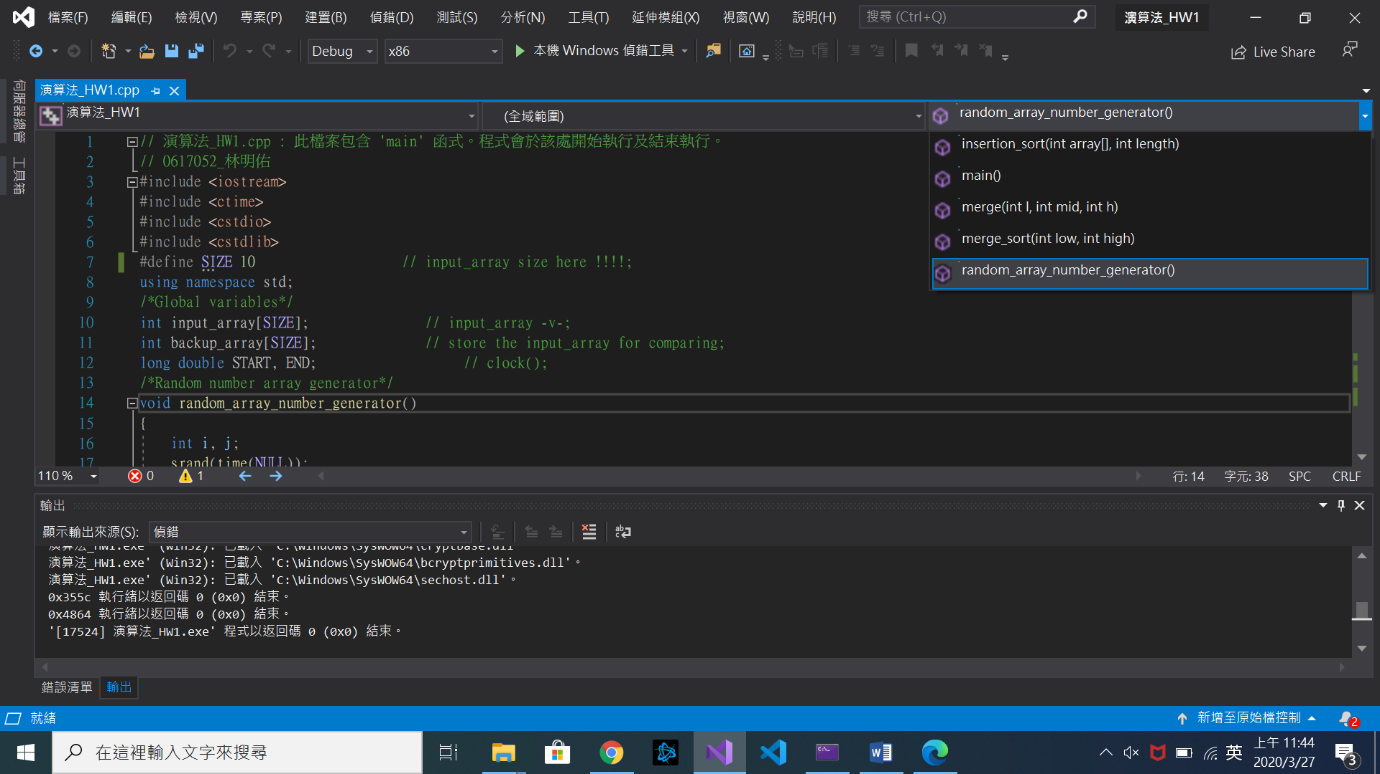
演算法\_HW1 0617052 林明佑

**Description of implementation:**

4 small functions + 1 main function



Input\_array[]: an array stores random numbers.

Insertion\_sort(): insertion sort main function.

Merge\_sort(): merge sort main function.

Merge(): merge two arrays for Merge\_sort.

