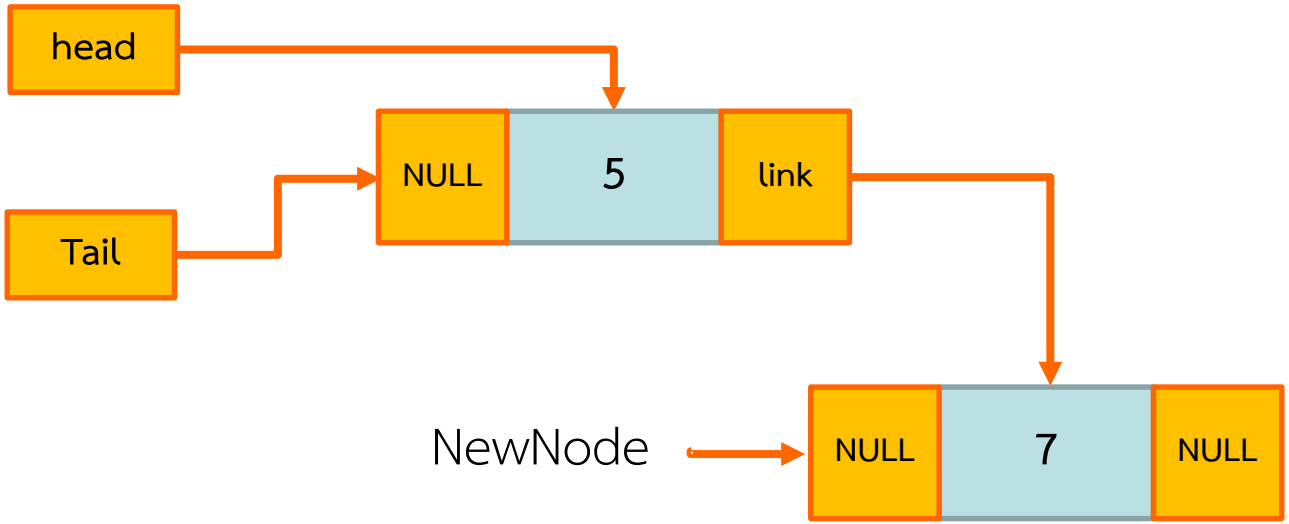
obj_DBlinkedlist.add(7);



count = 1

แนวคิดการเพิ่มข้อมูล [8]

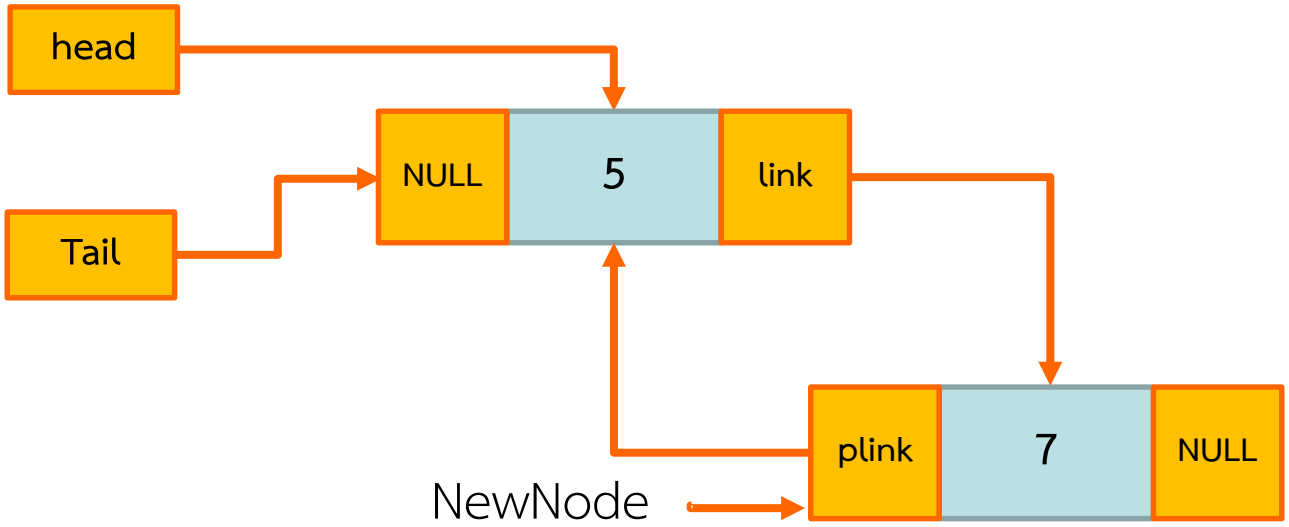
obj_DBlinkedlist.add(7);



count = 1

แนวคิดการเพิ่มข้อมูล [9]

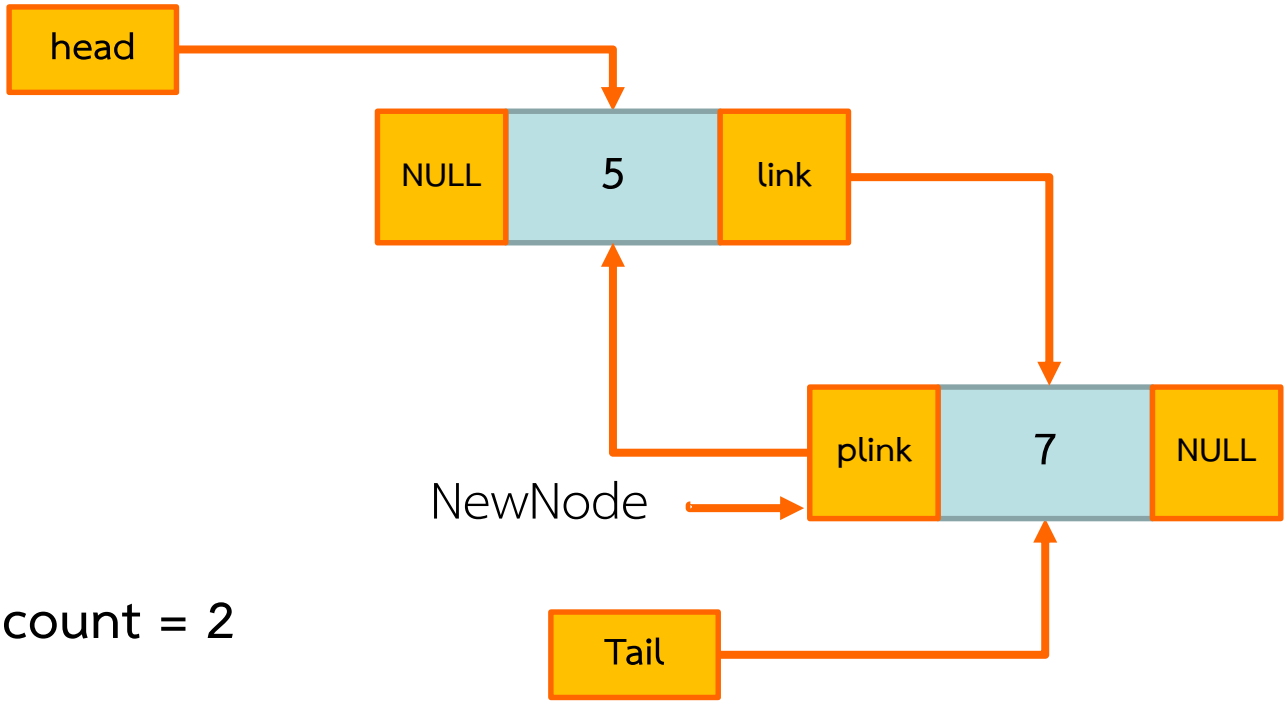
obj_DBlinkedlist.add(7);



