linked list

鏈結串列

問題

我們先來思考一個常見的問題：

我想要做一個彈幕遊戲，自然需要一個陣列來管理目前出現在螢幕上的所有子彈，這樣我才能夠讓子彈移動或是判斷有沒有打到東西。但子彈的數量是會隨著時間變化的，我也會需要可以隨時插入或是刪除子彈。那麼我的陣列到底應該要開多大呢？

1. 取平均值？

   陣列大小設成子彈的平均數量感覺很合理，但是一旦子彈數量超過平均值，程式就會當掉！

2. 取最大值？

   設成最大值就不用擔心程式當掉的問題了！但假設最多同時會有1000顆子彈出現在螢幕上，但平均卻只有5顆，這樣在大多數的時候就浪費了995個空格！

有沒有一個辦法能夠既不浪費空間，又不怕宣告的不夠大導致當機呢？

顯然固定長度的陣列不符合我們的需求。

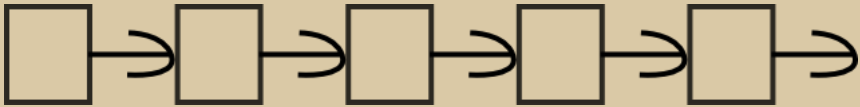
有沒有辦法在要用的時候再分配空間就好了呢？

鏈結串列 (linked list)

我們可以把陣列想像成是一排硬梆梆的書櫃。

要對增加或縮小他的容量都是一件不容易的事情，更別說是插入或刪除了！

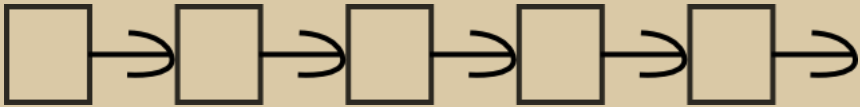
但如果我們能夠像鎖鏈一樣讓所有格子一環扣一環的呢？



如此一來，插入只要先斷開鎖鏈！再接回去就好了，感覺比使用陣列來得簡單多了！

鏈結串列 (linked list)

我們就把這樣的概念叫作「鏈結串列」。

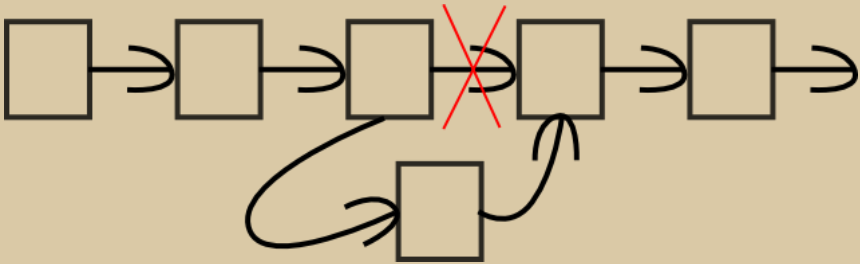


我們希望它可以做到下面幾件事情：

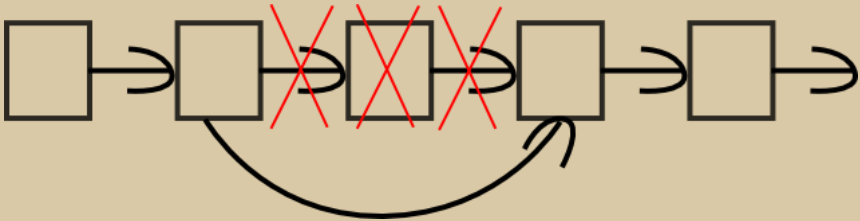
1. 快速在中間插入一個元素。
1. 快速刪除一個元素。
1. 有幾個元素就用幾格，不會浪費空間。

鏈結串列 (linked list)

插入



刪除



linked list - 實作

介紹完了基本概念，接下來讓我們想想該如何把它寫成程式吧！

因為除了資料，我們可能還需要維護其他東西，所以需要struct幫我們打包好！

```
struct node {
    //...
};
```

先來想想我們需要什麼東西：

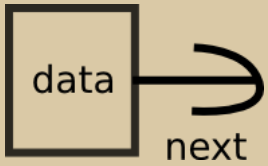
1. 需要能夠放資料
2. 需要能夠接到下一格
   - 是要怎麼跟下一格接在一起呢？
   - 如果能夠知道下一格的記憶體位置？
   - 我們可以用指標來做到這件事！！

linked list - 實作

經過剛才的分析，我們就能夠把我們需要的資料類型定義出來了！

```
struct node {
    int data; //假設要放的資料是整數
    node *next; //一個指標指向下一格
};
```