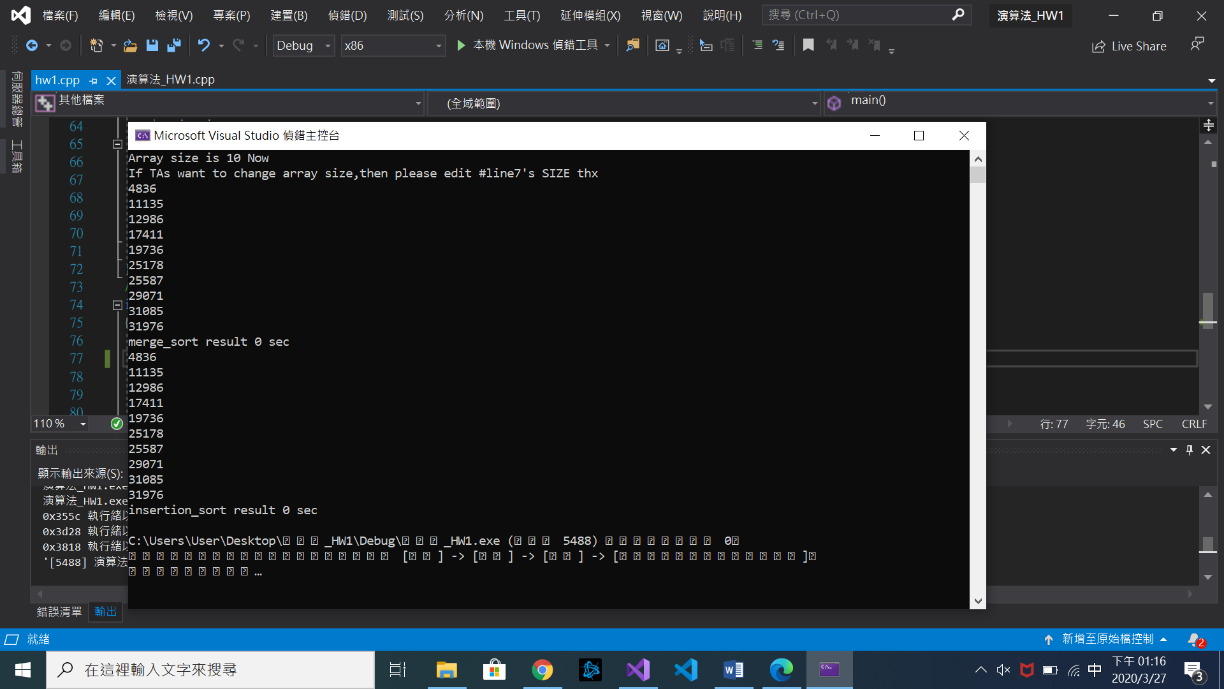
Random\_array\_number\_generator(): generate random numbers for input\_array[].

**Observation of different input\_array[] size:**

Array size 10

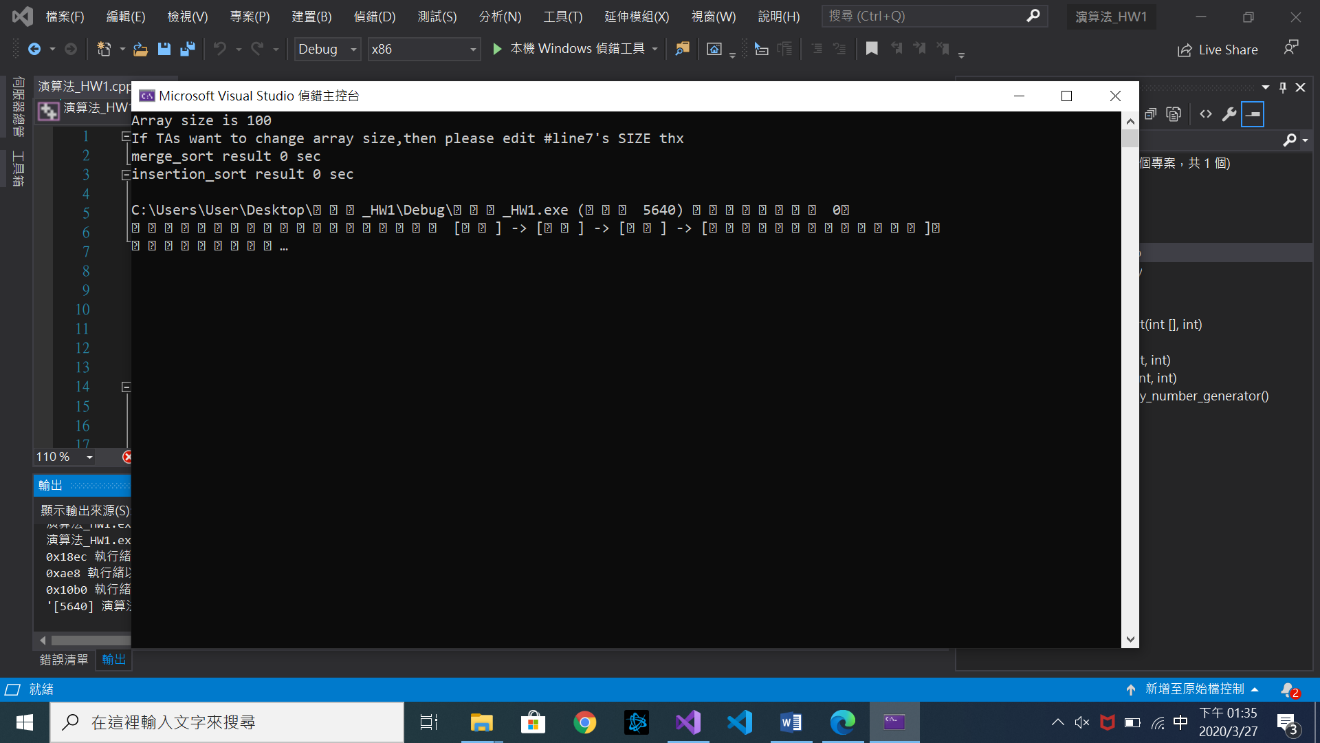
Merge sort < 0.001 sec, Insertion sort < 0.001 sec. (clock precision is 10^-3 sec)