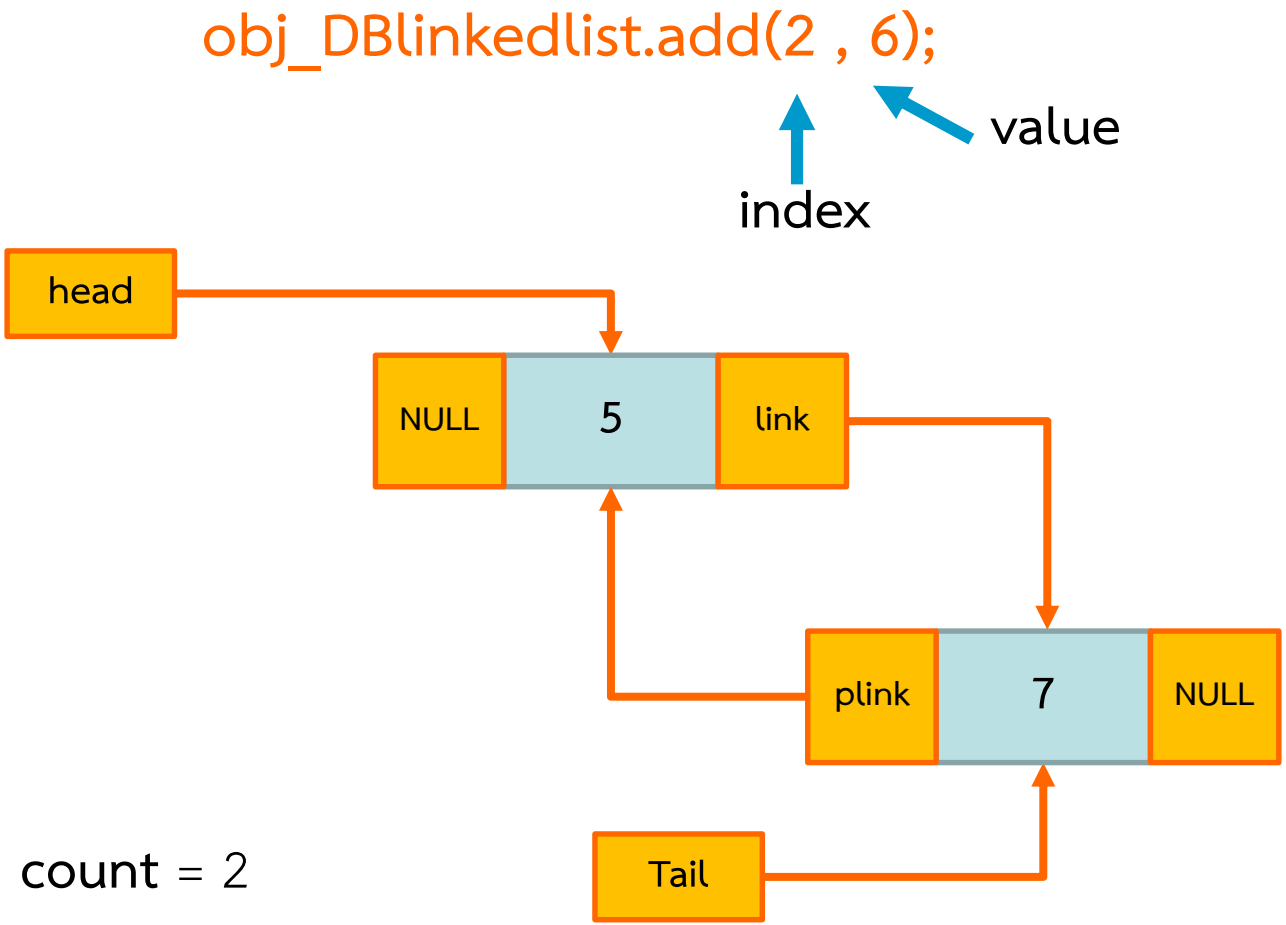
count = 1

แนวคิดการเพิ่มข้อมูล [10]

obj_DBlinkedlist.add(7);

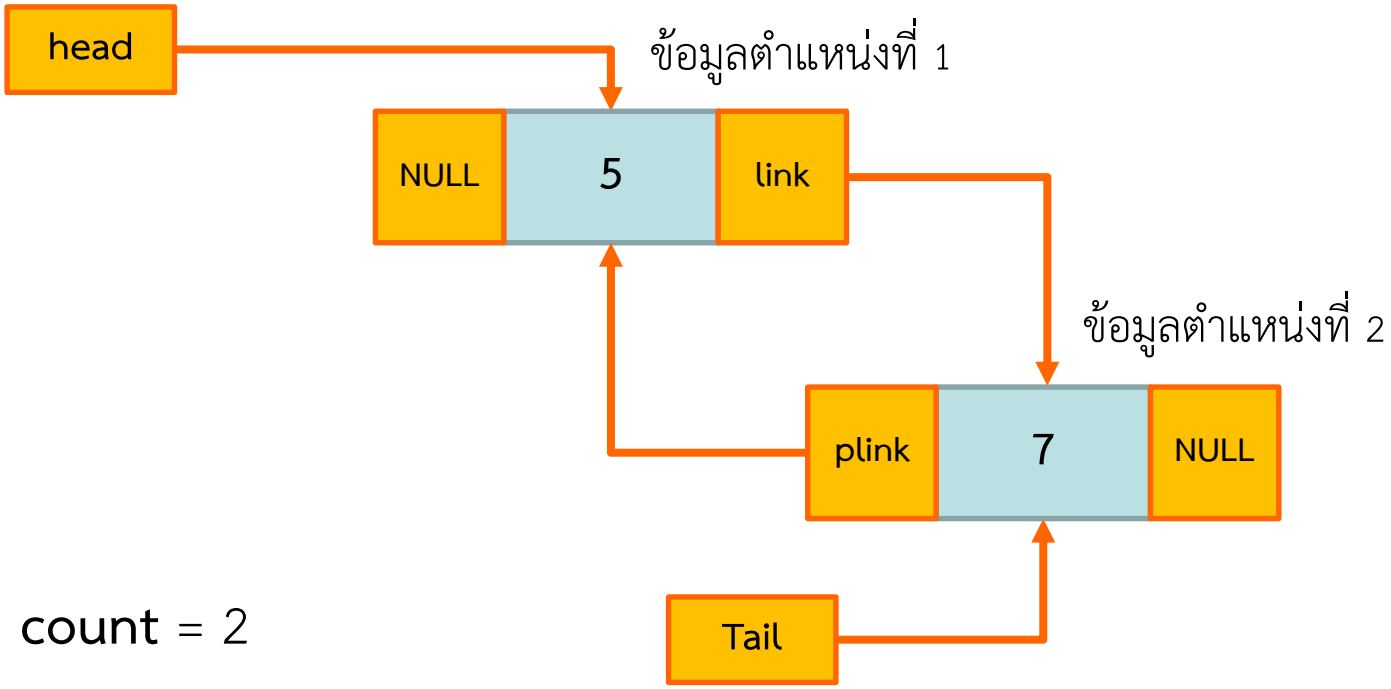


แนวคิดการเพิ่มข้อมูล กรณีระบุตำแหน่ง



แนวคิดการเพิ่มข้อมูล กรณีระบุตำแหน่ง [2]

obj_DBlinkedlist.add(2 , 6);

