

102逢甲資訊系程

Chap.11 鏈結串列



鏈結串列的基本介紹

為什麼需要鏈結串列？

- 如果要寫一個通訊錄程式，可提供使用者新增、修改、刪除好友，且好友須照姓氏排列，要怎麼做？

使用陣列？

- 宣告 `char name[9999][10]`

- 人數少，造成空間浪費
- 人數多，陣列無法儲存

- ✿ • 新增好友後，再進行姓氏排列

- 將與所有資料逐一比對，效率降低

- 刪除好友後，將之後的資料往前搬

- ✿ – 不斷搬動資料，效率降低

使用鏈結串列

- 新增資料時，才向系統要空間
- 新增好友，可直接搜尋同姓氏的位置插入資料
- 刪除好友後，可將空間釋放掉



單向鏈結串列：基本介紹

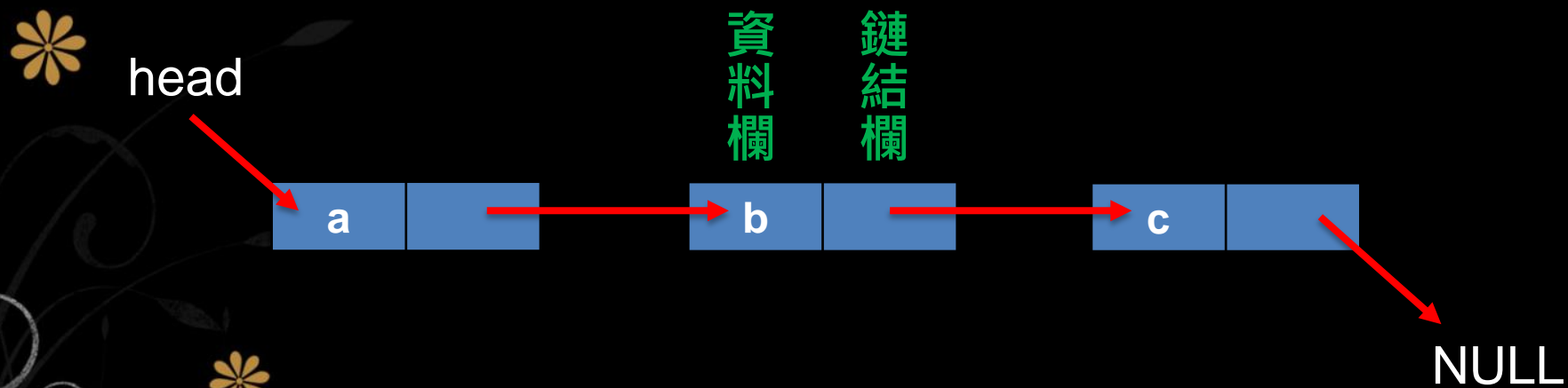
鏈結串列的架構

- 鏈結串列的名稱
 - 指向第一個節點的指標名稱 (即 head)

-  • 節點
 - 至少包含一個資料欄及一個位址欄

鏈結串列

- 示意圖



鏈結串列

- 由節點(node)構成，每個節點的資料結構至少有兩項，分別是資料(data)和鏈結(next)
- 每個節點除了紀錄資料外，還記錄下一個節點的位址，每個節點以此方式串連起來，形成鏈結串列

步驟一：定義節點結構

- 宣告

```
struct node {  
    char data ;  
    struct node *next ;  
};
```

- 節點示意圖



步驟二：建立串列頭

- 宣告

`struct node *head;` //指向node型態的指標

- 並設定初值為 **NULL**

`head = NULL;` //將head指向空指標

- 示意圖

head  NULL



單向鏈結串列：新增節點

新增節點

//新增資料

```
node* Add(node* head, char newData){
    node *newNode;
    newNode = (node*)malloc(sizeof(node));
    if(newNode != NULL){
        newNode->next = NULL;
        newNode->data = newData;
        if(head == NULL){
            head = newNode;
        }
        else{
            newNode->next = head;
            head = newNode;
        }
        return head;
    }
    else{
        printf("記憶體配置失敗...\n");
        system("pause");
    }
}
```

malloc()動態記憶體配置


- 函式庫

#include<stdlib.h>

- 語法

 void* malloc (size_t size) ;

