105學年 資料結構TA課

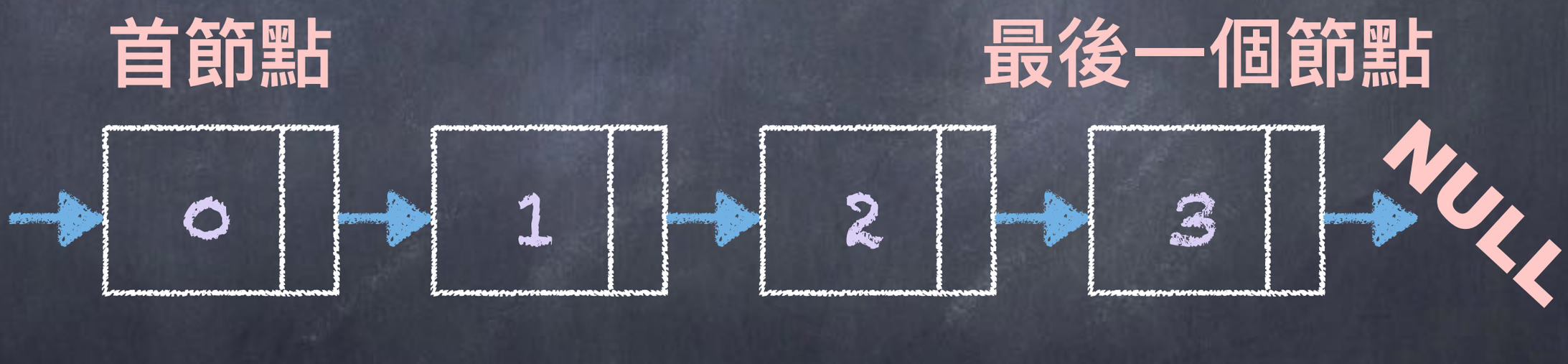
鏈結串列(linked lists)

Outline

- 鏈結串列？
- 鏈結串列與陣列的比較
- 新增資料
- 刪除資料
- 程式碼

鏈結串列？

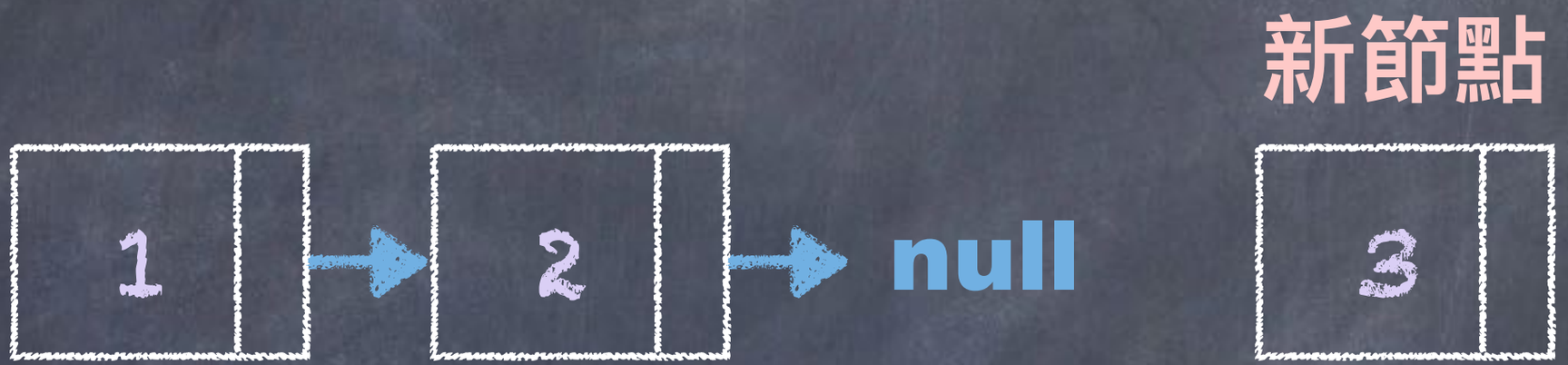
節點(node)