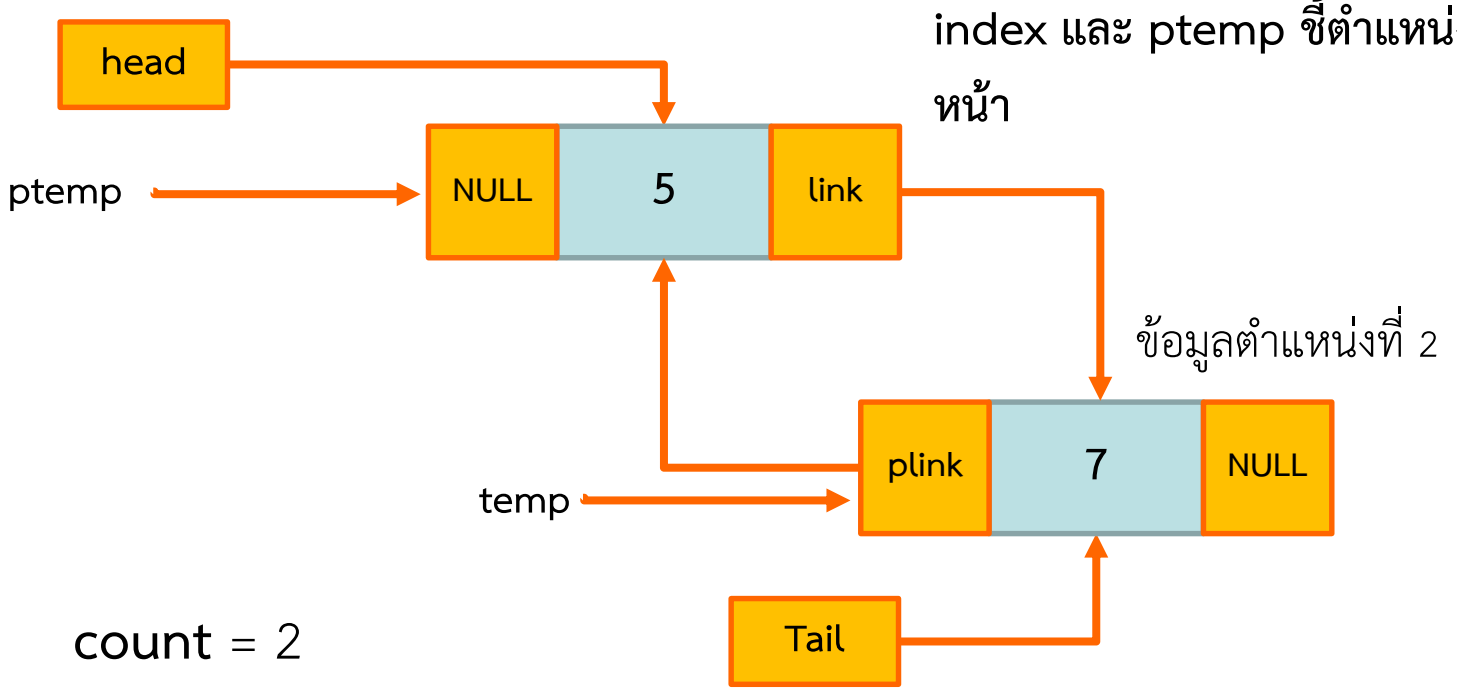


count = 2

แนวคิดการเพิ่มข้อมูล กรณีระบุตำแหน่ง [3]

obj_DBlinkedlist.add(2 , 6);

สร้างตัวชี้ temp มาชี้ที่ค่าของตำแหน่ง index และ ptemp ชี้ตำแหน่งก่อนหน้า

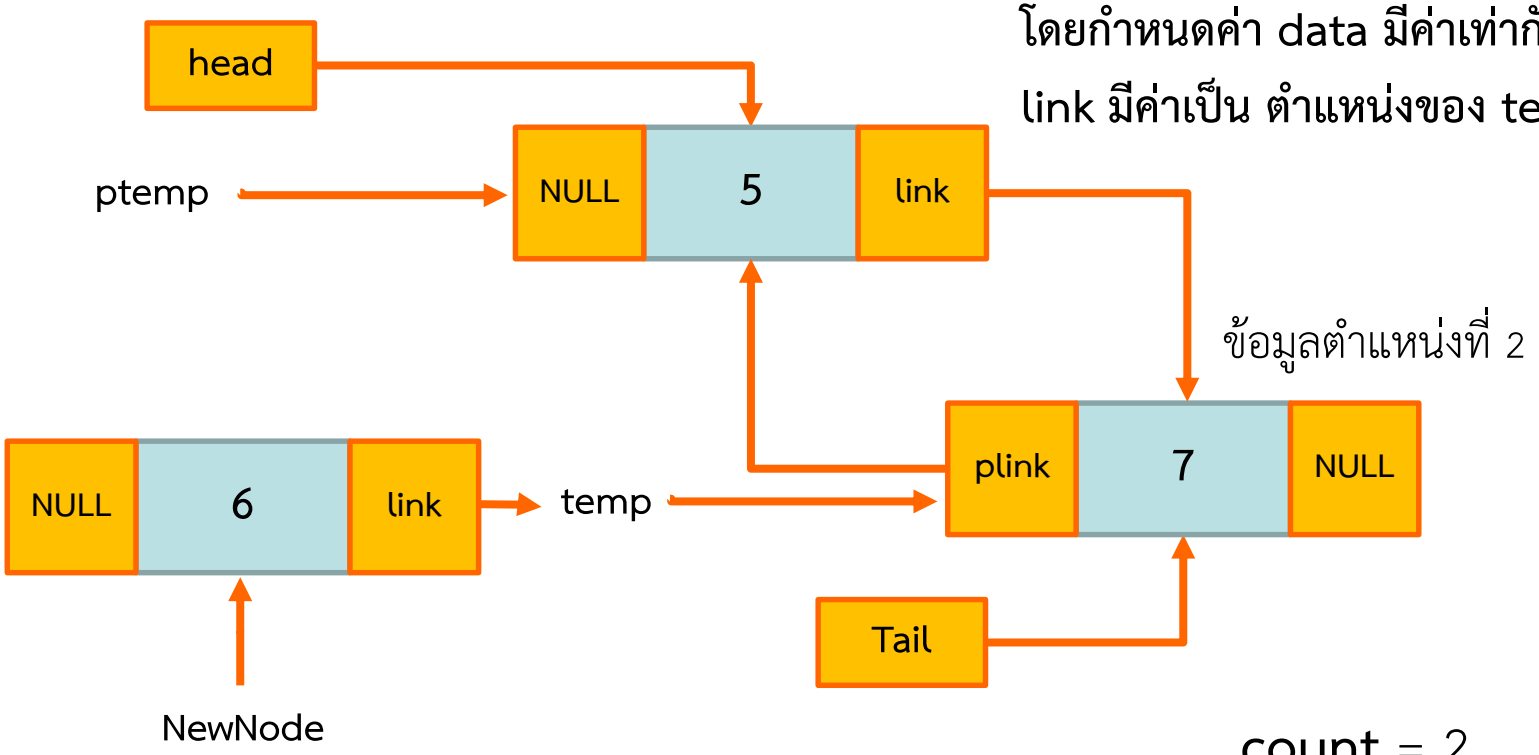


count = 2

แนวคิดการเพิ่มข้อมูล กรณีระบุตำแหน่ง [4]

obj_DBlinkedlist.add(2 , 6);