**I print all numbers in the input\_array to make sure that sorts are correctly done.**

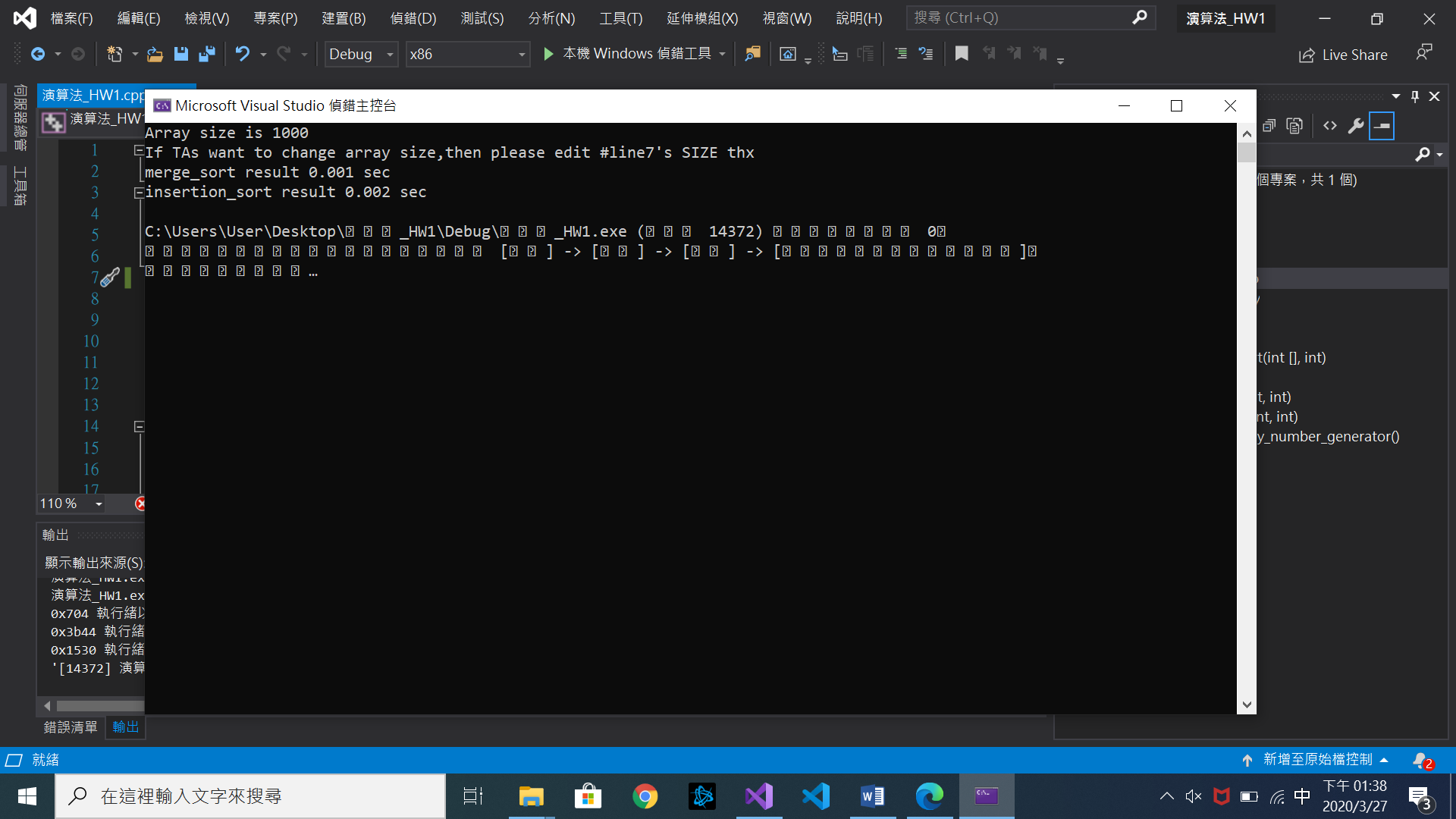