- 說明

 配置一個指定大小的空間，並傳回該空間的位址

malloc()

- 範例

```
int *ptr = ( int* )malloc( sizeof( int ) * 2 );  
//配置兩個整數空間，並將ptr指向那個空間
```

- malloc()會回傳一個 void* 型態，故須使用 (int*) 來強制轉換
- sizeof()用來計算資料所占位元組，
sizeof(int) 即回傳無號正整數 4

新增節點

- 副函式宣告

node* Add(node* head, char newData)

- 傳入值(鏈結頭, 要新增的資料)
- 回傳值(鏈結頭)

步驟一：新增一個節點

- 宣告

```
node* newNode ;
```

- 配置空間

```
newNode = (node*)malloc( sizeof(node) );
```

- 檢查記憶體是否配置成功

```
if( newNode != NULL )
```

步驟二：存入資料

- 存入資料

`newNode -> data = newData ;`

`newNode -> next = NULL ;`

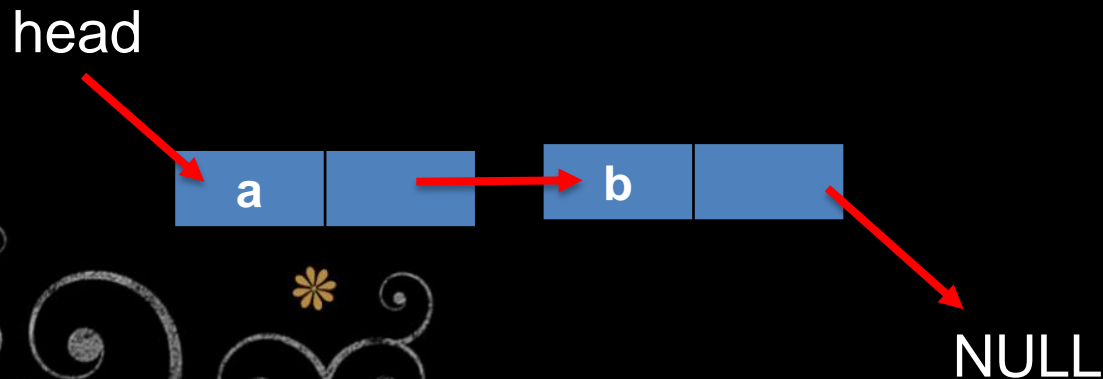


步驟三：鏈結節點

- 考慮兩種情況
 - 串列尚未有節點

head → NULL

- 串列已有節點



步驟三：鏈結節點

- 串列尚未有節點

```
if(head == NULL){  
    head = newNode;  
}
```

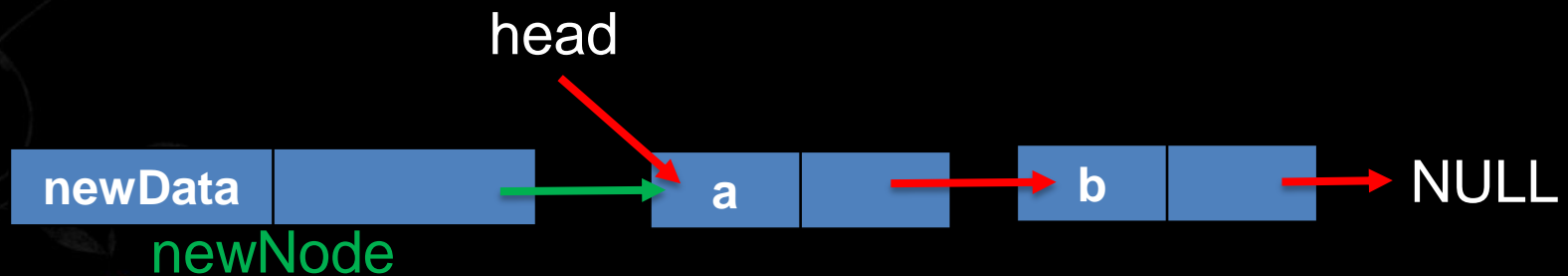


步驟三：鏈結節點

- 串列已有節點

else{

newNode->next = head;