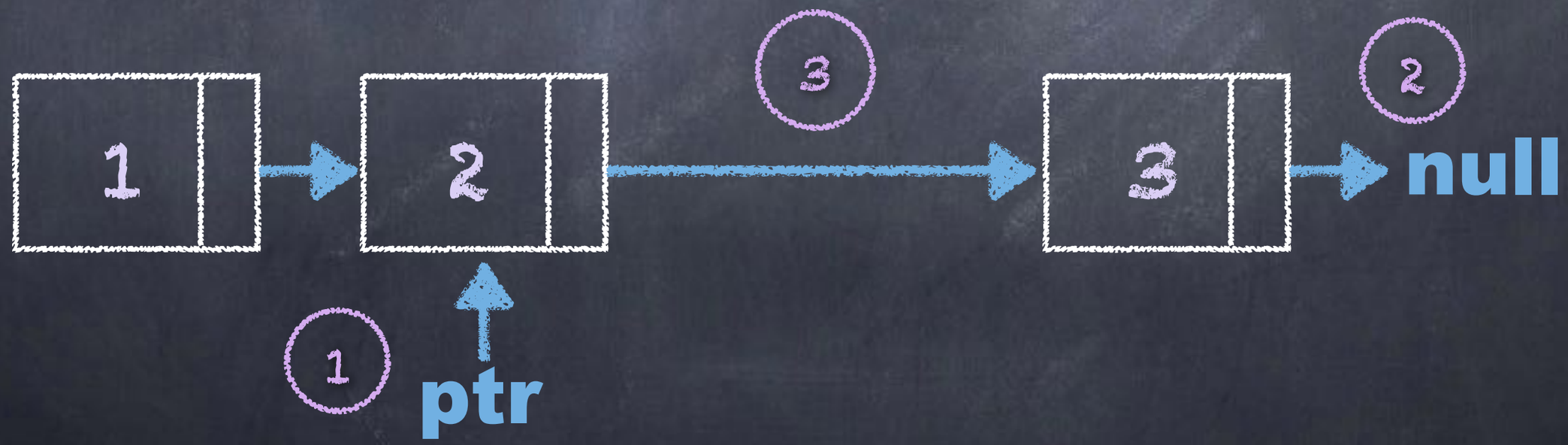


鏈結串列與陣列比較

	陣列	鏈結串列	
優點	- 循序讀取速度較快 - 沒有link空間需求 - 可靠度較高	- 循序讀取速度較慢(因為要先讀取指標才能讀取下一個node) - 需要額外的link空間 - 可靠度較低(因為連結斷掉資料就lost了)	缺點
缺點	- 佔用連續的記憶體空間 - 每個元素的資料型態皆一致 - 陣列大小要事先宣告，無法新增/刪除空間 - 元素搬移較費時，需O(n)	- 不一定要連續的記憶體空間 - 各個node間的資料型態可以不一樣 - 事前無需宣告所需大小，可任意新增/刪除node - 元素的新增/刪除較簡單，只需更改指標即可，僅花費O(1)	優點

