

實驗項目 – 函數的參數傳遞

1. 本節目的：

- 學習開發 C 語言程式
- 實現在 Visual Studio 2017 系統設計平台上

2. 設計重點：

- C 語言的函數的參數傳遞

3. 設計步驟：

1. 建立專案

方法 A. 透過 Github Classroom 下載並開啟專案

注意：透過方法 A 建立專案後，直接跳至● 開始依序撰寫 C 語言程式

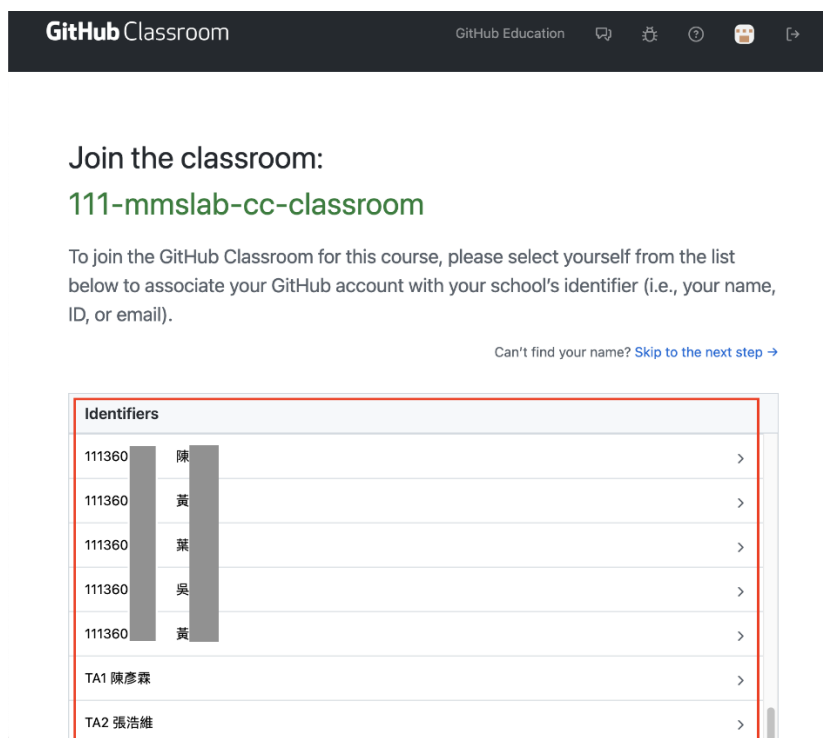
Step1. 點擊 Github Classroom 連結

- Ch4-Lab：<https://tools-api.italkutalk.com/cc/ch4-lab>

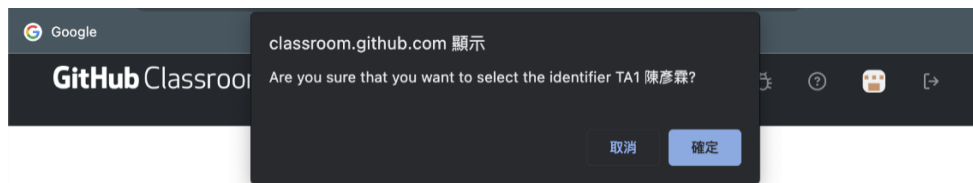
Step2. 將自己的 Github 帳號與 Classroom 學生連結

同一門課程 Github Classroom 的作業或實驗僅需連結一次（若曾經連結過，可以略過此步驟）。

在學生清單中，會列出本門課程尚未被連結的學生，請找到並點擊自己的學號/姓名



點擊後會跳出確認提示，確認無誤點擊確定。

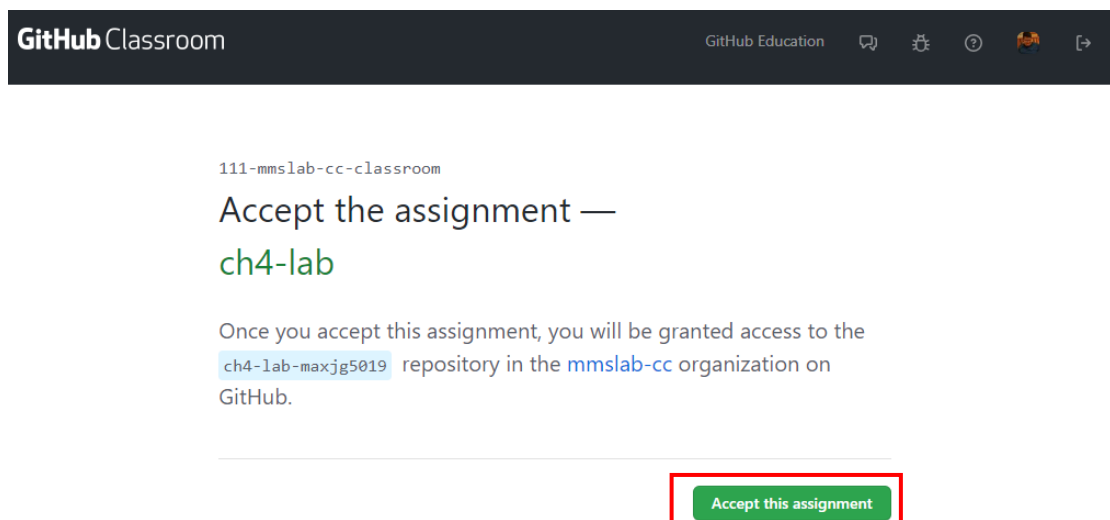


如果遇到以下問題：

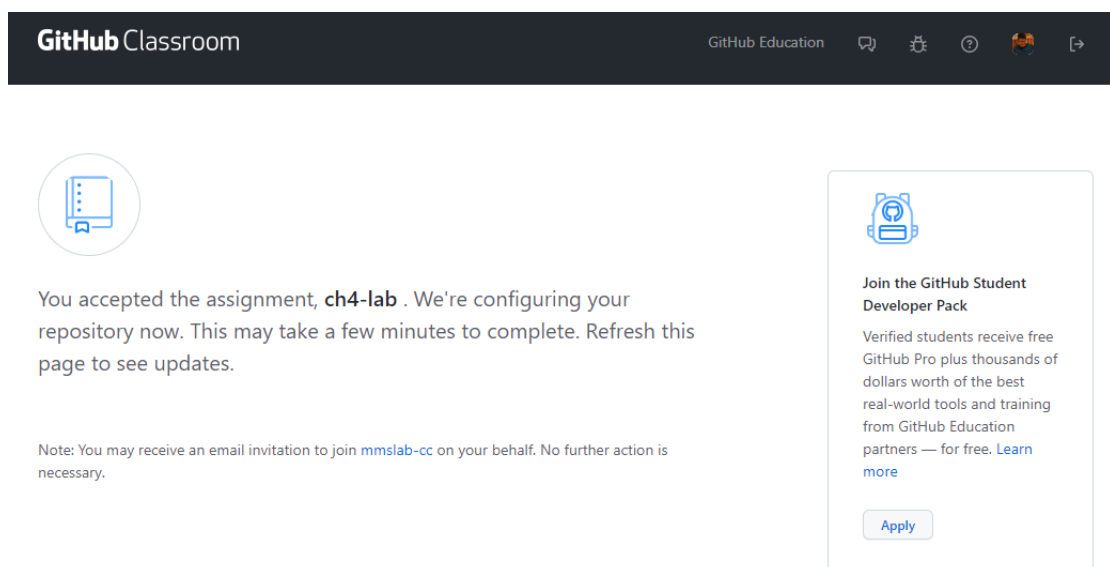
1. 名單中找不到自己的學號姓名
2. 選擇錯人
3. 學號姓名錯誤

請與課堂助教反應，助教將會協助處理。

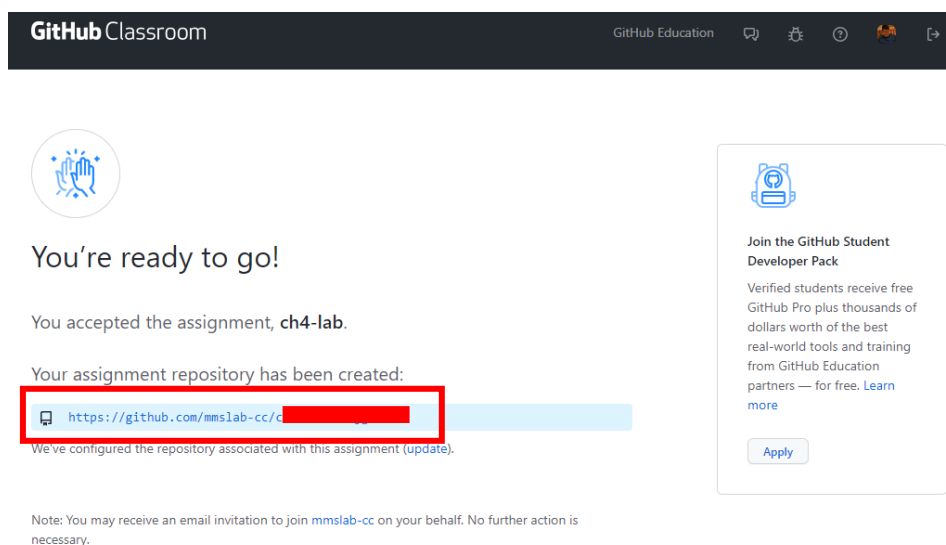
Step3. 接受 Assignment，點擊 Accept this assignment