如此一來我們就有了一個類似這樣的東西：



linked list - 實作

我們有了一個 node，接著要如何把他們串在一起，形成一條 linked list 呢？先來想想一條 linked list 需要哪些重要的東西：

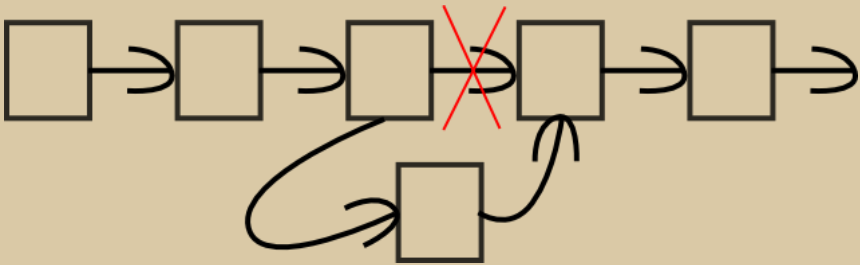
1. 開頭 - 有了開頭就有辦法沿著 next 走完整條 linked list
1. 結尾 - 要有個結尾才有走完的一天

於是我們用開頭來代表一個 linked list, 而 next == NULL 的 node 則代表 linked list 的結尾。這麼一來，只要知道了開頭，我們就有辦法走完整條 liked list ：

```
//假設開頭叫作 head
node *ptr = head;
while(ptr != NULL) {
    //對目前的 node 做處理
    ptr = ptr->next; //往前走一步
}
```

linked list - 實作

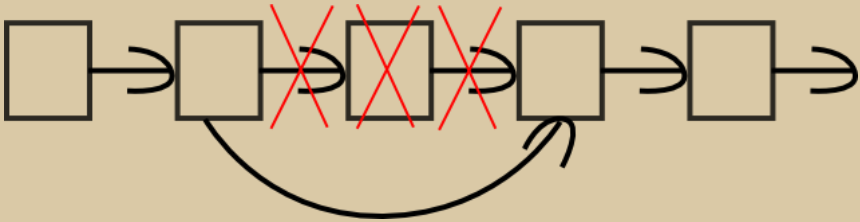
接著讓我們把插入/刪除的動作都寫成 code 吧！



```
int insert(node *ptr, int data) {
    node *now = new node;
    now->data = data;
    now->next = ptr->next; //先接
    ptr->next = now; //再斷
}
```

linked list - 實作

刪除



```
int remove(node *prev) {
    node *tmp = prev->next; //要刪的 node
    prev->next = tmp->next; //先接
    delete tmp; //再斷
}
```

linked list - 實作

刪除-為什麼要傳被刪除節點的前一個

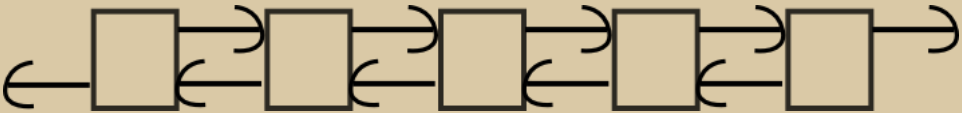
- 可不可以這樣

```
int remove(node* to_remove){
    //...
}
```

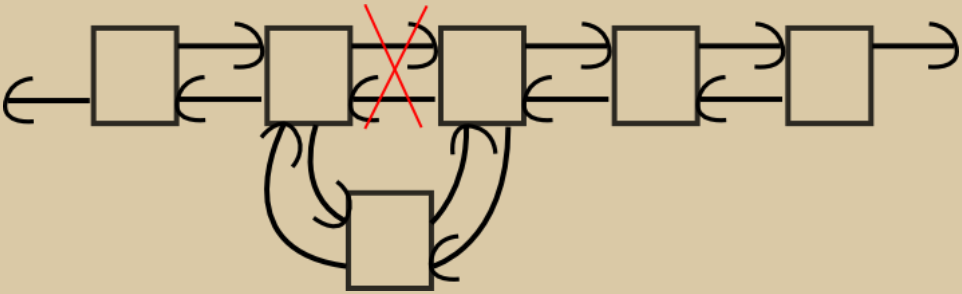
- 沒辦法，因為在刪除當前節點時，我們需要修改前一個節點的next指標。
- 所以我們可不可以多紀錄一些東西？

```
struct node{
    node* prev;
    node* next;
    // other...
};
```

doubly linked list



```
struct node {
    int data; //假設要放的資料是整數
    node *prev; //一個指標指向上一格
    node *next; //一個指標指向下一格
};
```



練習

singly linked list

作業

doubly linked list 的 `push_front` , `pop_front` , `push_back`, `pop_back`就當作作業吧！