Array size 100

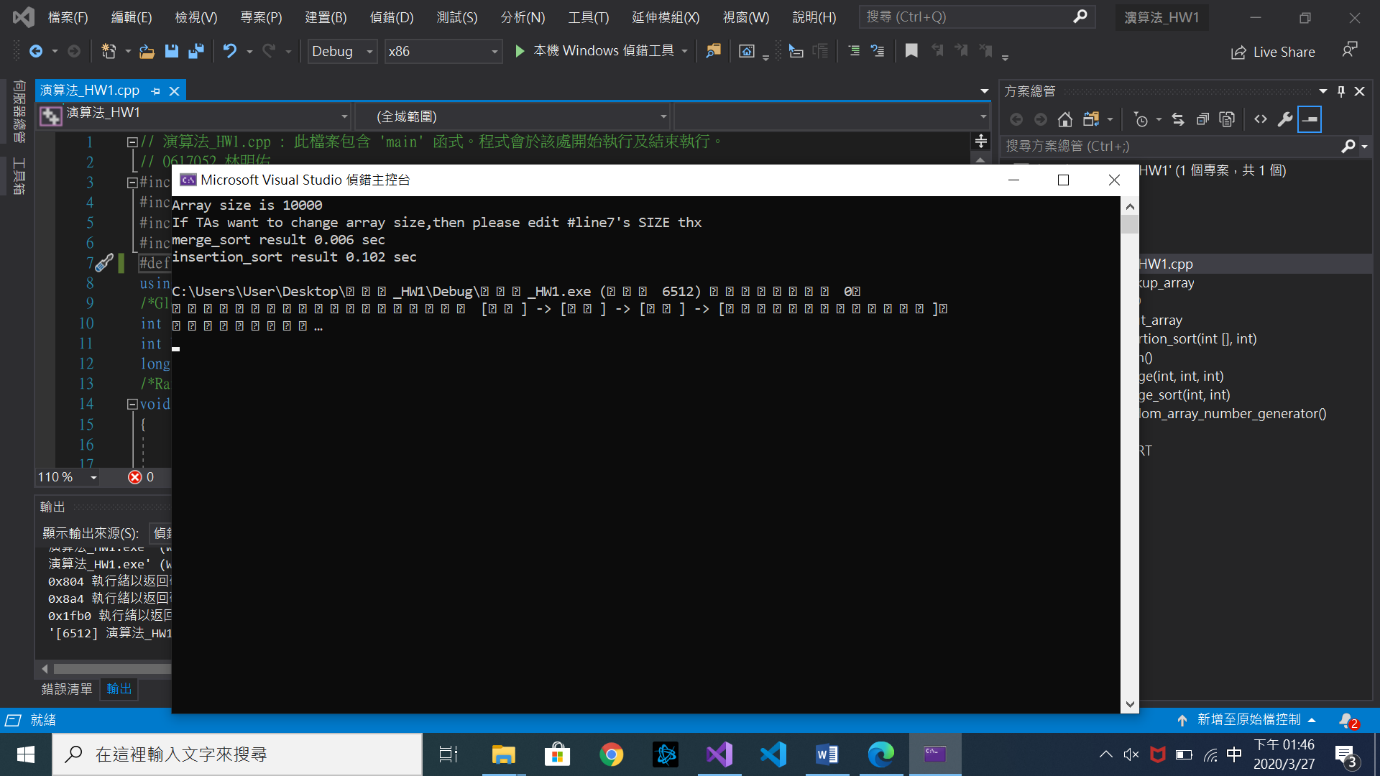
Merge sort < 0.001 sec, Insertion sort < 0.001 sec.

