

第十六章 串列 – **Linked List**

第十六章 大綱

▶ 16-0

- ▶ 陣列 vs. 鏈結串列

▶ 16-1

- ▶ 單向鏈結串列
(Singly Linked Lists)
- ▶ 練習題：建立一個單向鏈結串列

▶ 16-2

- ▶ 在單向鏈結串列中插入節點
- ▶ 練習題：在一個單向鏈結串列中插入節點

▶ 16-3

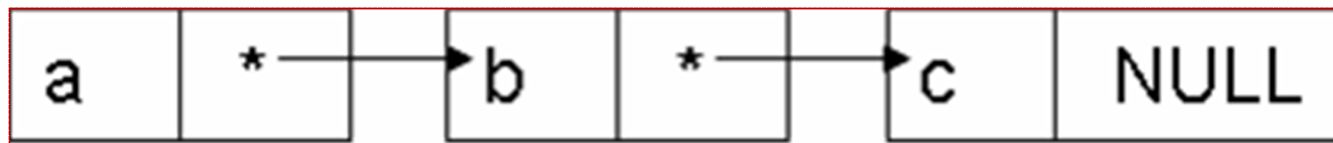
- ▶ 在單向鏈結串列中移除節點

16-0 陣列 vs. 鏈結串列

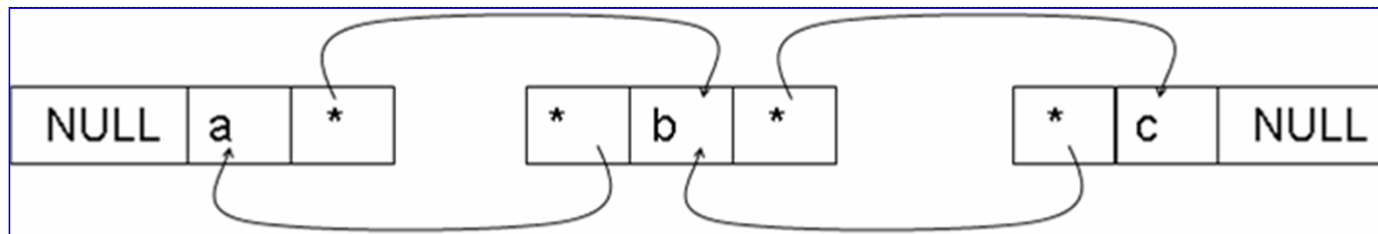
- ▶ 兩者共通之處：均為相同型態的資料所形成的集合
- ▶ 用陣列儲存資料的限制：
 - ▶ 在程式設計階段就要決定陣列的大小
 - ▶ 資料的插入或刪除往往有時會涉及大量陣列元素的搬動，
例：用陣列儲存一般文章時，假設文章有10000個字，若在開頭要插入或刪除一個字，均需移動10000個陣列元素
- ▶ 使用鏈結串列來儲存資料可克服以上兩個限制

16-1 單向鏈結串列

- ▶ 鏈結串列，依前後節點間的連結方式可分為以下兩種：
 - ▶ **單向鏈結串列**，每個節點均含以下兩部份：
 - ▶ Part 1：用於儲存資料的欄位（可以有多个欄位）
 - ▶ Part 2：指向下一節點的指標
 - ▶ **雙向鏈結串列**，每個節點均含以下三部份：
 - ▶ 用於儲存資料的欄位（可以有多个欄位）
 - ▶ 用於指向前方節點的指標
 - ▶ 用於指向後方節點的指標