步驟三：鏈結節點

- 串列已有節點(續)

```
head = newNode;
```

```
}
```

head





單向鏈結串列：印出串列

印出串列

//顯示資料

```
void Show(node* ptr) {  
    printf("Head -> ");  
    while(ptr != NULL) {  
        printf("%c -> ", ptr->data);  
        ptr = ptr->next;  
    }  
    printf("NULL\n");  
}
```


印出串列

- 副函式宣告

`void Show(node* ptr)`

- 傳入值(鏈結頭)

印出串列(續)

- 條件

`while(ptr != NULL)`



- 印出資料

`printf(“ %c”, ptr->data);`

- 移動指標

`ptr = ptr->next ;`





單向鏈結串列：插入節點

步驟一：新增一個節點



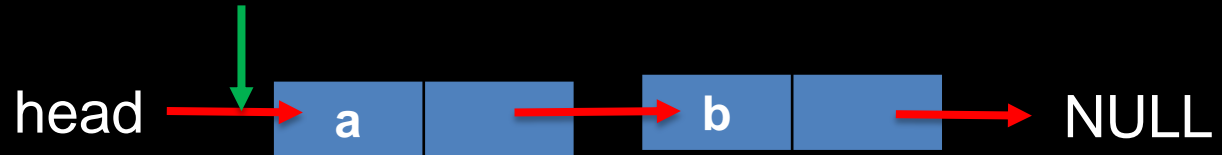
The diagram shows a new node structure. It consists of a light blue rectangular box divided into two parts. The left part contains the text 'newData'. The right part is empty. A red arrow points from the right side of this box to the text 'NULL'.

