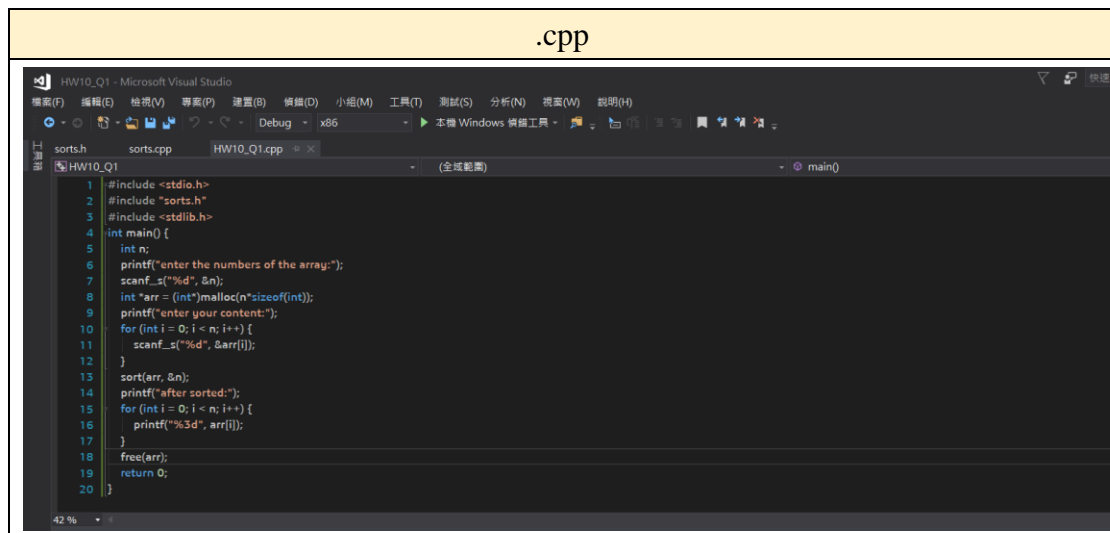
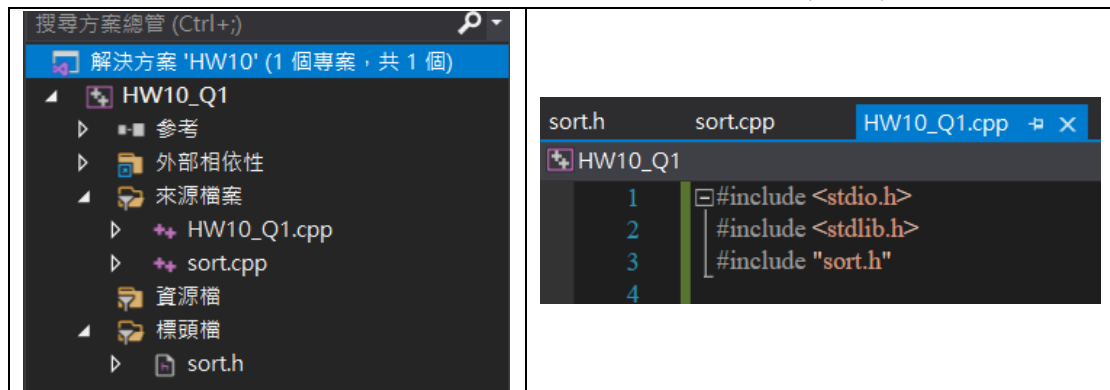