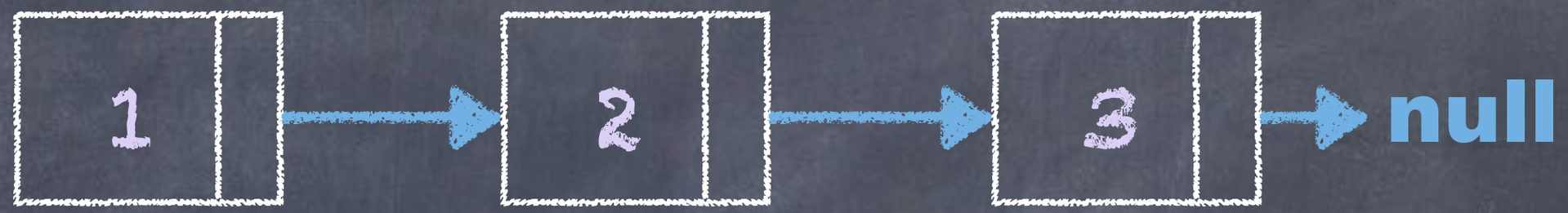
新增資料



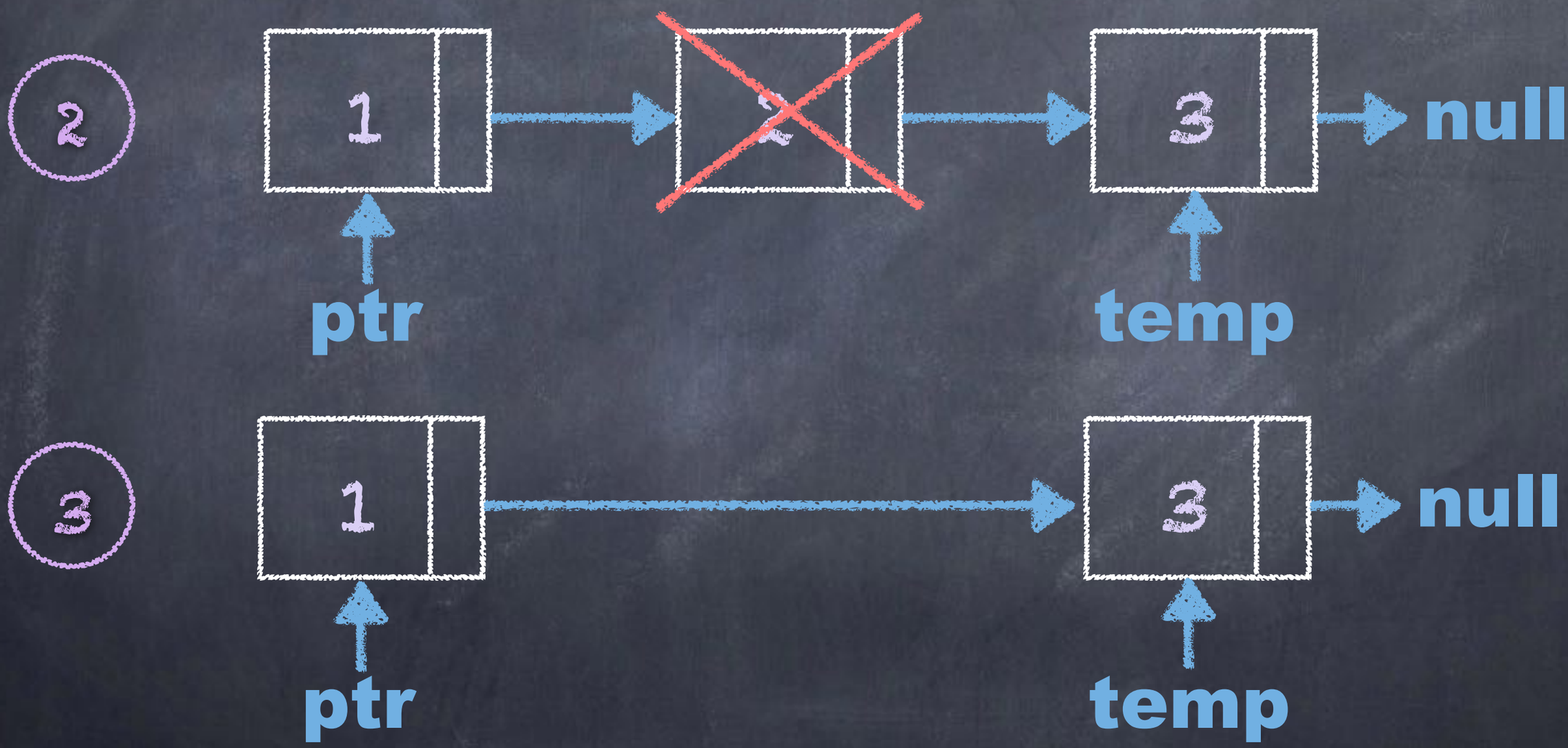


刪除資料

要刪除的節點



刪除資料



程式碼

```
struct NODE {
    int number; //該節點的元素
    NODE *link; //指向下一個node的指標
}
```

程式碼

```
class Linklist {
private:
    NODE *first; // 一定要有首指標
    NODE *tail;  // 最後一個節點
public:
    Linklist();
    void addNode(int n); // 新增節點
    void delNode(); // 刪除最後一個節點
    void list();    // 列出所有節點元素
};
```

程式碼

```
void Linklist::addNode(int n) {
    Node *new_num = new Node; // 新增節點
    new_num->number = n;
    new_num->next = NULL;

    if (first == NULL) {
        first = new_num; // 若鏈結串列沒有節點
        tail = new_num;
    } else {
        tail->next = new_num;
        tail = new_num;
    }
}
```

程式碼

delNode(); 和 list(); 交給你們了...

練習時間

練習時間

- 使用鏈結串列完成...
  - 基本鏈結串列的新增/刪除節點
  - 堆疊和佇列
  - 將上次作業使用鏈結串列實作

歐～對了，明天考英文