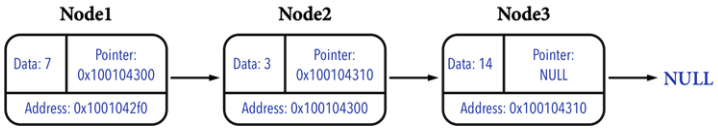
Linked list(鏈結串列)定義

Linked list(鏈結串列)是一種常見的資料結構，其使用 node(節點)來記錄、表示、儲存資料(data)，並利用每個 node 中的 pointer 指向下一個 node，藉此將多個 node 串連起來，形成 Linked list，並以 NULL 來代表 Linked list 的終點。



Linked List 是由「節點」(Node)組成的有序串列集合，節點又稱串列節點(List Node)。每一個節點至少包含一個「資料欄」(Data Field)和「鏈結欄」(Linked Field)。

鏈結串列加入或刪除一個節點非常方便，不需要大幅搬動資料，只要改變鏈結的指標即可。

Array & Linked List 比較

Array

優點：

1. random access：只要利用 index 即可在 O(1)時間對 Array 的資料做存取。
2. 較 Linked list 為節省記憶體空間：因為 Linked list 需要多一個 pointer 來記錄下一個 node 的記憶體位置。

缺點:

1. 新增/刪除資料很麻煩：若要在第一個位置新增資料，就需要 O(N)時間把矩陣中所有元素往後移動。同理，若要刪除第一個位置的資料，也需要 O(N)時間把矩陣中剩餘的元素往前移動

適用時機：

1. 希望能夠快速存取資料。
2. 要求記憶體空間的使用越少越好。

Linked list

優點：

1. 新增/刪除資料較 Array 簡單，只要對 O(1)個 node(所有與欲新增/刪除的 node 有 pointer 相連的 node)調整 pointer 即可，不需要如同 Array 般搬動其餘元素。
2. 若要刪除特定 node，或者在特定位置新增 node，有可能需要先執行 O(N)的

「搜尋」。

3. Linked list 的資料數量可以是動態的，不像 Array 會有 resize 的問題。

缺點：

1. 因為 Linked list 沒有 index，若要找到特定 node，需要從頭(ListNode *first)開始找起，搜尋的時間複雜度為 O(N)。
2. 需要額外的記憶體空間來儲存 pointer。

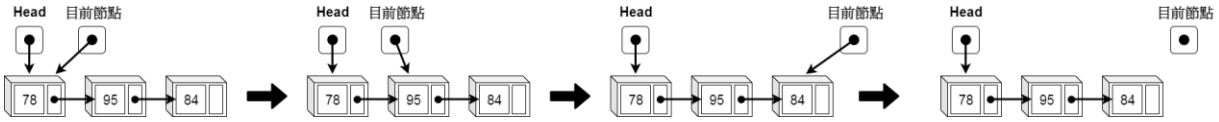
適用時機：

1. 無法預期資料數量時，使用 Linked list 就沒有 resize 的問題。
2. 需要頻繁地新增/刪除資料時。
3. 不需要快速查詢資料。

Linked List 走訪、新增與刪除

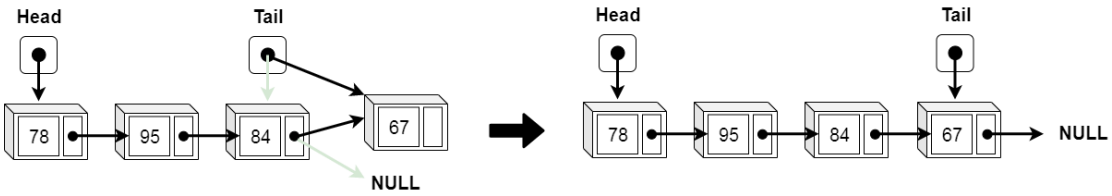
走訪

先設定一個「目前節點」來指向目前的節點，完成走訪就輸出此節點，繼續把「目前節點」移向下一個節點。若「目前節點」變成 None 表示它已走訪完畢。



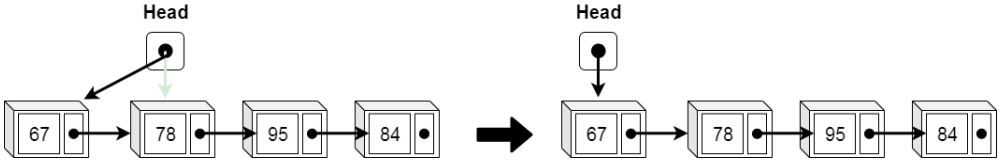
從尾節點加入資料

從尾節點插入 資料時，先把尾節點的指標指向新節點，再把尾節點的指標指向新節點。



從首節點加入資料

把插入的項目設為首節點即可。做法是把加入資料的新節點設為首節點，先以暫存變數儲存，再把指標移向下一個節點即可。



刪除串列的節點

將欲刪除的節點的前一個節點的指標指向欲刪除節點的下一個節點，並把欲刪除的節點的指標設為 None。