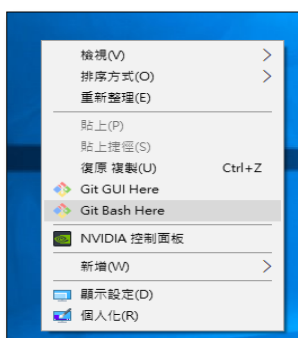
接受 Assignment 後，Github classroom 會幫你建立專屬的 repository，而建立專屬的 repository 需要一段時間，請等 10 秒左右刷新此頁面



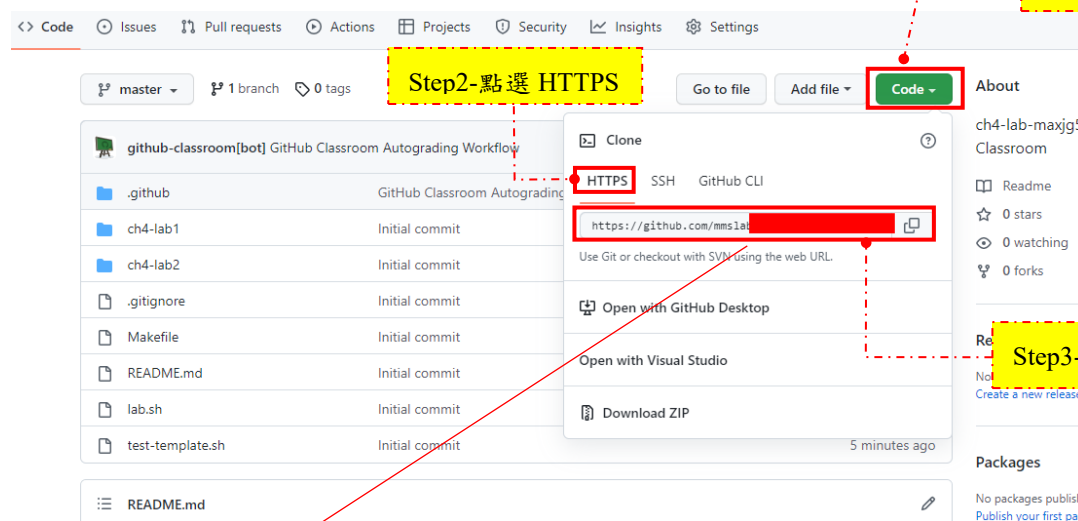
刷新頁面後，將會看到屬於自己的 repository 連結，並點擊該連結。



Step4. 將專案 Clone 到自己電腦
到桌面開啟 Git Bash



複製專案在遠端資料庫的位置，並在 Git bash 輸入指令進行下載專案(命令列點擊右鍵可以選貼上)



\$ git clone <https://github.com/xxx/xxx.git> (記得改成自己的資料庫連結網址)

輸入指令情況：

```
Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop
$ git clone https://github.com/your-github-id/ch4-lab2.git
Cloning into 'ch4-lab2'...
remote: Counting objects: 7, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 0), reused 7 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), done.

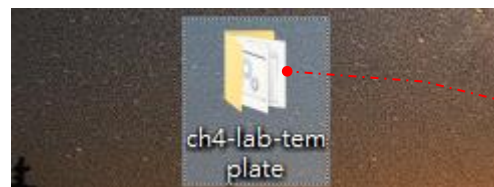
Jdway@DESKTOP-GBM49C1 MINGW64 ~/Desktop
$ |
```

Step5. 完成下載專案後請到桌面開啟剛下載的專案資料夾，開啟專案檔案。

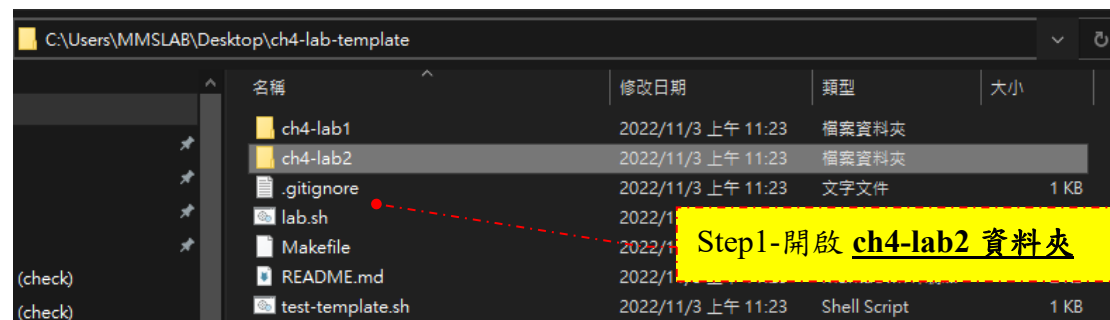
檔案路徑為：Desktop\ch1-lab-{你的 Github 帳號 ID}\

ch4-lab2\Ch4_Lab2_p.X\p.X

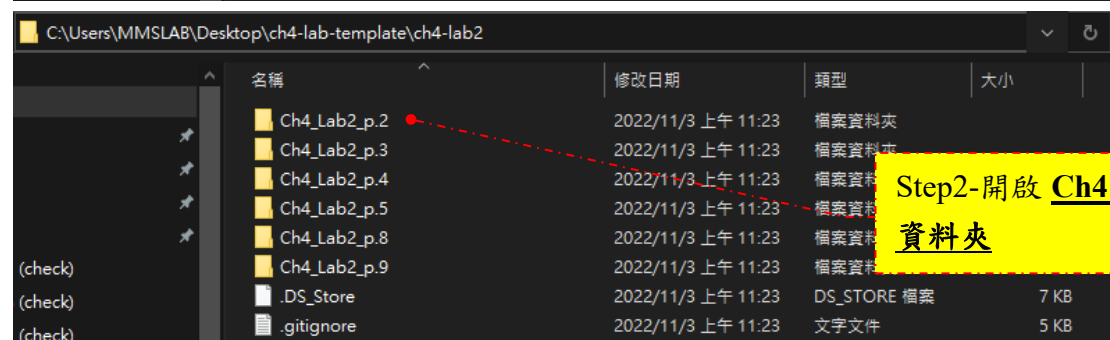
● (此 Lab 較為特殊，X 為各個子專案之代號，下圖用 p.2 示範)



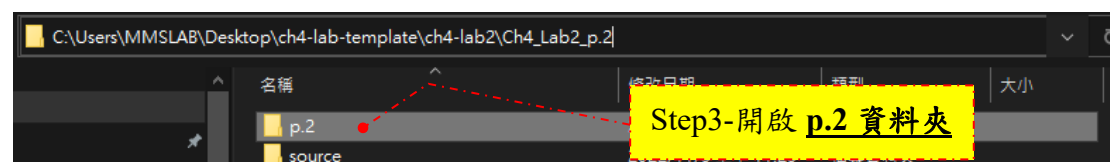
於桌面開啟專案資料夾資料夾
資料夾名稱會是
ch1-lab-{你的 Github 帳號 ID}