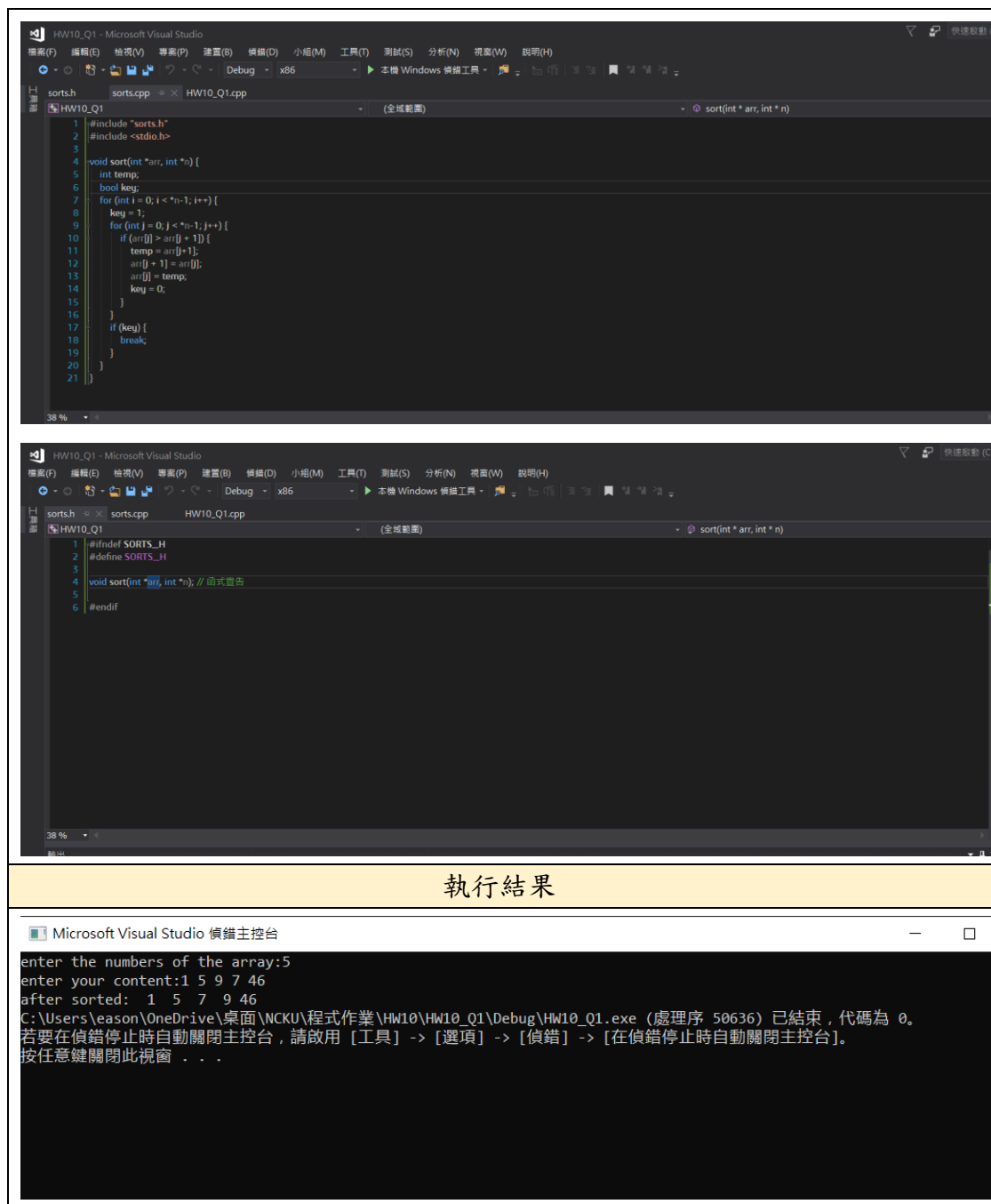