สร้าง new object จาก class node โดยกำหนดค่า data มีค่าเท่ากับ 6 และ link มีค่าเป็น ตำแหน่งของ temp

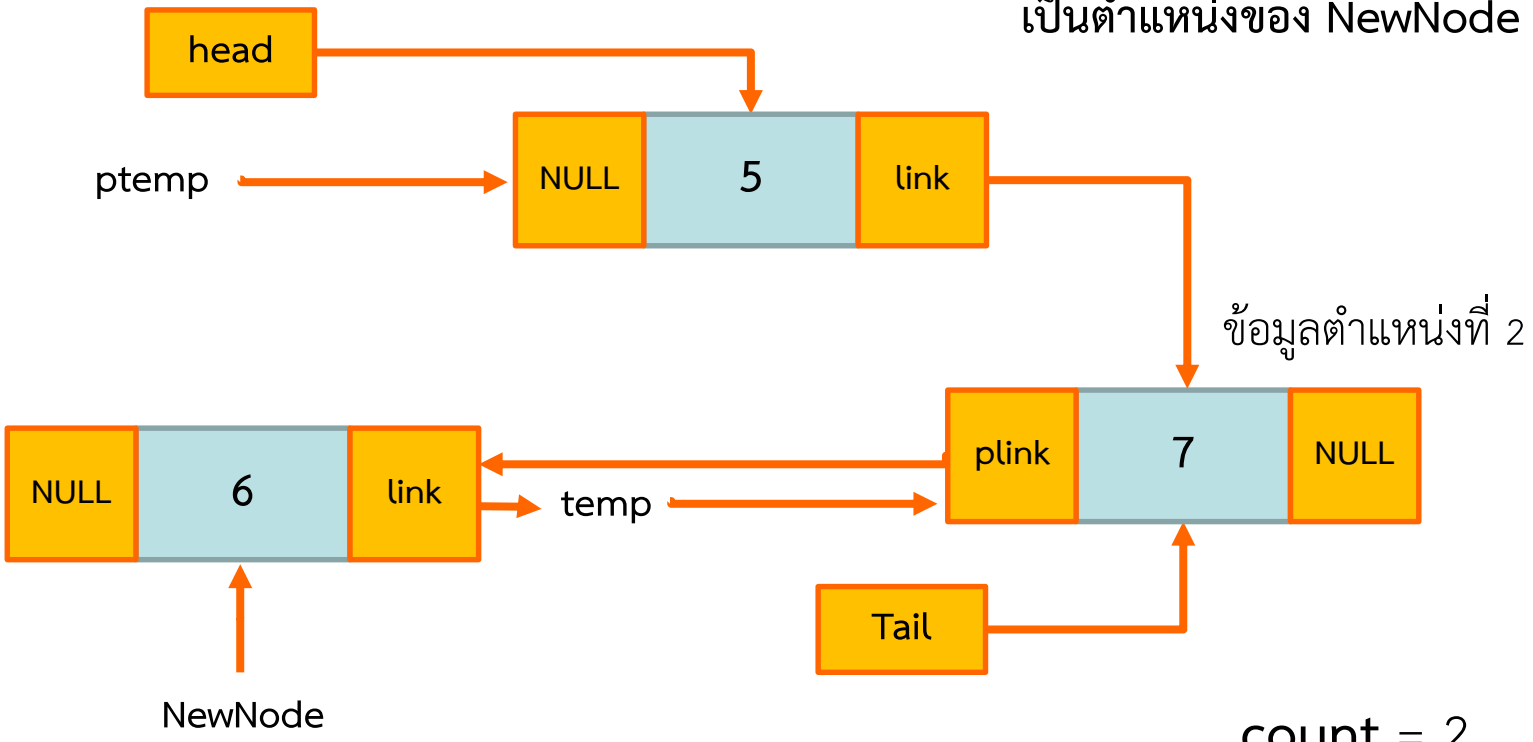


count = 2

แนวคิดการเพิ่มข้อมูล กรณีระบุตำแหน่ง [5]

obj_DBlinkedlist.add(2 , 6);