單向鏈結串列



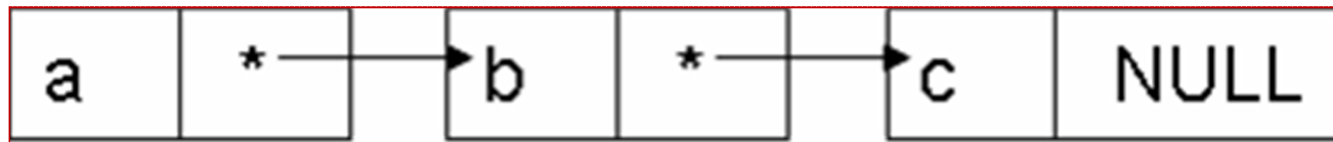
雙向鏈結串列

16-1 單向鏈結串列 – 實作方式

- ▶ 以C實作鏈結串列時，鏈結串列中的**每一個節點都是一個結構**。以下列鏈結串列為例，每一個節點可以下列結構來表示：

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

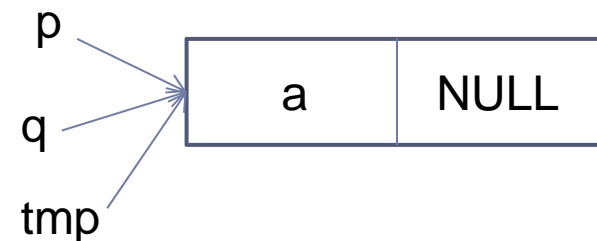
其中next欄位用於儲存下一個節點的位址，亦即指到下一個節點。



16-1 建立單向鏈結串列 (1/4)

```
1  /* SinglentlyLinkedList_1.cpp */
2  #include <stdio.h>
3  #include <stdlib.h>
4  struct node {
5      char nodeName;
6      struct node *next;
7  };
8  typedef struct node NODE;
9  int main() {
10     //p指向鏈結串列的開頭
11     //q指向鏈結串列的結尾
12     //tmp用於指向新產生的節點
13     NODE *p, *q, *tmp;
14     /* 建立第一個節點 */
15     tmp = (NODE *)malloc(sizeof(NODE));
16     tmp->nodeName='a';
17     tmp->next=NULL;
18     p=tmp;
19     q=tmp;
20 }
```

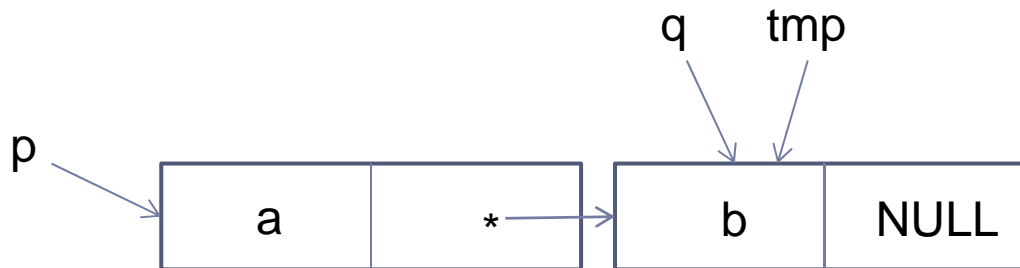
- ▶ Line 15: 配置節點所需的記憶體空間
- ▶ Line 16~17: 指定這個節點的資料內容
- ▶ Line 18: 讓p指向鏈結串列的第一個節點
- ▶ Line 19: 讓q指向鏈結串列的最後一個節點



16-1 建立單向鏈結串列 (2/4)

```
21  /* 建立第二個節點 */
22  tmp=(NODE *)malloc(sizeof(NODE));
23  tmp->nodeName='b';
24  tmp->next=NULL;
25  q->next=tmp;
26  q=tmp;
```

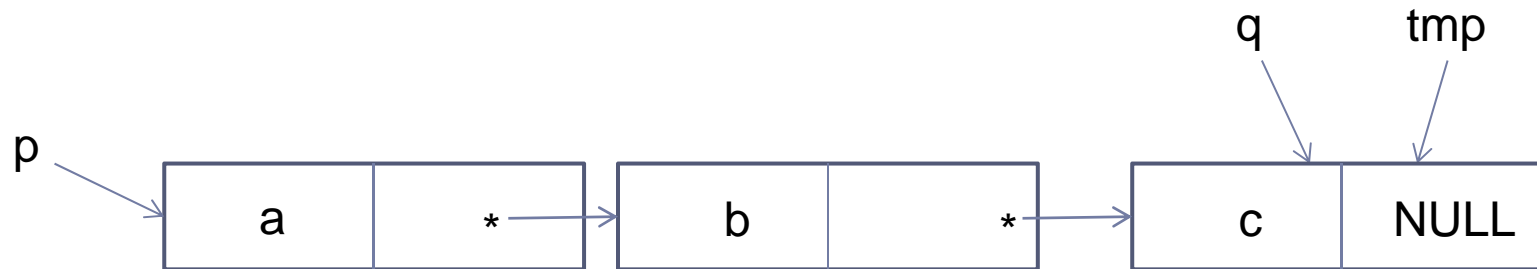
- ▶ Line 22~24: 配置第二個節點的記憶體空間及指定節點的欄位內容
- ▶ Line 25: 把新節點加到串列的結尾
- ▶ Line 26: 讓 q 指向最後一個節點