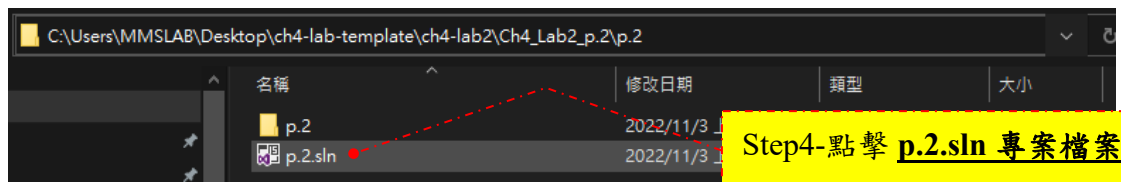
Step1-開啟 ch4-lab2 資料夾



Step2-開啟 Ch4 Lab2 p.2
資料夾



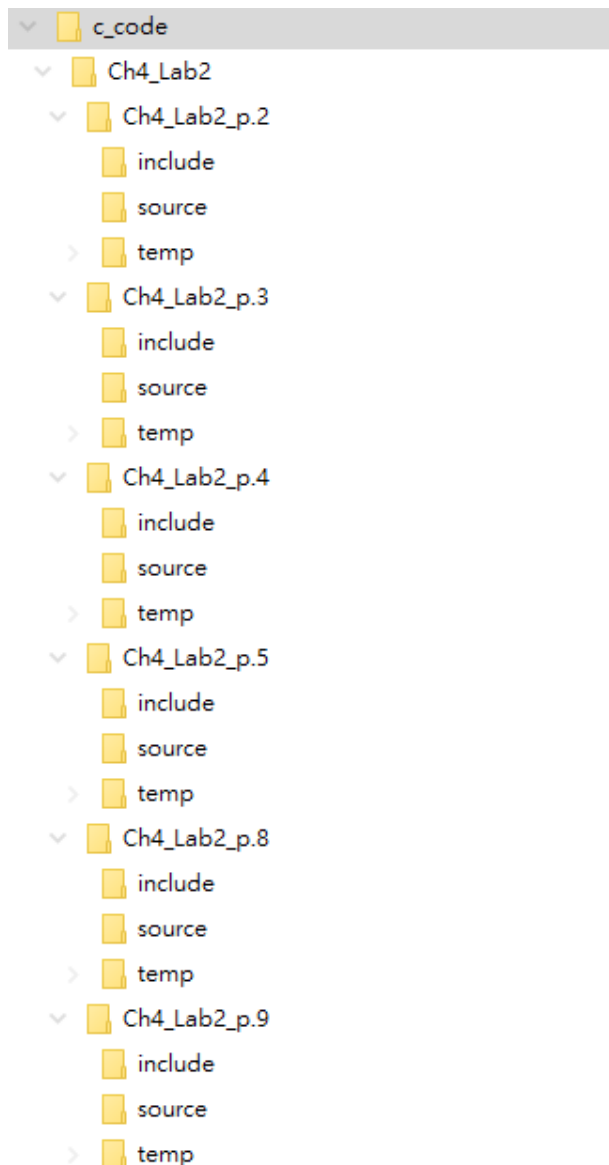
Step3-開啟 p.2 資料夾

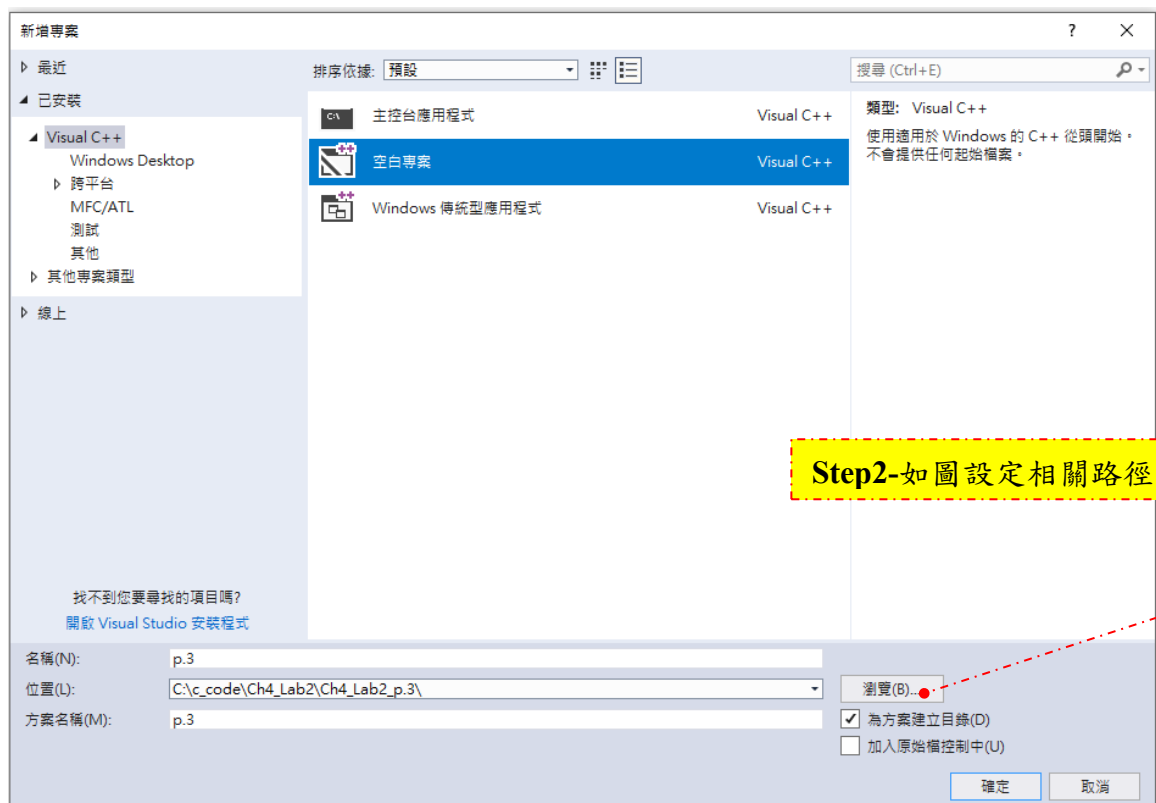
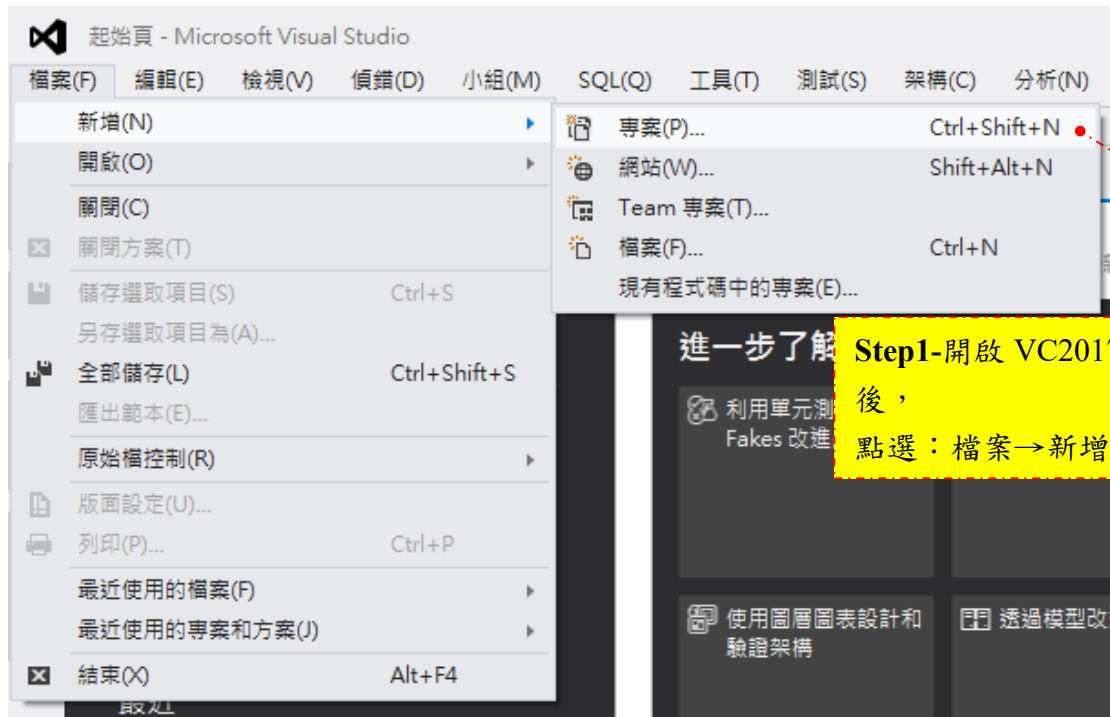