เปลี่ยน plink ของตัวที่ temp ชี้ ให้เป็นตำแหน่งของ NewNode

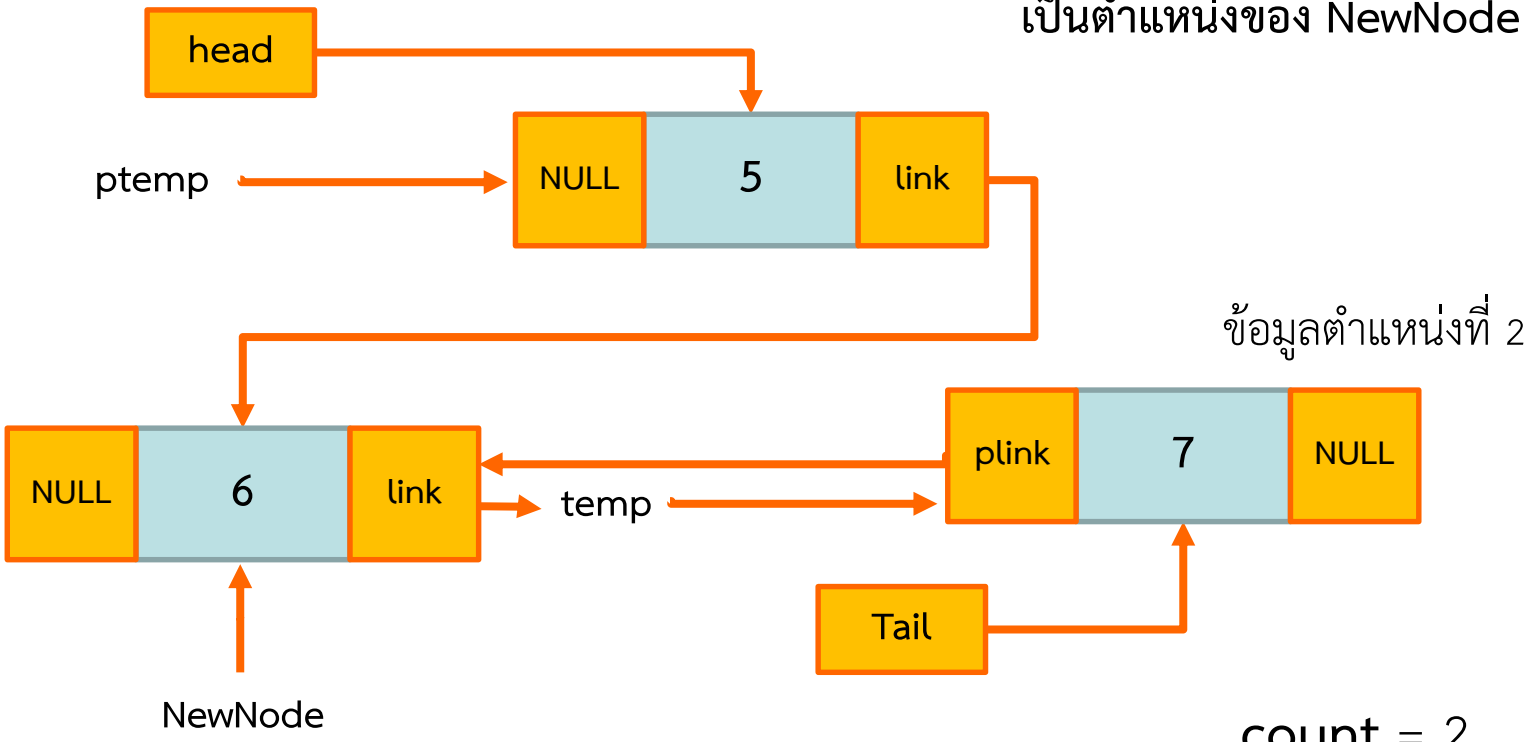


count = 2

แนวคิดการเพิ่มข้อมูล กรณีระบุตำแหน่ง [6]

obj_DBlinkedlist.add(2 , 6);

เปลี่ยน link ของตัวที่ ptemp ชี้ ให้เป็นตำแหน่งของ NewNode

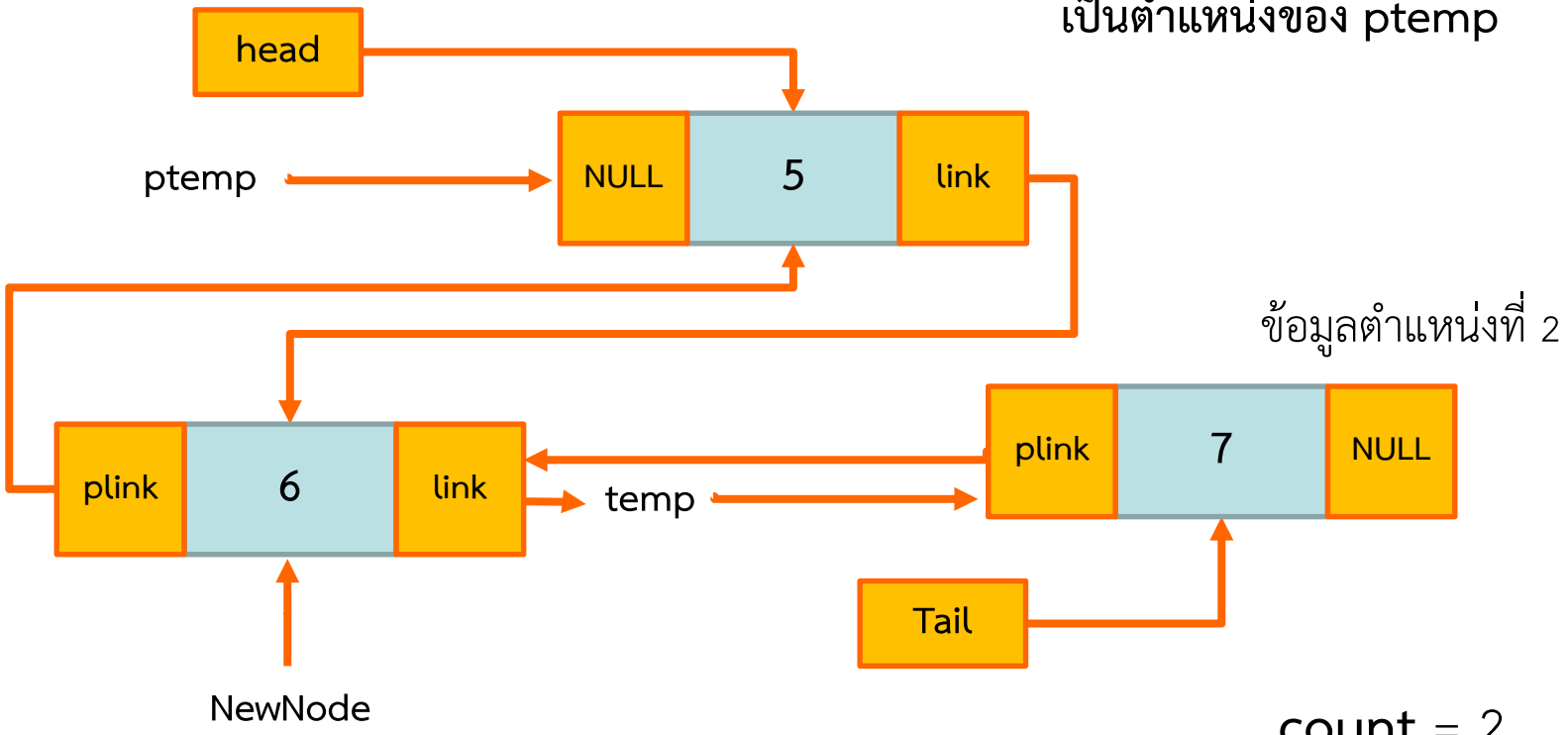


count = 2

แนวคิดการเพิ่มข้อมูล กรณีระบุตำแหน่ง [7]

obj_DBlinkedlist.add(2 , 6);