**I think clock precision is not enough, so Array size 100 and 10 have the same result.**

Array size 1000

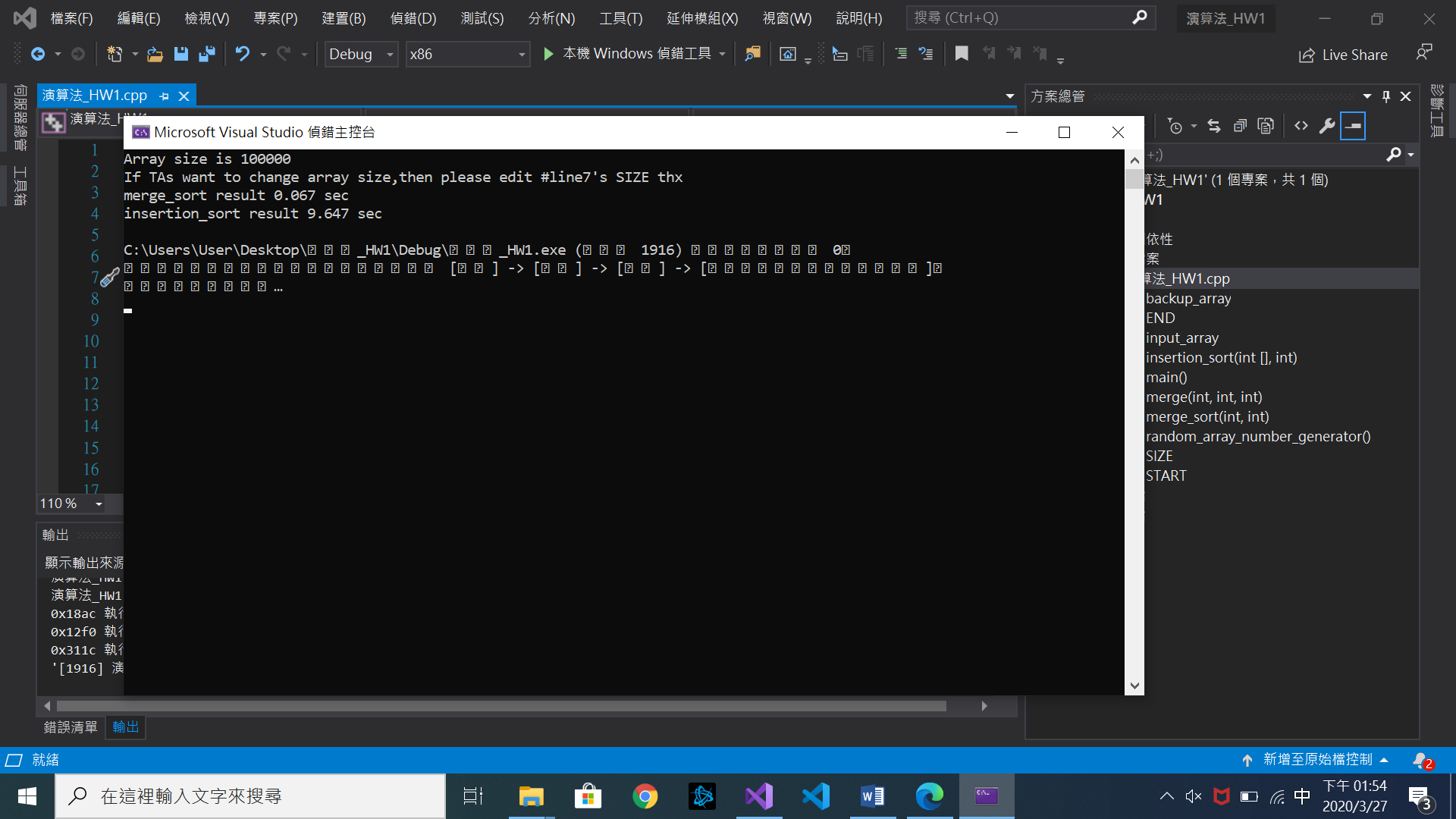
Merge sort 0.001 sec, Insertion sort 0.002 sec.

Array size 10000

Merge sort 0.006 sec, Insertion sort 0.102 sec. **(Much larger difference)**

Array size 100000

Merge sort 0.067 sec, Insertion sort **9.647** sec.



**Summary:**

Insertion sort is **O(*n*2)**, Merge sort is O(nLogn).

從test cases(array size 10, 100, 1000, 10000, 100000)的實驗結果, 可以得知 Insertion sort的run time成長速度永遠都大於Merge sort, 進而驗證老師在課堂上所講O的定義。