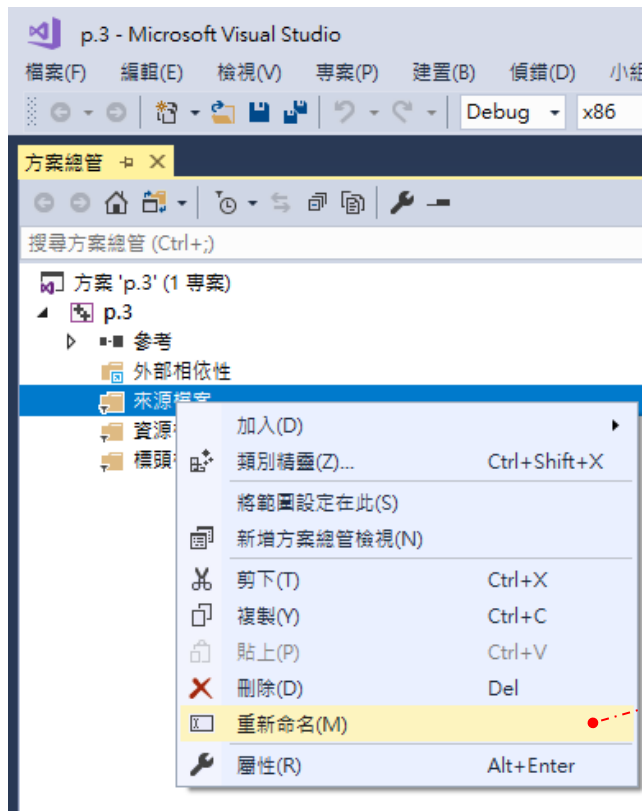
注意：透過方法 A 建立專案後，直接跳至 ● 開始依序撰寫 C 語言程式

方法 B. 透過 Visual Studio 新建專案

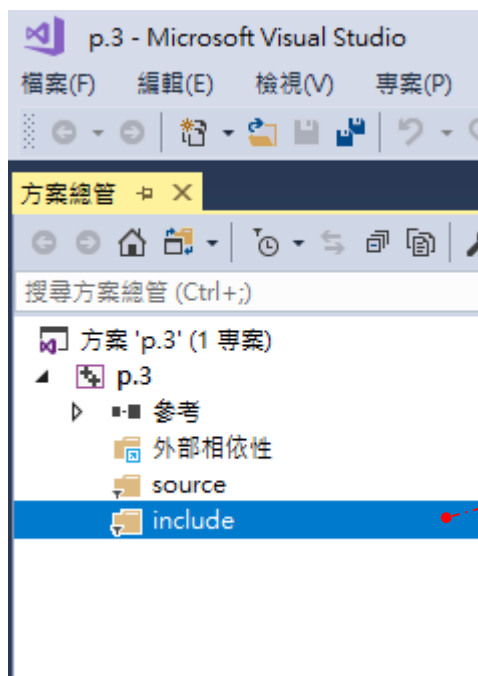
在 C:\c_code 資料夾內新增名為“Ch4_Lab2”的資料夾，再於 Ch4_Lab2 資料夾內建立 Ch4_Lab2_p.2、Ch4_Lab2_p.3...等 6 個資料夾，再分別建立 include、source、temp 等資料夾，建立完成後如下圖



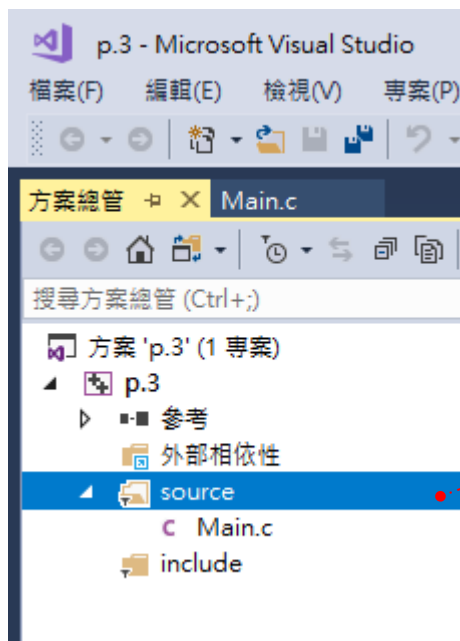
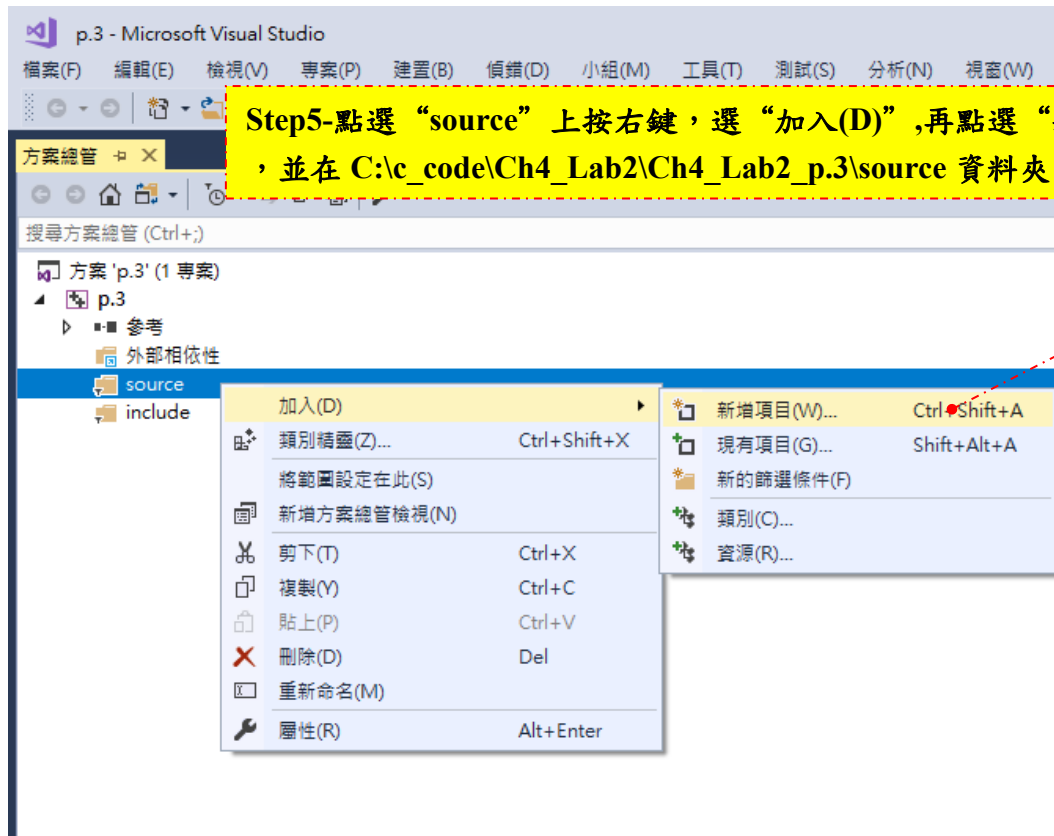