1. 請撰寫一 C 程式，透過泡沫排序法將使用者輸入的一組整數由小到大排序，並顯示排序前後的結果。程式要求如下：
 - 請使用者先輸入欲排序數量 n，再分別輸入 n 個整數。(5%)
 - 使用自訂函式的傳址呼叫(call-by-address)方式。(5%)
 - 將使用者輸入的 n 個整數從小到大排序後輸出。(10%)
 - 請將自訂含式寫在標頭檔(sort.h)及副檔(sort.cpp)中，再 include 進主程式中進行自訂函數的使用。(10%)





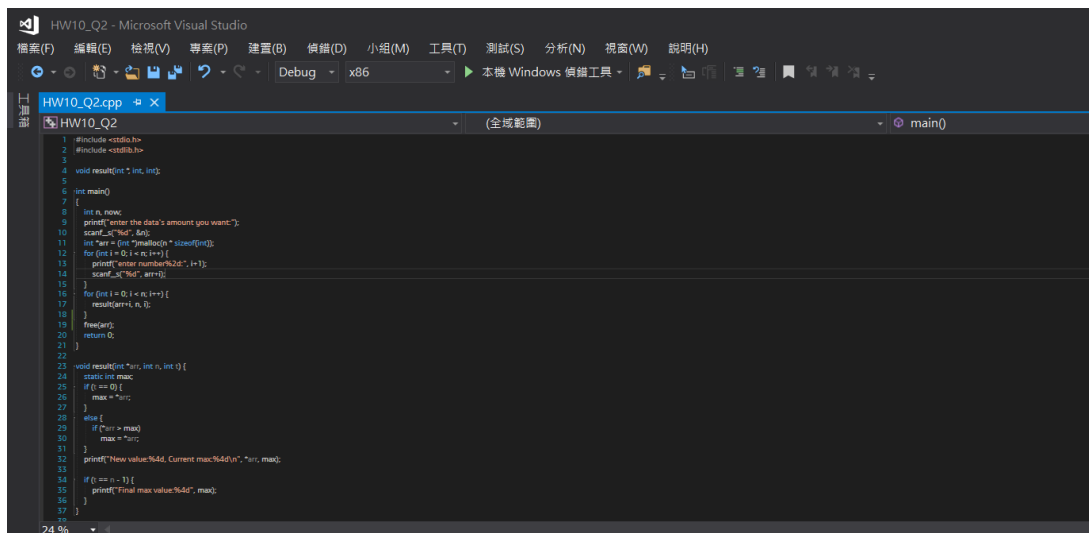
2. 請撰寫一 C 程式，讓使用者輸入 n 個整數，並找出其中的最大值。程式要求如下：

- 請使用者先輸入資料數量 n ，再分別輸入 n 個整數。(5%)
- 每次輸入一個數字後，程式會打印當前輸入的數字以及目前的最大值。(10%)
- 使用靜態變數(static)儲存目前的最大值。(5%)

- 輸入完成後，程式會顯示整個數列的最大值。(10%)


```
輸入共有幾筆資料: 5
輸入第1個整數:32
輸入第2個整數:56
輸入第3個整數:45
輸入第4個整數:6
輸入第5個整數:566
New value: 32, Current max: 32
New value: 56, Current max: 56
New value: 45, Current max: 56
New value: 6, Current max: 56
New value: 566, Current max: 566
Final max value: 566
請按任意鍵繼續 . . . |
```

.cpp



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void result(int *arr, int n);
5
6 int main()
7 {
8     int n, now;
9     printf("enter the data's amount you want:");
10    scanf("%d", &n);
11    int *arr = (int *)malloc(n * sizeof(int));
12    for (int i = 0; i < n; i++) {
13        printf("enter number%d:", i+1);
14        scanf("%d", &arr[i]);
15    }
16    for (int i = 0; i < n; i++) {
17        result(arr+i, n-i);
18    }
19    free(arr);
20    return 0;
21 }
22
23 void result(int *arr, int n, int i) {
24     static int max;
25     if (i == 0) {
26         max = *arr;
27     }
28     else {
29         if (*arr > max)
30             max = *arr;
31     }
32     printf("New value:%d, Current max:%d\n", *arr, max);
33
34     if (i == n-1) {
35         printf("Final max value:%d", max);
36     }
37 }
```

執行結果