newData

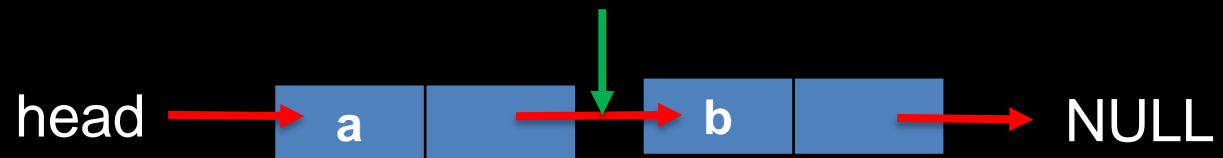
NULL

步驟二：鏈結節點

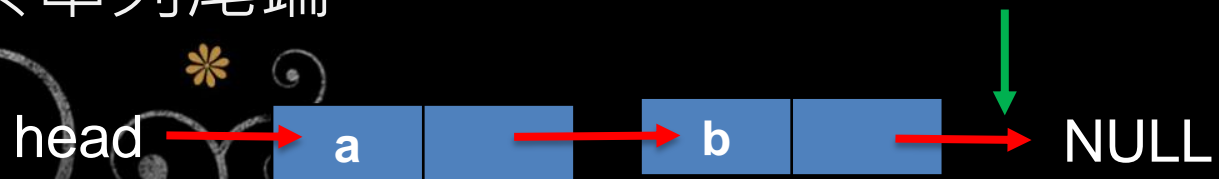
- 考慮三種情況
 - 插入串列前端



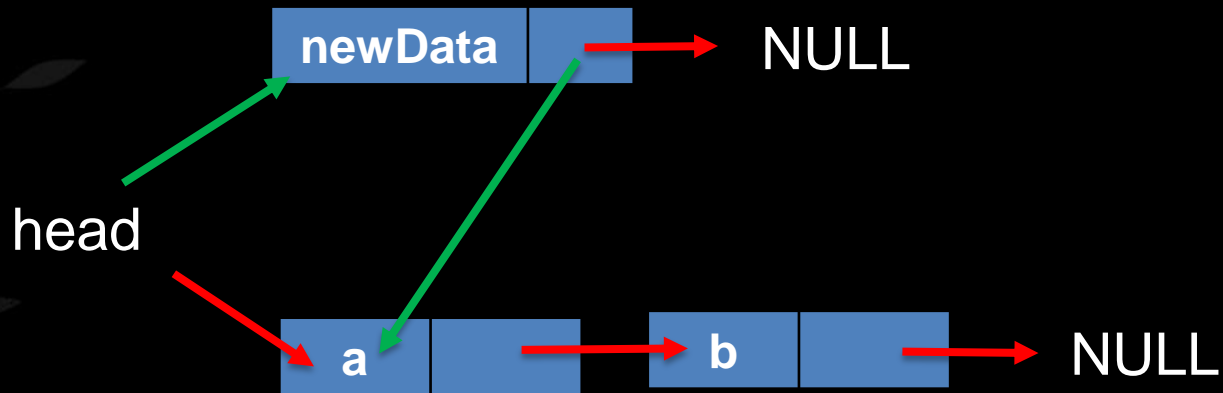
- 插入於兩節點間



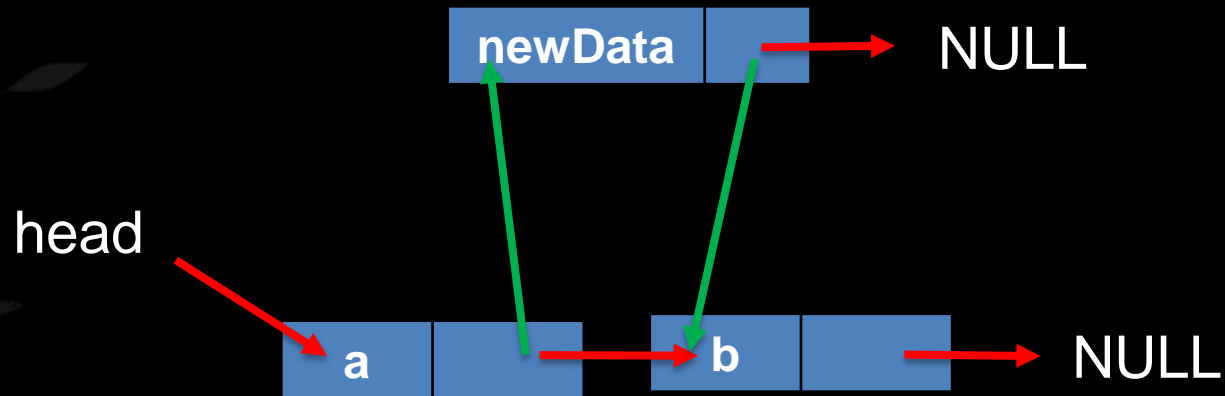
- 插入於串列尾端



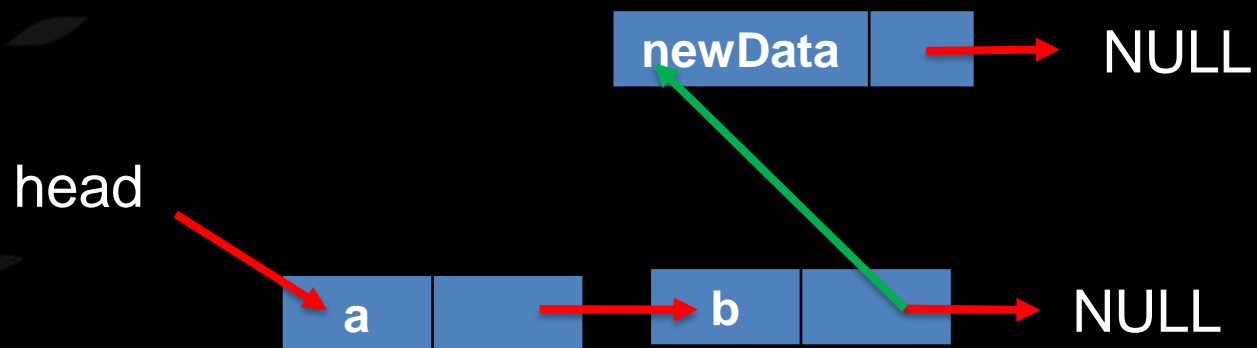
插入串列前端



插入於兩節點間



插入於串列尾端





單向鏈結串列：刪除節點

free()