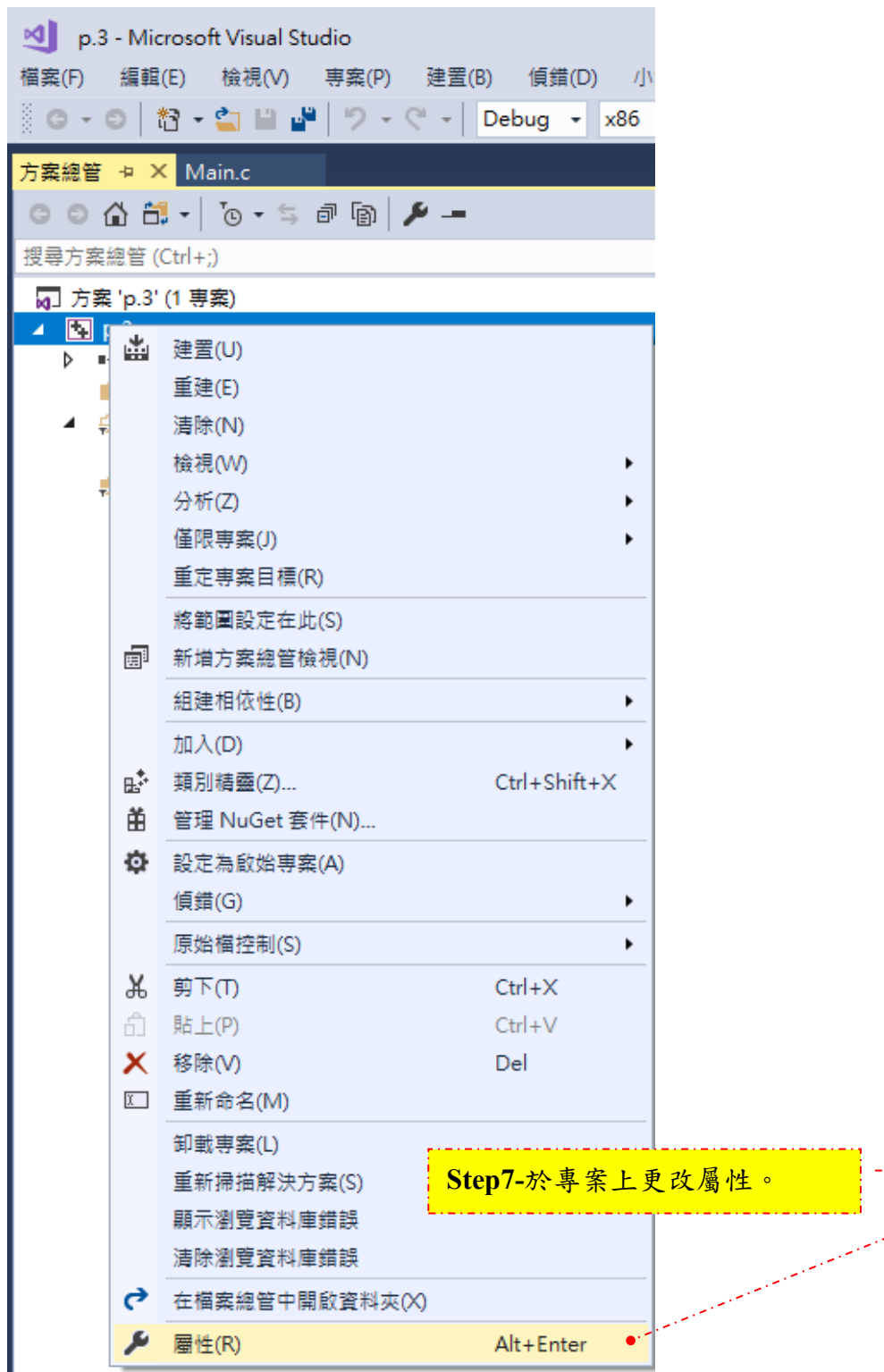
Step3-將原始程式檔重新命名為 source。



Step4-將標頭檔重新命名為 include，並將資源檔刪除，如圖所示。



Step6-完成後，如圖所示。



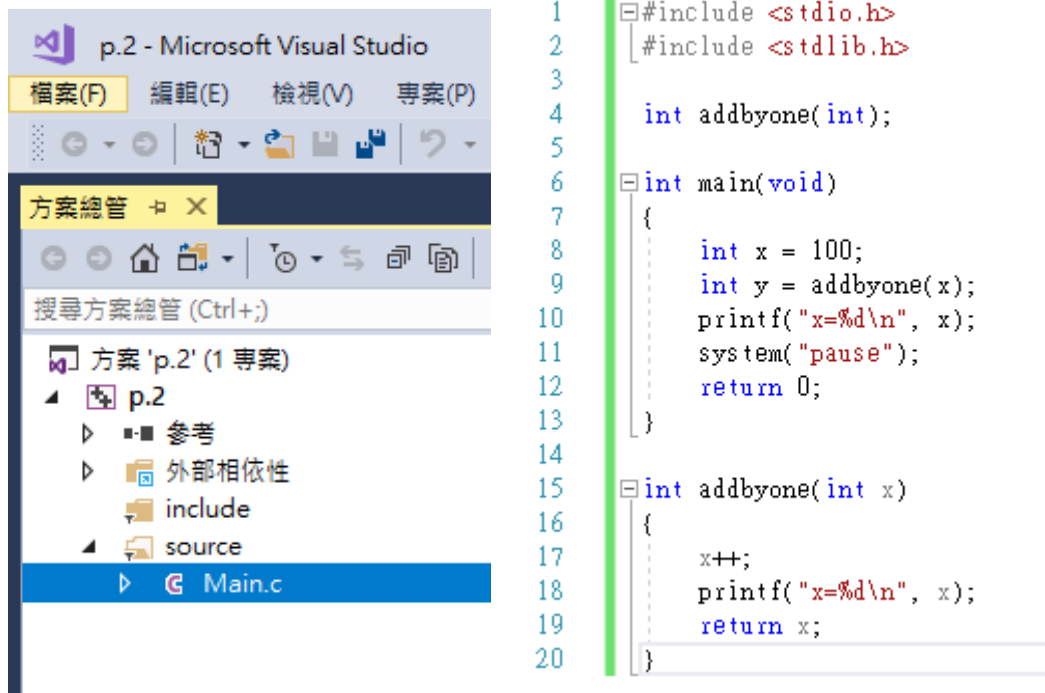
● 開始依序撰寫 C 語言程式

➤ p.2

– Call by value

- 函數呼叫: function(a, b)
- 函數定義: void function(int x, int y)
- 主要把數值拷貝到函示，函示與主程式的變數互不相干

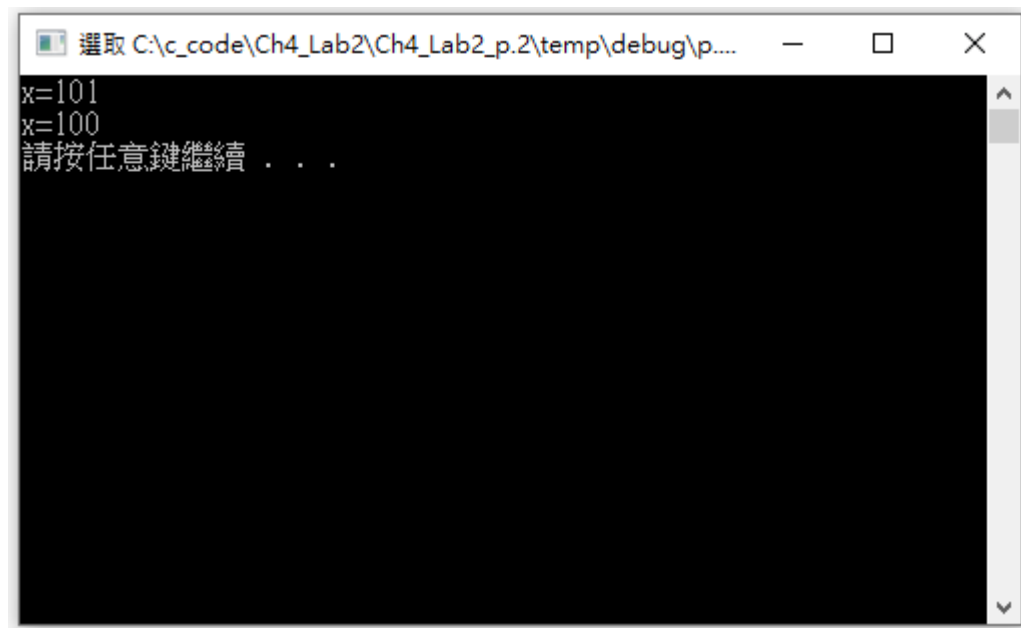
1. 開始撰寫 C 語言程式



The screenshot shows the Microsoft Visual Studio interface. On the left, the 'Solution Explorer' (方案總管) displays the project structure for 'p.2 - Microsoft Visual Studio'. The project 'p.2' contains a 'source' folder with a file named 'Main.c'. The main editor window shows the C code for 'Main.c'.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int addbyone(int);