16-1 建立單向鏈結串列 (3/4)

```
28  /* 建立第三個節點 */
29  tmp=(NODE *)malloc(sizeof(NODE));
30  tmp->nodeName='c';
31  tmp->next=NULL;
32  q->next=tmp;
33  q=tmp;
```

- ▶ Line 29~31: 產生第三個節點及指定節點的欄位內容
- ▶ Line 32: 把新節點加到串列的結尾
- ▶ Line 33: 讓 q 指向最後一個節點



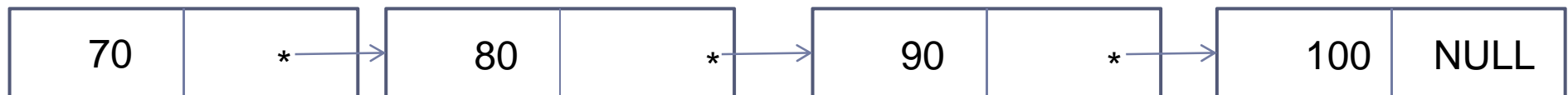
16-1 建立單向鏈結串列 (4/4)

```
35     printf("鏈結串列中的節點資料:\n");
36     tmp=p;
37     while (tmp != NULL) {
38         printf("%c\n", tmp->nodeName);
39         tmp=tmp->next;
40     }
```

- ▶ Line 36: 讓tmp指向串列的開頭
- ▶ Line 38: 輸出目前節點的nodeName欄位
- ▶ Line 39: 讓tmp指向下一個節點

練習題

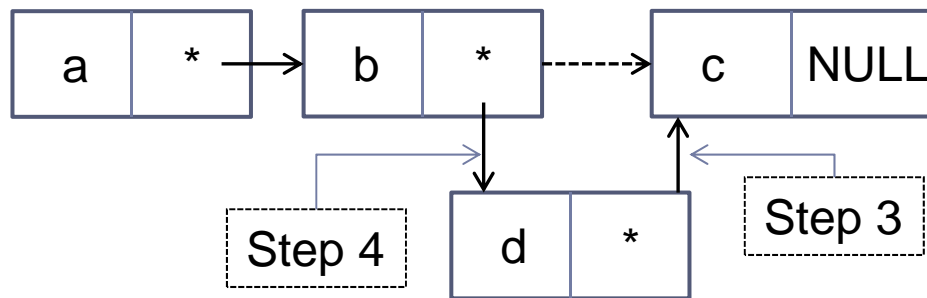
- ▶ 開啟SinglelyLinkedList_2.cpp，完成指定步驟建立一個記錄C語言成績的鏈結串列，串列共含4個節點，串列參考內容如下，建立完畢後，輸出串列內容及平均成績，程式執行結果參閱以下視窗畫面。



```
C語言成績:
=====
      70
      80
      90
     100
=====
平均: 85.00
請按任意鍵繼續 . . .
```

16-2 在單向鏈結串列的中間插入節點

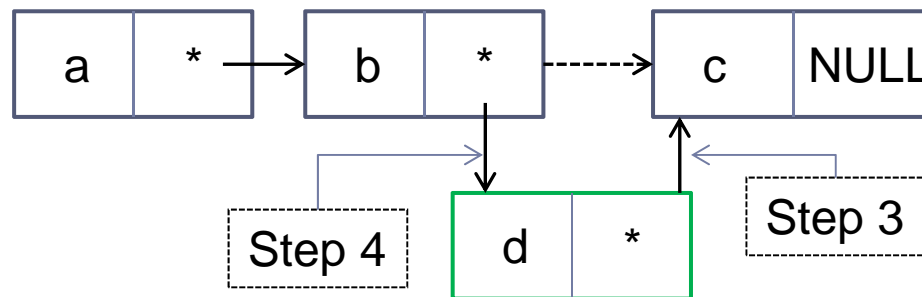
- ▶ 若要在以下單向鏈結串列的b這個節點後面加入d這個節點，則其實作程式序如下：
 1. 建立新節點d
 2. 從鏈結串列中找到b這個節點（要新增的位置）
 3. b所指向的節點改由d來指向(如下圖的step 3)
 4. 讓b指向新節點d(如下圖的step 4)