- 說明

- 釋放原先所建立的記憶體空間
- 須注意並非刪除原先記憶體空間的資料，也非將操作的指標改為指向 **NULL**，而是告訴作業系統，程式不會再去利用這塊記憶體空間。

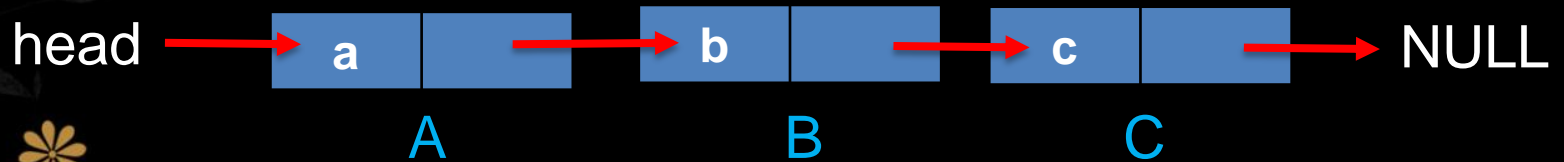
- 範例

 `free(head);`

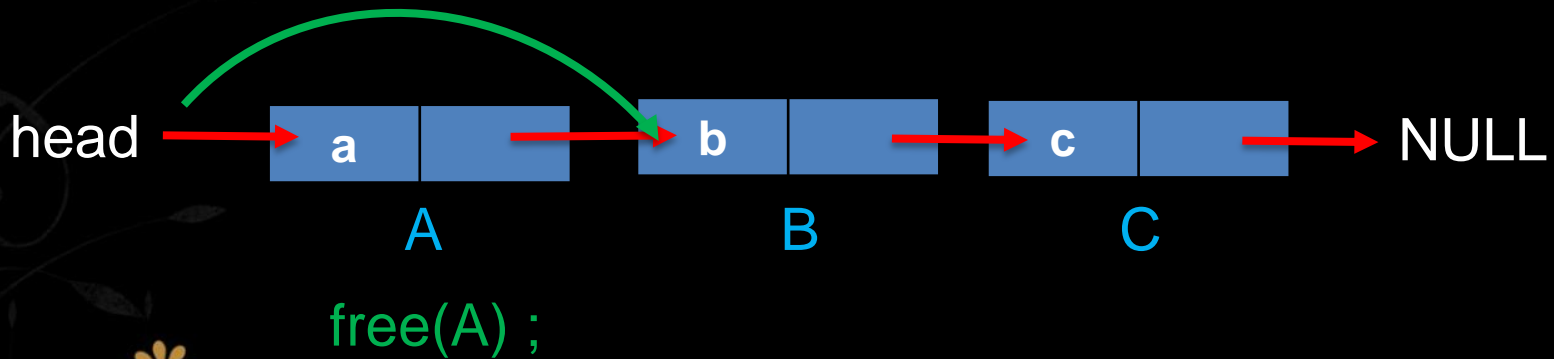
步驟一：斷開鏈結

- 考慮二種情況
 - 刪除第一個節點(A)

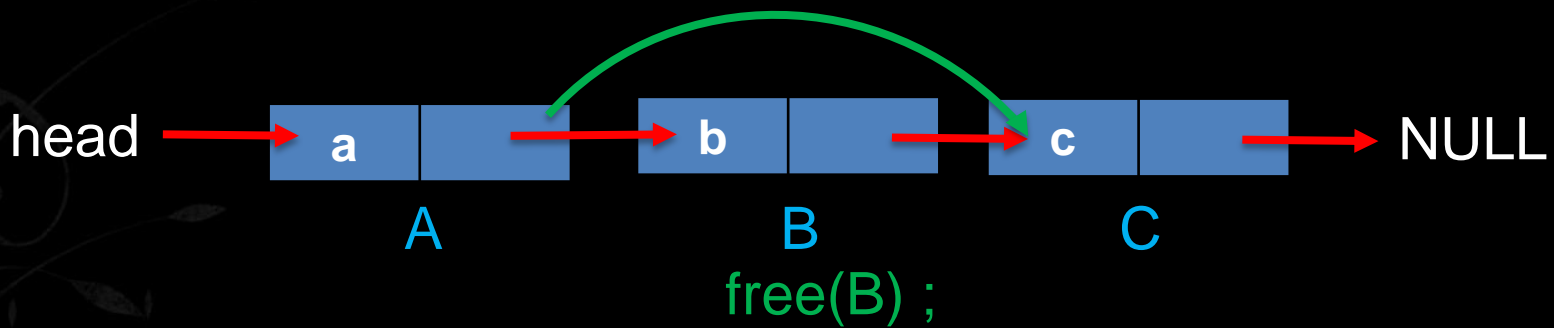
— 刪除兩點間的節點(B) & 刪除尾節點(C)