5
6  int main(void)
7  {
8      int x = 100;
9      int y = addbyone(x);
10     printf("x=%d\n", x);
11     system("pause");
12     return 0;
13 }
14
15 int addbyone(int x)
16 {
17     x++;
18     printf("x=%d\n", x);
19     return x;
20 }
```

● 執行測試結果



The screenshot shows a command prompt window titled '選取 C:\c_code\Ch4_Lab2\Ch4_Lab2_p.2\temp\debug\p....'. The output of the program is displayed as follows:

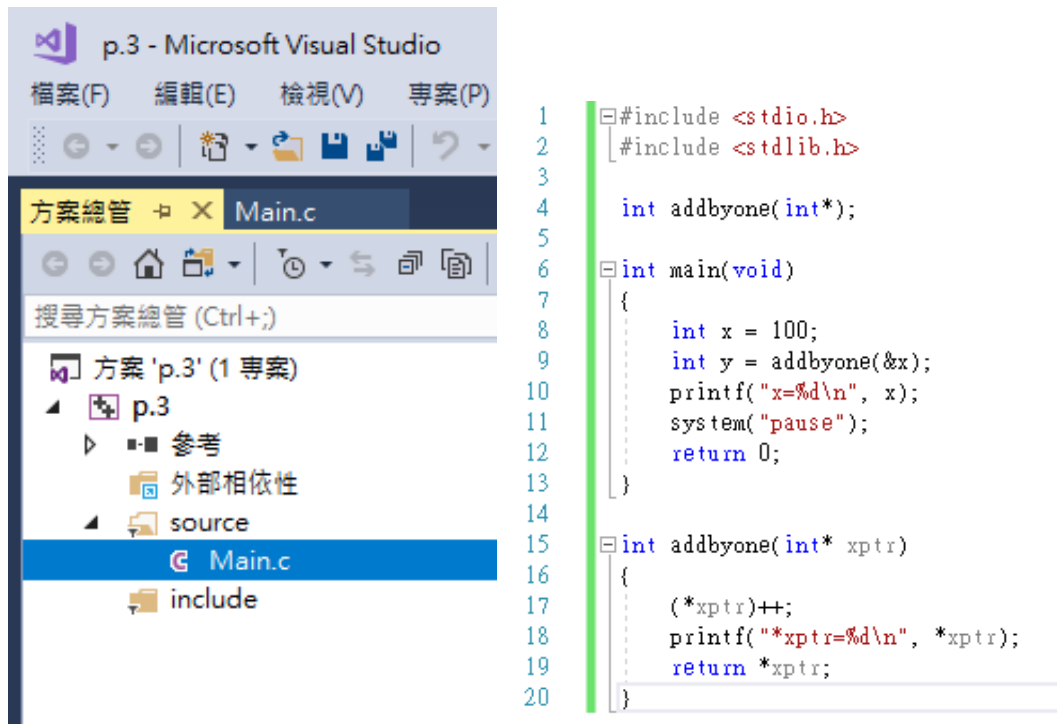
```
x=101
x=100
請按任意鍵繼續 . . .
```

➤ p.3

– Call by address

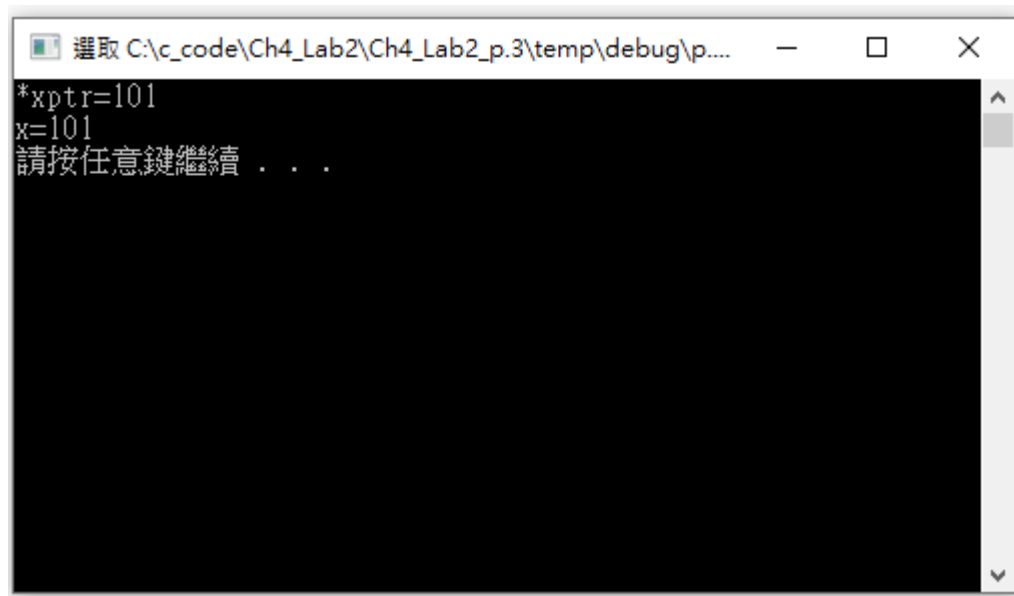
- 函數呼叫: `function(&a, &b)`
- 函數定義: `void function(int * x, int *y)`
- 呼叫函數主要傳給函數位址(&x)，函數則以指標指導相對應的變數 (*xptr)，函數運算會更改相對應的變數內容

1. 開始撰寫 C 語言程式



