

E94116075 駱穎玟 工科系 115 級 程式作業 Bonus

➤ 實現方法

先建立 `structure node` 並將其定義成 `NODE` 的型態

以下分述數種不同函式的實現方法

- `printList`
把 `arr1` 跟 `arr2` 存成 `NODE` 型態的 `first` 和 `second`，然後傳入此函式中，將 `first` 改名為 `node`，`print` 出每個 `node` 的 `data`。
- `createList`
宣告三個 `NODE` 型態變數(`current previous first`)，讓 `for` 迴圈跑陣列長度的次數，用動態陣列配給 `current` 放置空間，並將 `arr` 的值存入 `current` 的 `data` 中，如果 `i=0`，將 `current` 的值給 `first`，`i` 不=0 時將 `previous` 的 `next` 設為 `current`，將 `current` 的 `next` 設為 `NULL`，將目前的 `current` 變成 `previous` 以讓程式繼續進行
- `combineList`
將 `first1` 最後一個 `node` 指向 `first2` 的第一個元素
- `insertFirstNode`
將要插入的 `node` 指向原本第一個 `node`，`return` 要插入的 `node` 即可
- `listlength`
用 `while` 尋找 `list` 中的最後一個值，並記錄總共尋找次數
- `reverseList`
太複雜了，直接用圖說



➤ 心得

在撰寫的過程中我遇到並釐清了以下幾項問題

- `int* ptr` 和 `int **ptr` 的差別
查到的資料顯示沒有差別，只是強調的點不同。
- 甚麼時候要用 `void` 宣告函式
沒有回傳任何東西時，而當有回傳東西時則依照回傳的東西來宣告函式的型態。
- 為甚麼 `NODE` 型態的變數常叫 `first`

因為 **list** 是由第一個值開始找，所以其實可以用第一個 **node** 去代表整個 **list**，在使用時再利用 **next** 尋找其他串接在後面的 **node**

- 如果沒有動態配置記憶體在新節點時
以往都用靜態配置的記憶體，使我常常忘記要使用動態配置記憶體。如果忘記引入動態記憶體，電腦會不停地幫我引入新的 **node**，由此可知引入動態記憶體的環節是不可缺少的
- 動態配置與靜態配置

比較	動態配置	靜態配置
配置記憶體時間	執行時期	編譯時期
彈性空間	有	無
例子	malloc	int num[3]

- 其他可能用到的函式
除了題目要求的程式，我順便完成了 **insertnode** 跟 **searchnode** 的函式，藉由兩者互相搭配便能在指定的 **node** 後方插入新的 **node**

```
NODE *insertnode(NODE *node,int item){
    NODE *newnode=malloc(sizeof(NODE*));
    newnode->data=item;
    newnode->next=node->next;
    node->next=newnode;
}

NODE *searchnode(NODE *first,int num){
    NODE *node=first;
    while((node->data)!=num){
        printf("%d", (node->next)->data);
    }
}
```

而這次是我第一次接觸指標相關的操作，其實滿令人費解，例如動態配置對我來說十分抽象，我覺得程式語言就跟學其他語言的過程一樣，需要一段適應時間，但成為習慣之後就會有卓越的進步。

➤ 參考資料

[C 刷考古] 字串反轉、Linked list 反轉 <https://www.delftstack.com/zh-tw/tutorial/data-structure/linked-list-reversal-algorithm/>

連結串列反轉| D 棧 - Delft

Stack<https://samuel830209.medium.com/c%E5%88%B7%E8%80%83%E5%8F%A4-%E5%AD%97%E4%B8%B2%E5%8F%8D%E8%BD%89-linked-list%E5%8F%8D%E8%BD%89-ec7fed95abf6>

課堂簡報