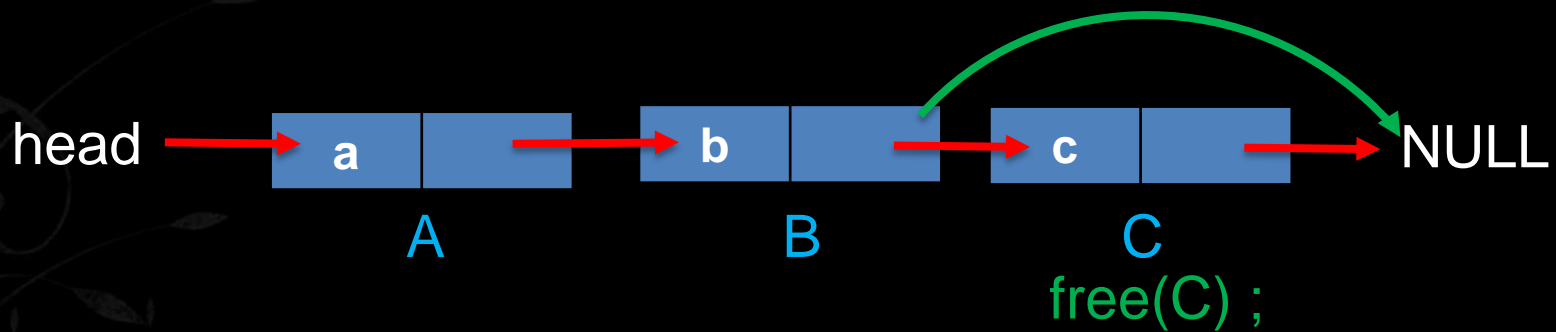
刪除第一個節點



刪除兩點間的節點



刪除尾節點



練習題

- 請實作“插入節點”的功能
- 請實作“刪除節點”的功能

進階題

請設計一程式，由使用者輸入 **2 個一元多次方程式**，並分別建構成兩個 Linked List。之後，設計一功能界面，供使用者選擇執行兩多項式之相加或相乘計算，並印出運算結果。

註：1) 多項式輸入為連續輸入下列格式：

係數 次方

當輸入 **係數為0** 時表示多項式結束。

若輸入 **多個相同次方** 的係數時，則將這些係數相加成為該次方的係數。

例如 輸入 -3 1 3 3 1 1 1 0 0

表示多項式 $3X^3 - 2X + 1$ 即 $3X^3 + (-3+1)X + 1$

- 2) 多項式允許 **可不按次方大小順序** 輸入，但在多項式的 Linked List 中必須 **按照次方大小順序** 排列，也必須按次方大小順序印出。請自行撰寫程式處理。
- 3) 相加或相乘後的結果係數為零時，不可將此次方印出 (即必須從多項式 Linked List 中移除)。列印時使用下列格式印出，如上例： $3X^3 - 2X + 1$ 則表示 $3X^3 - 2X + 1$ 。

網站推薦

程式解題網

- 高中生程式解題系統
- Lucky貓的 UVA (ACM) 園地

程式教學網

- 程式語言教學誌

- 良葛格學習筆記

比賽證照資訊

- 大學程式能力檢定(CPE)



謝謝聽講