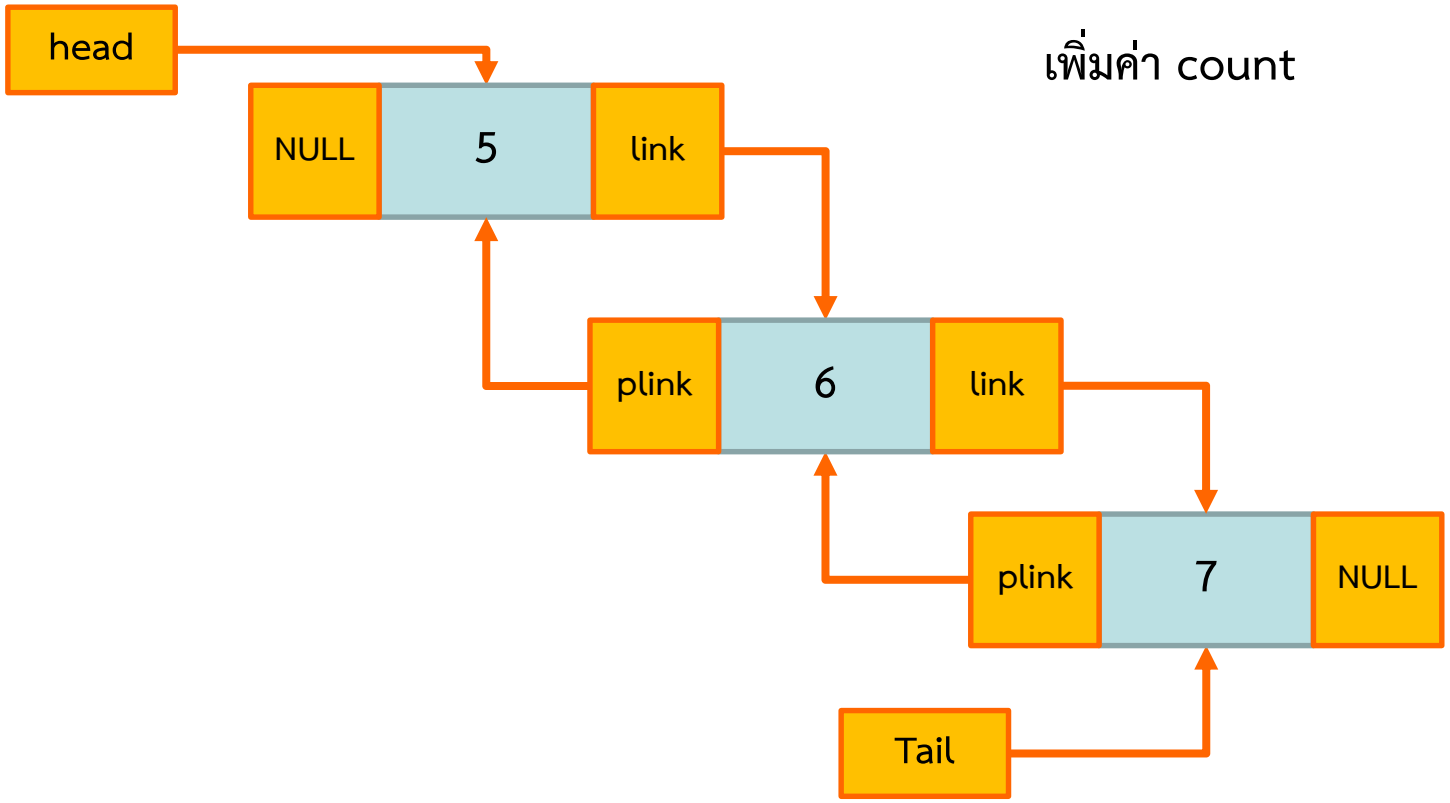
เปลี่ยน plink ของตัว NewNode ชี้ ให้เป็นตำแหน่งของ ptemp



count = 2

แนวคิดการเพิ่มข้อมูล กรณีระบุตำแหน่ง [8]

obj_DBlinkedlist.add(2 , 6);

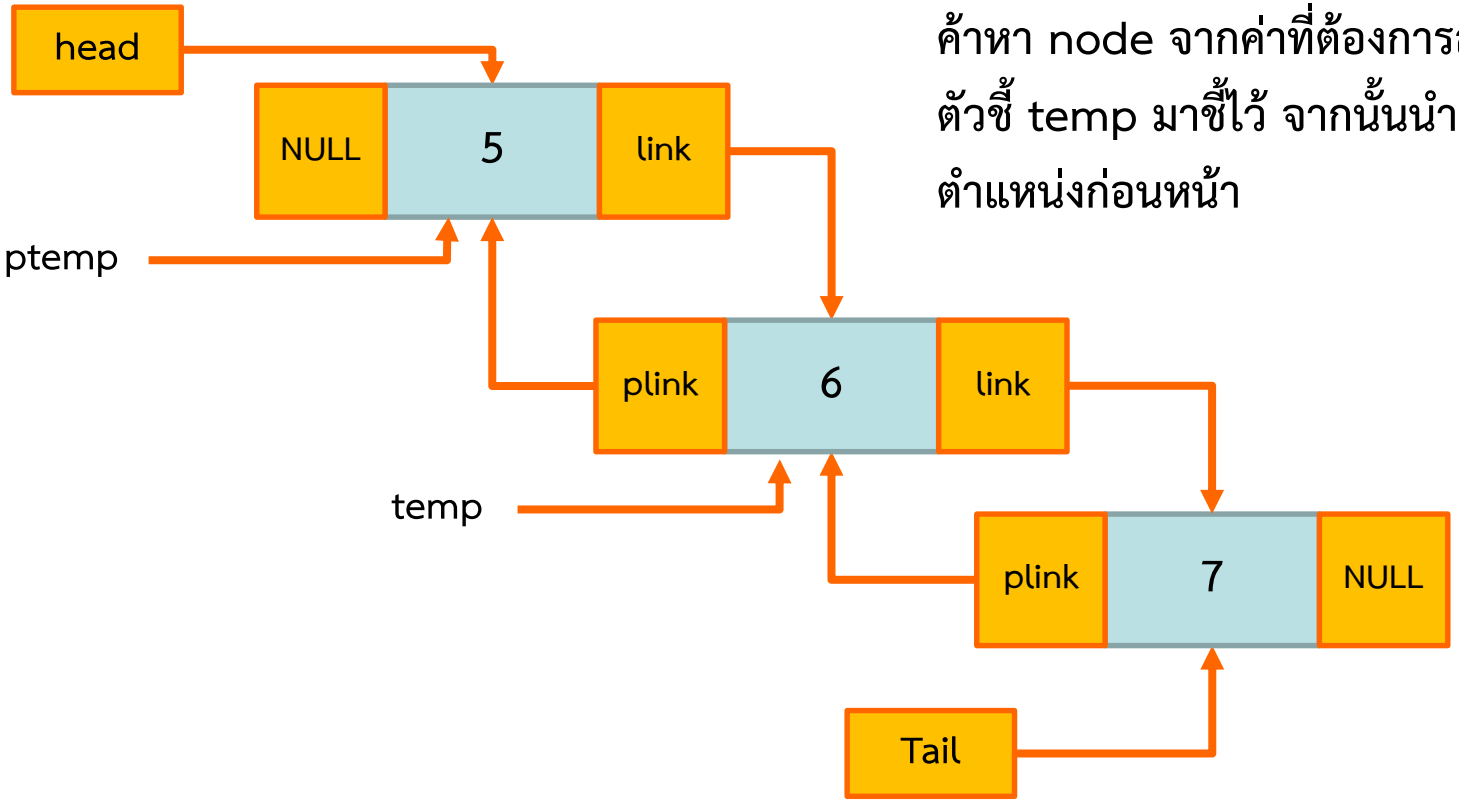


เพิ่มค่า count

count = 3

แนวคิดการลบข้อมูล

obj_DBlinkedlist.remove(6);