```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int addbyone(int*);
5
6  int main(void)
7  {
8      int x = 100;
9      int y = addbyone(&x);
10     printf("x=%d\n", x);
11     system("pause");
12     return 0;
13 }
14
15 int addbyone(int* xptr)
16 {
17     (*xptr)++;
18     printf("xptr=%d\n", *xptr);
19     return *xptr;
20 }
```

2. 執行測試結果



```
*xptr=101
x=101
請按任意鍵繼續 . . .
```

➤ p.4

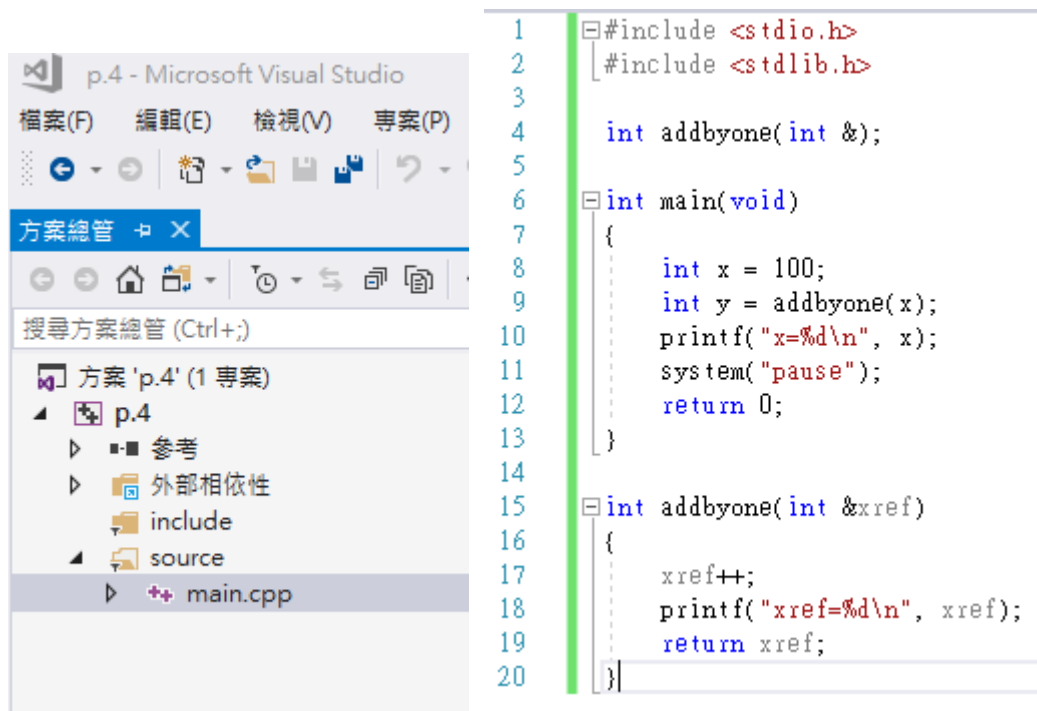
– Call by reference • 函數呼叫: function(a, b)

- 函數定義: void function(int &x, int &y)

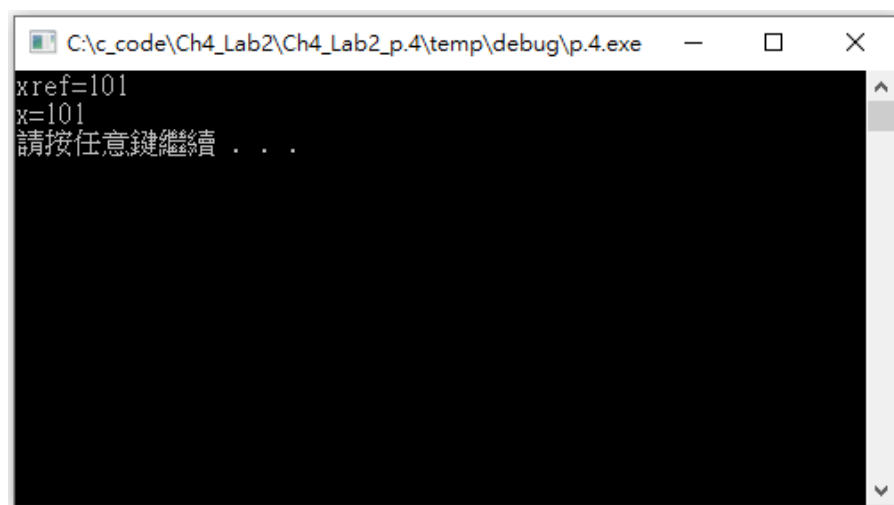
- 呼叫函數主要傳給函數參考變數或物件(x)，函數會以位址(&xref)建立起相連等號，並表示使用相同記憶體空間，函數運算會會更改相對應的變數內容

- 因為 C 沒有支援，需要改用 C++，因此 Main.c 變成 main.cpp 為正常現象

1. 開始撰寫 C 語言程式



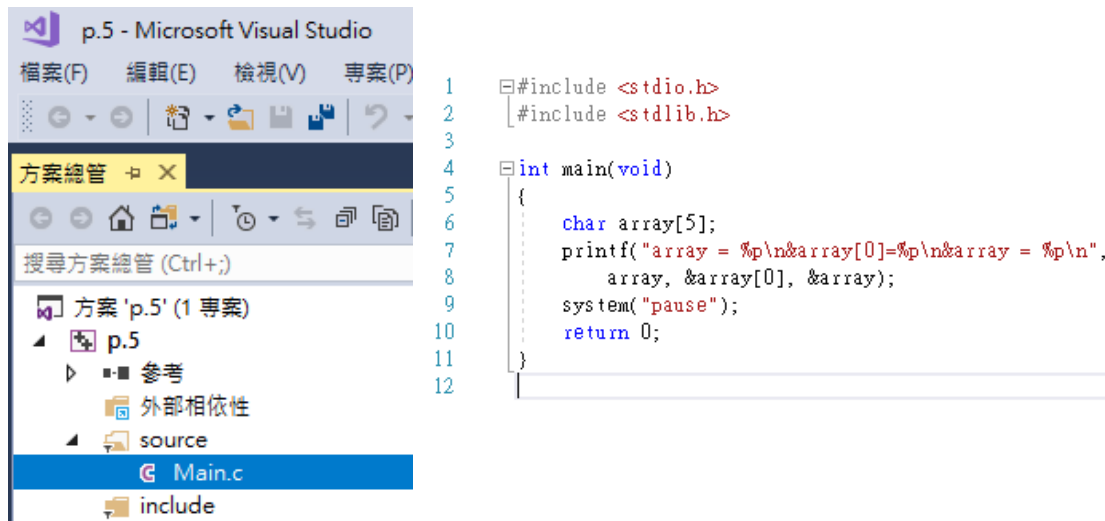
2. 執行測試結果



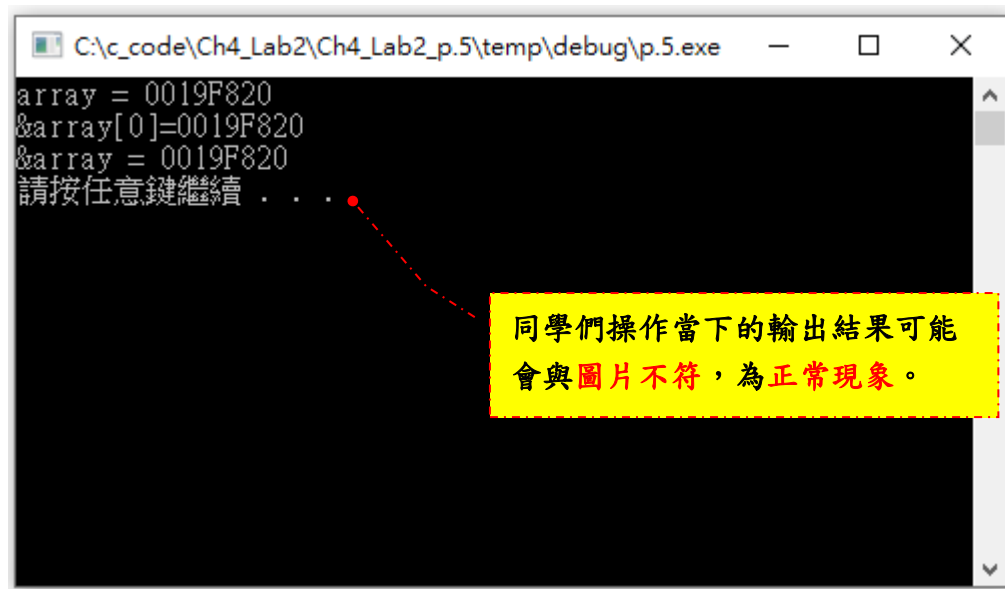
➤ p.5

- 程式利用%p 轉換指定詞 (一個用來列印位址的特殊轉換指定詞) 印出 array, &array[0]和&array, 來驗證陣列名稱確實是此陣列第一個元素所在的位址。
- %p 轉換指定詞通常會將位址以十六進制數的形式印出來。

1. 開始撰寫 C 語言程式



2. 執行測試結果

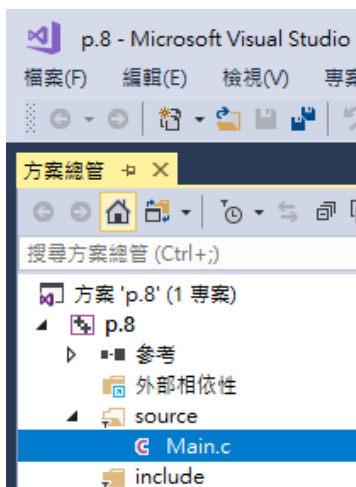


➤ p.8

● 傳遞陣列引數給函式

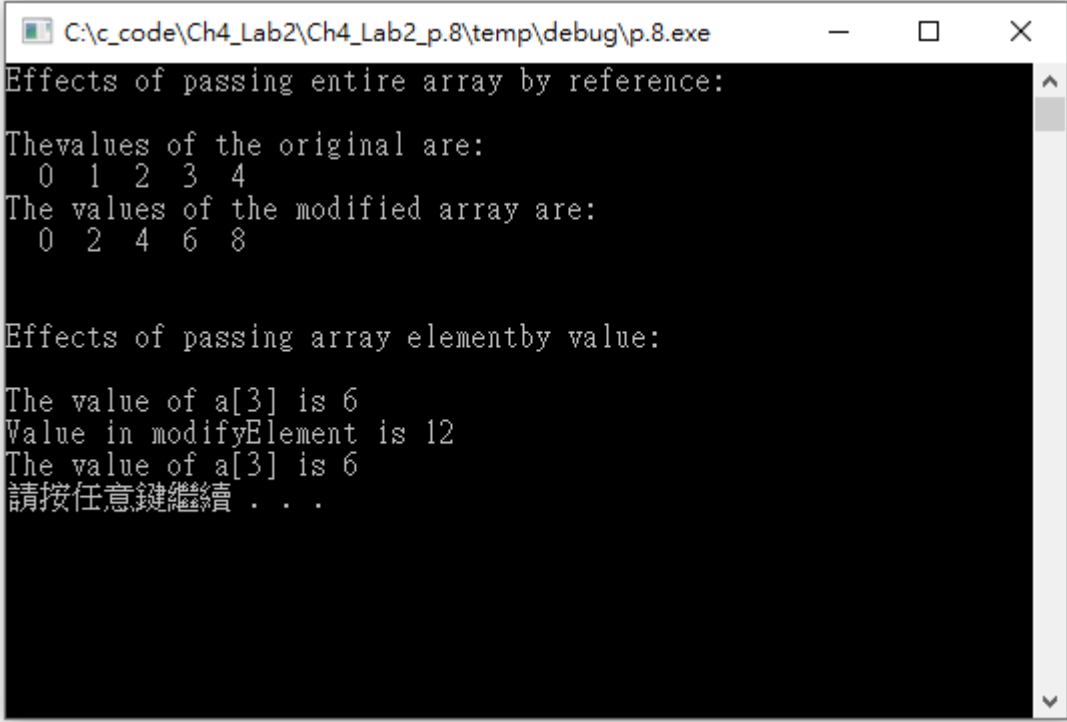
- 陣列(a[5])自動以 Call by reference (傳參考) 來呼叫傳遞
 - 函數呼叫: modifyArray(a)
 - 函數定義: void modifyArray(int b[])
- 參數 b 接收一個整數陣列
- 陣列的中括號裡不需要指定陣列的大小

1. 開始撰寫 C 語言程式



```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define SIZE 5
4
5  void modifyArray(int b[], int size);
6  void modifyElement(int e);
7
8  int main(void)
9  {
10     int a[SIZE] = { 0,1,2,3,4 };
11     int i;
12
13     printf("Effects of passing entire array by reference:\n\nThe"
14           "values of the original are:\n");
15
16     for (i = 0; i < SIZE; i++)
17     {
18         printf("%3d", a[i]);
19     }
20     printf("\n");
21
22     modifyArray(a, SIZE);
23     printf("The values of the modified array are:\n");
24     for (i = 0; i < SIZE; i++)
25     {
26         printf("%3d", a[i]);
27     }
28
29     printf("\n\nEffects of passing array element"
30           "by value:\n\nThe value of a[3] is %d\n", a[3]);
31
32     modifyElement(a[3]);
33     printf("The value of a[3] is %d\n", a[3]);
34
35     system("pause");
36     return 0;
37 }
38
39 void modifyArray(int b[], int size)
40 {
41     int j;
42
43     for (j = 0; j < size; j++)
44     {
45         b[j] *= 2;
46     }
47 }
48
49 void modifyElement(int e)
50 {
51     printf("Value in modifyElement is %d\n", e *= 2);
52 }
53
54
```

2. 執行測試結果



```
C:\c_code\Ch4_Lab2\Ch4_Lab2_p.8\temp\debug\p.8.exe
Effects of passing entire array by reference:

The values of the original are:
0 1 2 3 4
The values of the modified array are:
0 2 4 6 8