```
Microsoft Visual Studio 偵錯主控台
enter the data's amount you want:5
enter number 1:98
enter number 2:2
enter number 3:0
enter number 4:198
enter number 5:45
New value: 98, Current max: 98
New value: 2, Current max: 98
New value: 0, Current max: 98
New value: 198, Current max: 198
New value: 45, Current max: 198
Final max value: 198
C:\Users\eamon\OneDrive\桌面\NCKU\程式作業\HW10\HW10_Q2\Debug\HW10_Q2.exe (處理序 30224) 已結束, 代碼為 0.
若要在偵錯停止時自動關閉主控台, 請啟用 [工具] -> [選項] -> [偵錯] -> [在偵錯停止時自動關閉主控台].
按任意鍵關閉此視窗 . . .
```

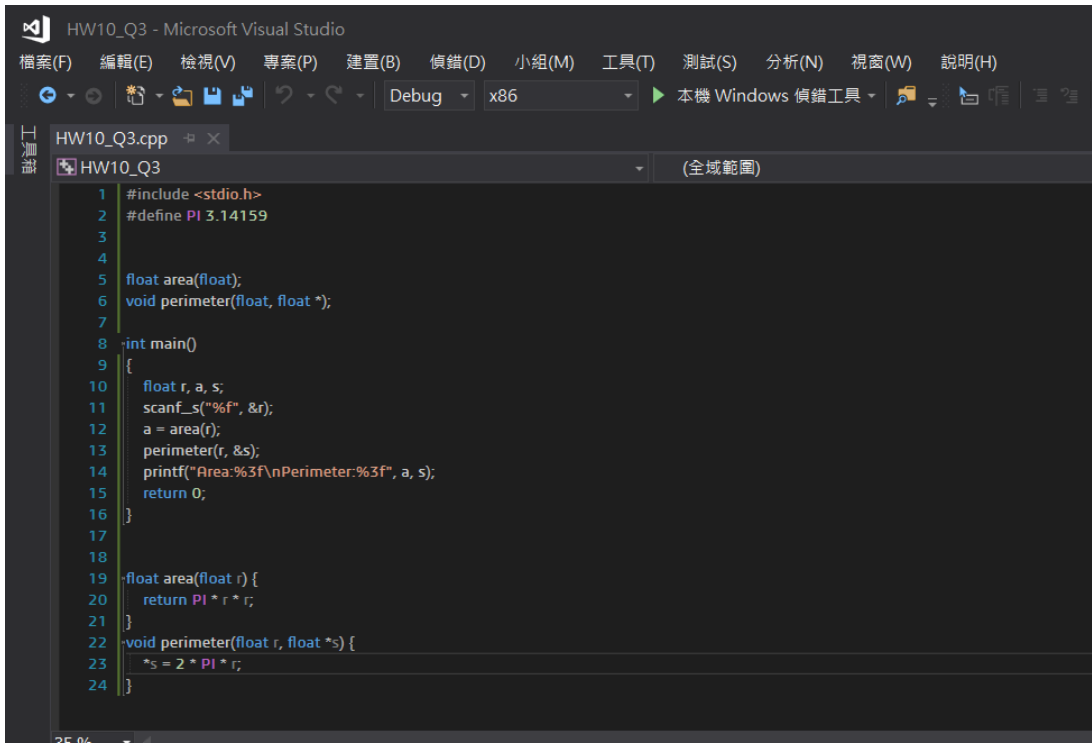
3. 請撰寫一個 C 程式，讓使用者輸入圓的半徑，分別透過傳值呼叫 (Call-by-Value) 和傳址呼叫 (Call-by-Address) 完成以下功能：

- 使用傳值呼叫寫一個函式計算圓的面積，函式接收半徑並返回計算的面積。(10%)
- 使用傳址呼叫寫一個函式計算圓的周長，函式接收半徑與周長變數的指標，將計算結果存入該指標指向的變數。(10%)
- 說明傳值呼叫 (Call-by-Value) 和傳址呼叫 (Call-by-Address) 的差異，簡單描述兩者適用場景。(10%)

說明：

在使用傳值呼叫的時候，會從主函式傳入一個變數的”值”，但是在副函式中改變這個值的時候並不會影響主函式的值，因為兩者所佔據的記憶體位置完全不同，基本上可以當成副函式的那個變數算是一個 copy。在使用傳址呼叫的時候，會從主函數傳述一個變數的”位址”，再傳入副函式之後，因為所使用的記憶體位置和主函式是一樣的，所以在副函式中改變這個值的時候在主函式也會跟著改變。在適用性

上取決於不同情形，像是當我不想讓主函式的值跟著波動而只想在副函式做運算得到結果時，那我就可以用傳址呼叫，像是計算圓面積時我不想讓我的半徑改變，所以我就選擇傳值呼叫。當我想要回傳多個值、保存一些在副函式的多個結果時，我就可以選擇傳址呼叫，像是陣列本身的傳入就是很好的例子，可以將改變後的陣列(多個數值)保存。

.cpp	
	
執行結果	



4. 說明在撰寫自訂函式時，函數型態為 void，以及函數型態為其他資料型態(如 int, float, double, etc.)，差異為何？(10%)

答：當令一個函式的函數型態為 void 時，代表沒有回傳值，所以在函式的最後不能 return 回來，會使編譯器產生錯誤。反之，像是 int、float、double 等等會有回傳值從函數傳回來，而傳回來的數值就是 return 後頭的數值，同時也代表需要使用 return 來回傳東西回來，不然自訂出來的函式就是一段廢 code。