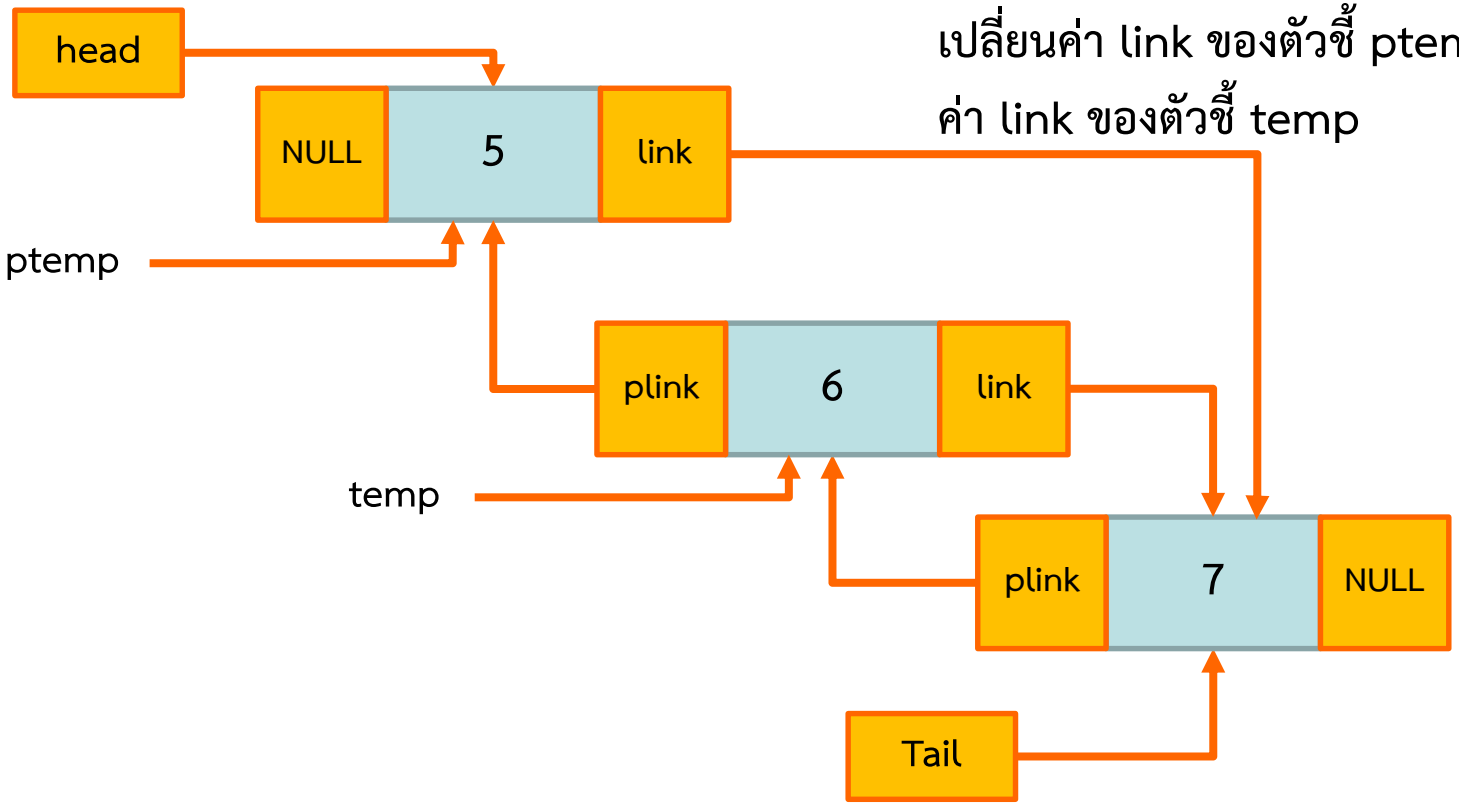


ค้าหา node จากค่าที่ต้องการลบ และสร้างตัวชี้ temp มาชี้ไว้ จากนั้นนำ ptemp ชี้ตำแหน่งก่อนหน้า

count = 3

แนวคิดการลบข้อมูล [2]

obj_DBlinkedlist.remove(6);

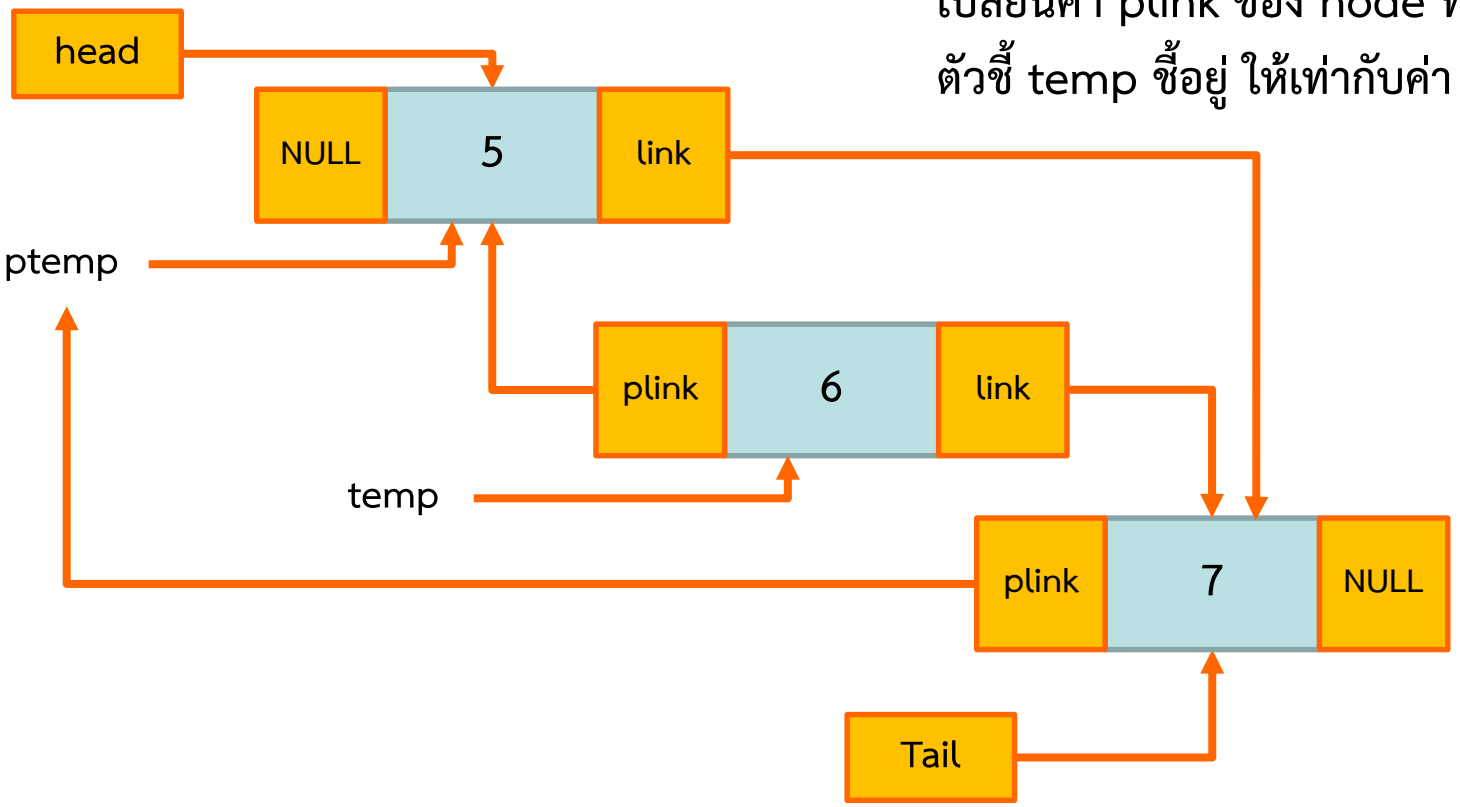


เปลี่ยนค่า link ของตัวชี้ ptemp ให้เท่ากับ
ค่า link ของตัวชี้ temp

count = 3

แนวคิดการลบข้อมูล [3]

obj_DBlinkedlist.remove(6);