16-2在單向鏈結串列中插入節點

```
41 //建立所要新增的節點
42 newNode=(NODE *)malloc(sizeof(NODE));
43 newNode->nodeName='d';
44 newNode->next=NULL;
45 //找到要新增的位置
46 tmp=p;
47 while (tmp->nodeName != 'b') {
48     tmp=tmp->next;
49 }
50 newNode->next=tmp->next;
51 tmp->next=newNode;
52 printf("\n新增節點後鏈結串列中的節點資料:\n");
53 tmp=p;
54 while (tmp != NULL){
55     printf("%c\n", tmp->nodeName);
56     tmp=tmp->next;
57 }
```

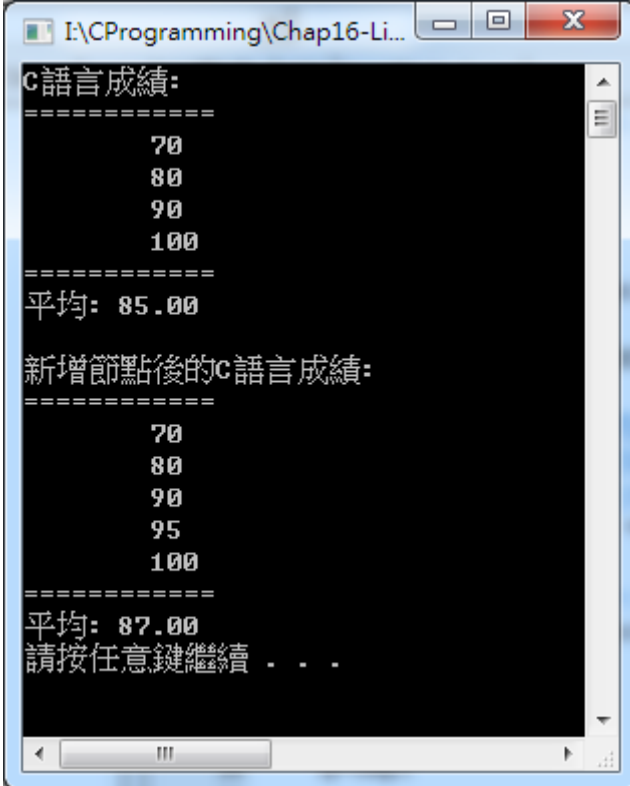
1. 建立新節點d：
Line 42~44
2. 從鏈結串列中找到b這個節點（要新增的位置）：
Line 46~49
3. b所指向的節點改由d來指向(如下圖的step 3)：
Line 50
4. 讓b指向新節點d(如下圖的step 4)：
Line 51



▶ **注意：**step 3與step 4不能對調

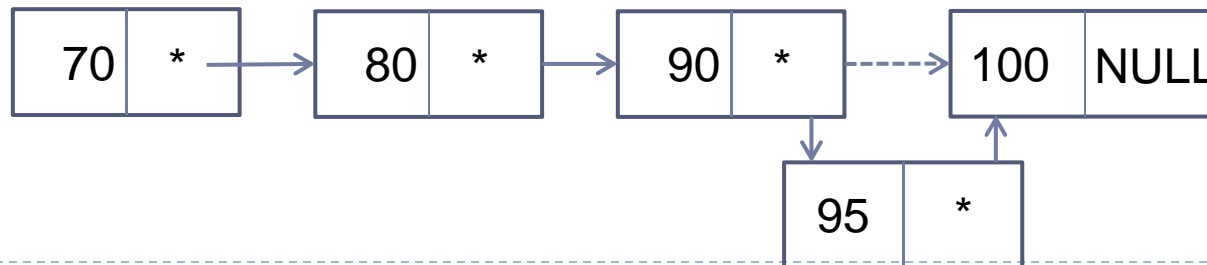
練習題

- ▶ 開啟InsertNode_2.cpp，完成指定步驟以新增一個節點到既有的串列中，新增後的串列共含5個節點，串列參考內容如下。建立完畢後，再次輸出串列內容及平均成績，程式執行結果參閱右圖視窗畫面。



```
C語言成績:
=====
      70
      80
      90
     100
=====
平均: 85.00

新增節點後的C語言成績:
=====
      70
      80
      90
      95
     100
=====
平均: 87.00
請按任意鍵繼續 . . .
```



16-3 在單向鏈結串列中間移除節點

例：下列鏈結串列中，若要移除的節點為b，則移除步驟為：

1. 讓b的前一個節點指向b的下一個節點（即建立綠色部份的連結）
2. 呼叫free函式釋放b所佔用的記憶體空間