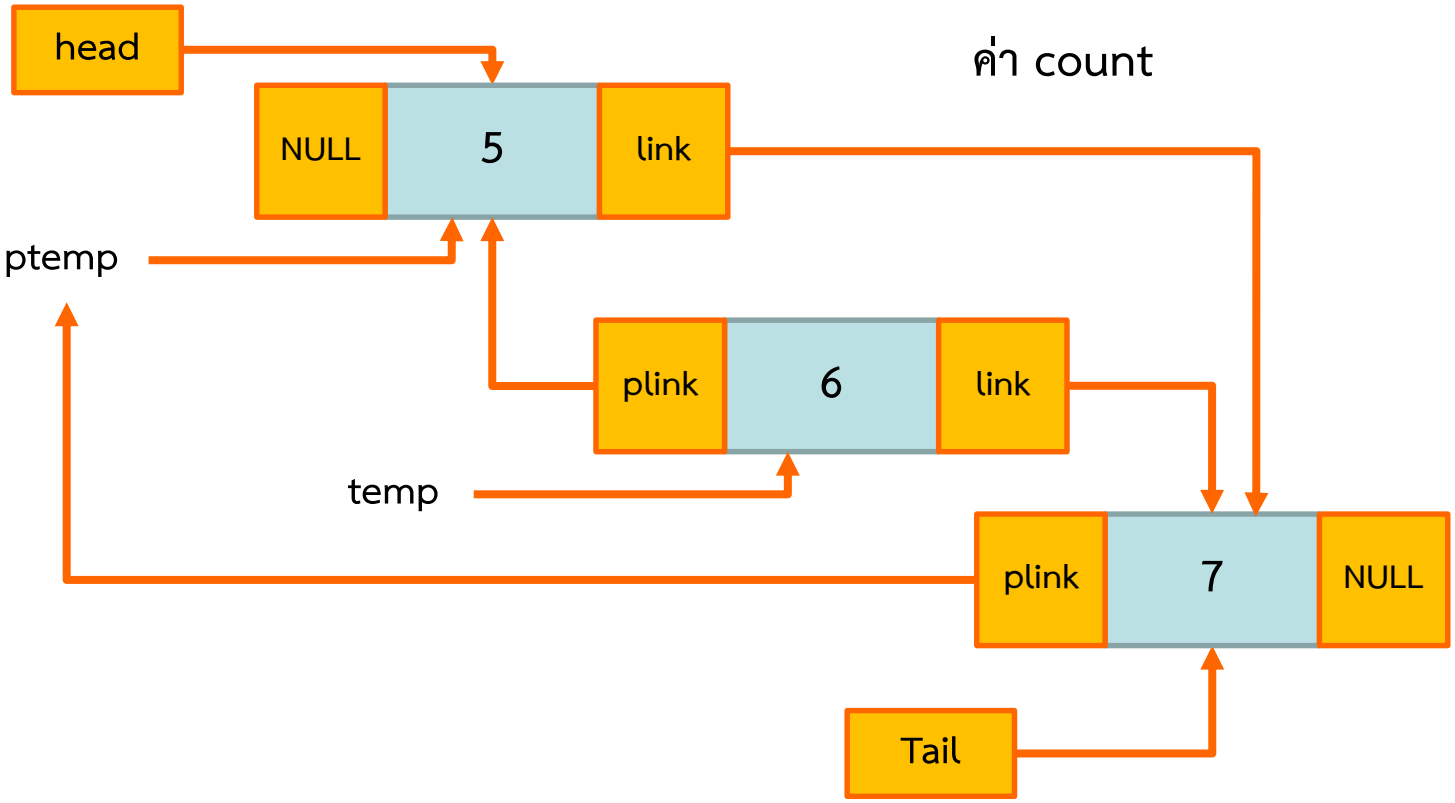


count = 3

แนวคิดการลบข้อมูล [4]

obj_DBlinkedlist.remove(6);

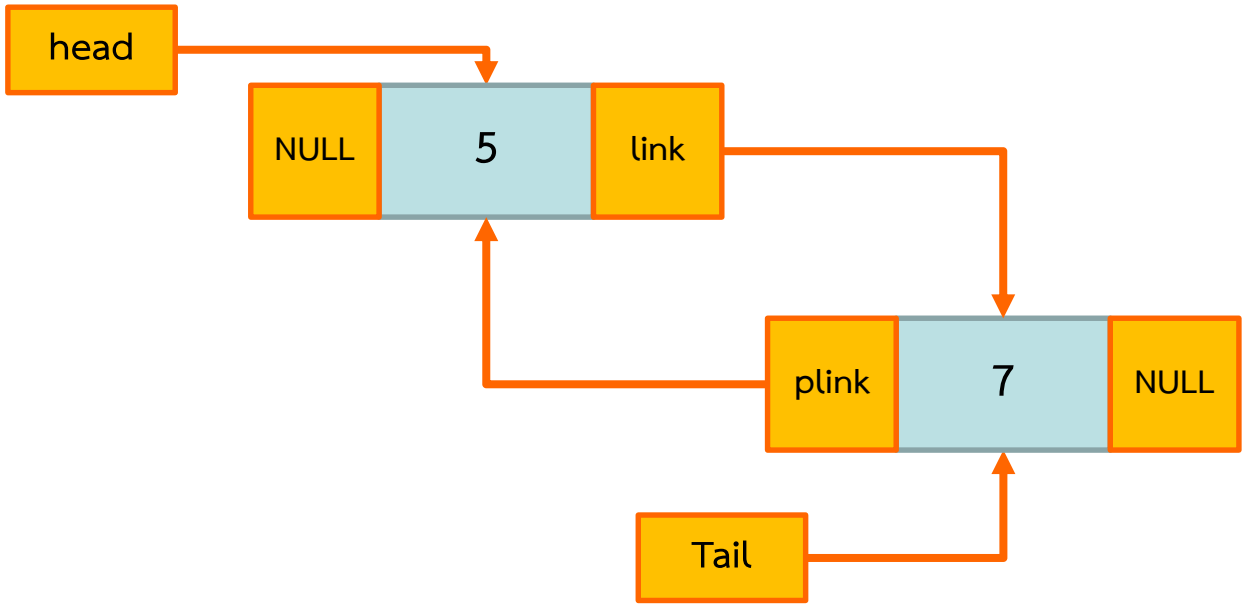
คืนหน่วยความจำโหนด temp และลดค่า count



count = 2

แนวคิดการลบข้อมูล [4]

obj_DBlinkedlist.remove(6);



count = 2

แบบฝึกหัด

1. สร้าง คลาส DbLinkedList เพื่อจัดเก็บข้อมูลเลขจำนวนเต็ม โดยมีความสามารถในการจัดการข้อมูล ดังนี้
    - สามารถเพิ่มข้อมูลแบบปกติและแบบระบุตำแหน่ง
    - สามารถเปลี่ยนแปลง หรือ แก้ไขข้อมูล
    - สามารถลบข้อมูล
    - สามารถค้นหาข้อมูล
    - สามารถแสดงข้อมูลทั้งหมด และแสดงข้อมูลทั้งหมดแบบย้อนกลับ
    - สามารถตรวจสอบจำนวนข้อมูลที่มีทั้งหมด
    - สามารถตรวจสอบพื้นที่ว่างในการเก็บข้อมูล
2. นำคลาสที่สร้างขึ้นไปทดสอบการใช้งานในฟังก์ชัน main โดยทำการสร้างเป็นลักษณะเมนูสำหรับทดลองทุกความสามารถที่มีในคลาส DbLinkedList ของตนเอง

Class ของแบบฝึกหัด