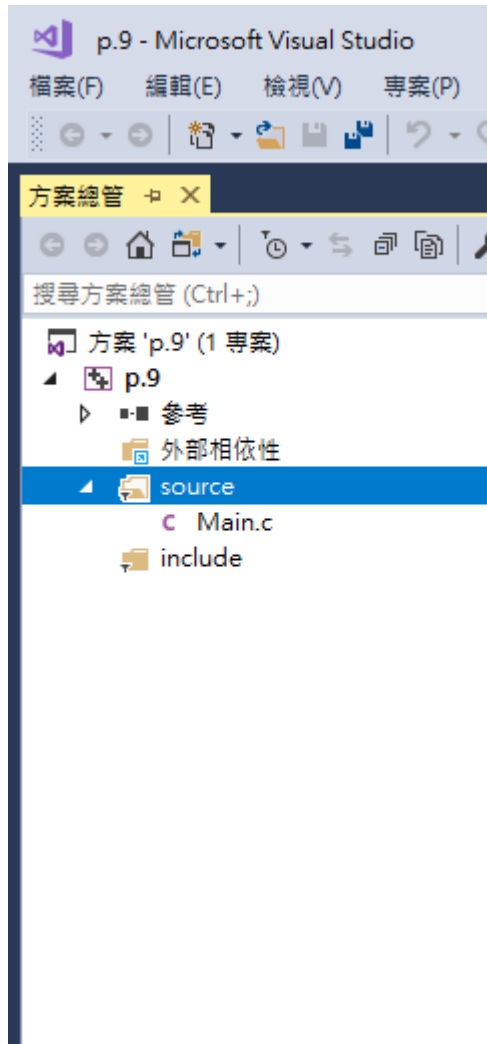
Effects of passing array element by value:

The value of a[3] is 6
Value in modifyElement is 12
The value of a[3] is 6
請按任意鍵繼續 . . .
```


➤ p.9

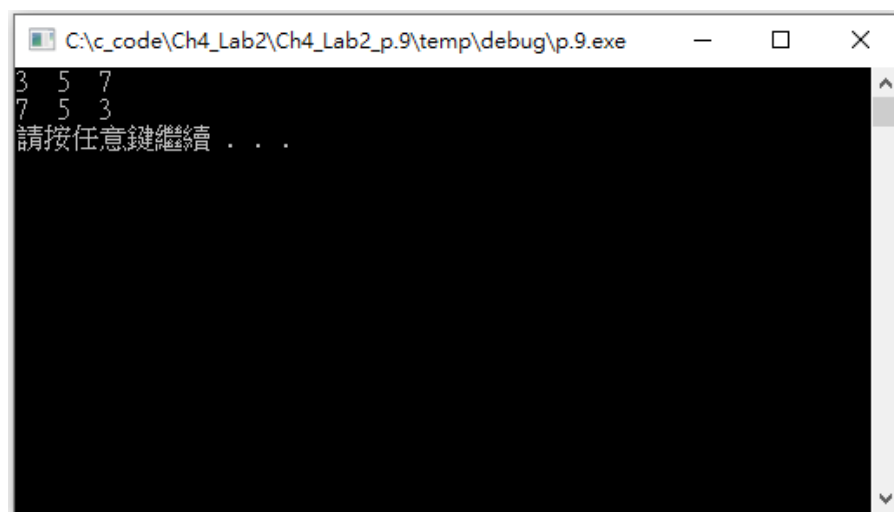
● Call by Address

1. 開始撰寫 C 語言程式



```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void inverse(int *);
5
6  int main()
7  {
8      int a[3] = { 3,5,7 }, i;
9      for (i = 0; i < 3; i++)
10         printf("%d ", a[i]);
11     printf("\n");
12
13     inverse(a);
14
15     for (i = 0; i < 3; i++)
16         printf("%d ", a[i]);
17     printf("\n");
18
19     system("pause");
20     return 0;
21 }
22 void inverse(int *b)
23 {
24     int tmp[3], i;
25     for (i = 0; i < 3; i++)
26         tmp[2 - i] = b[i];
27     for (i = 0; i < 3; i++)
28         b[i] = tmp[i];
29 }
30
31
```

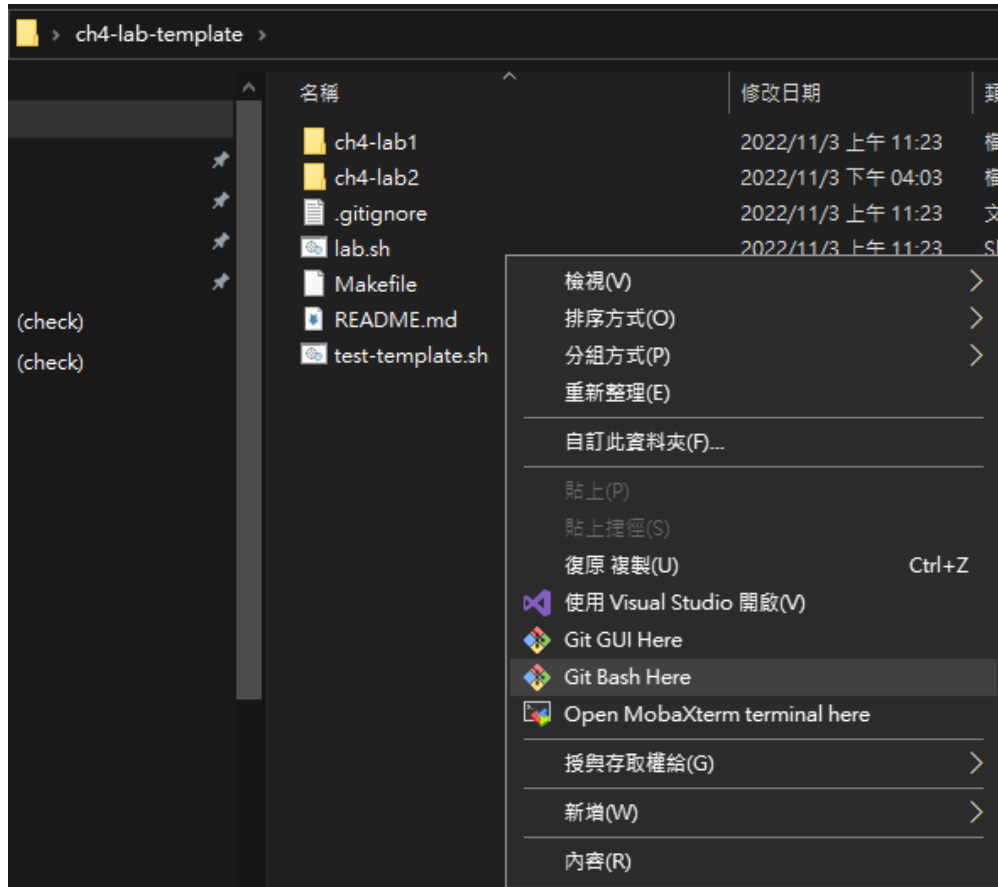
2. 執行測試結果



● 上傳實驗至 Github Classroom

請參考從下方的上傳專案說明，將專案透過 Git 指令 push 到 Github classroom

1. 專案編輯完成後，於本周課程專案的資料夾點選**滑鼠右鍵**，選擇 **Git Bash Here** 開啟 Git 終端機。



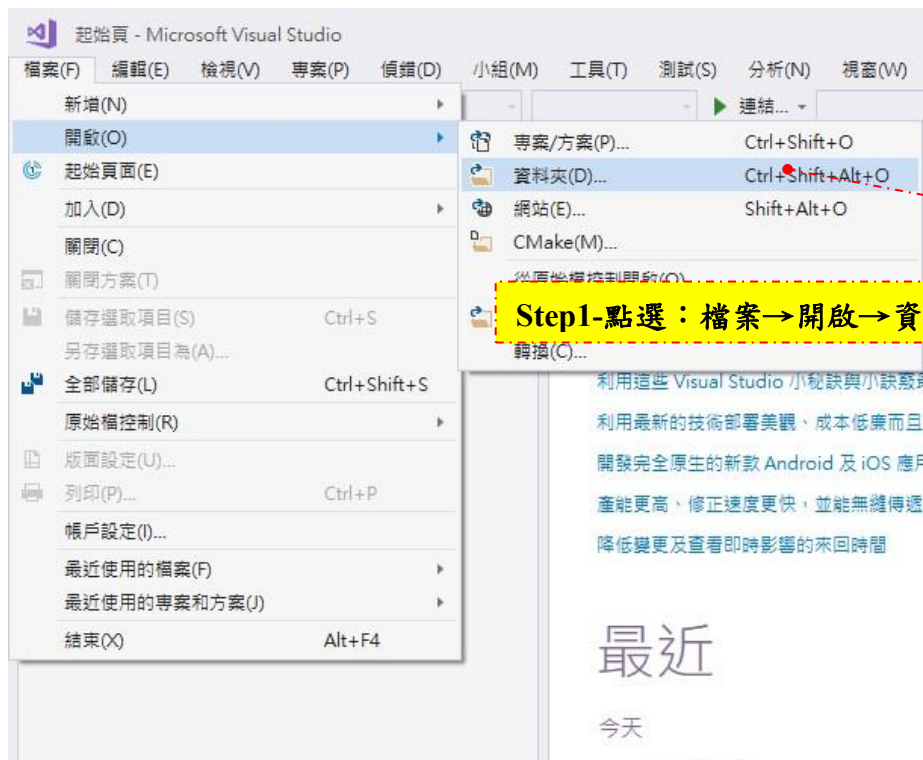
2. 於終端機中依序輸入以下指令，將程式上傳至遠端資料庫。

```
git add .
```

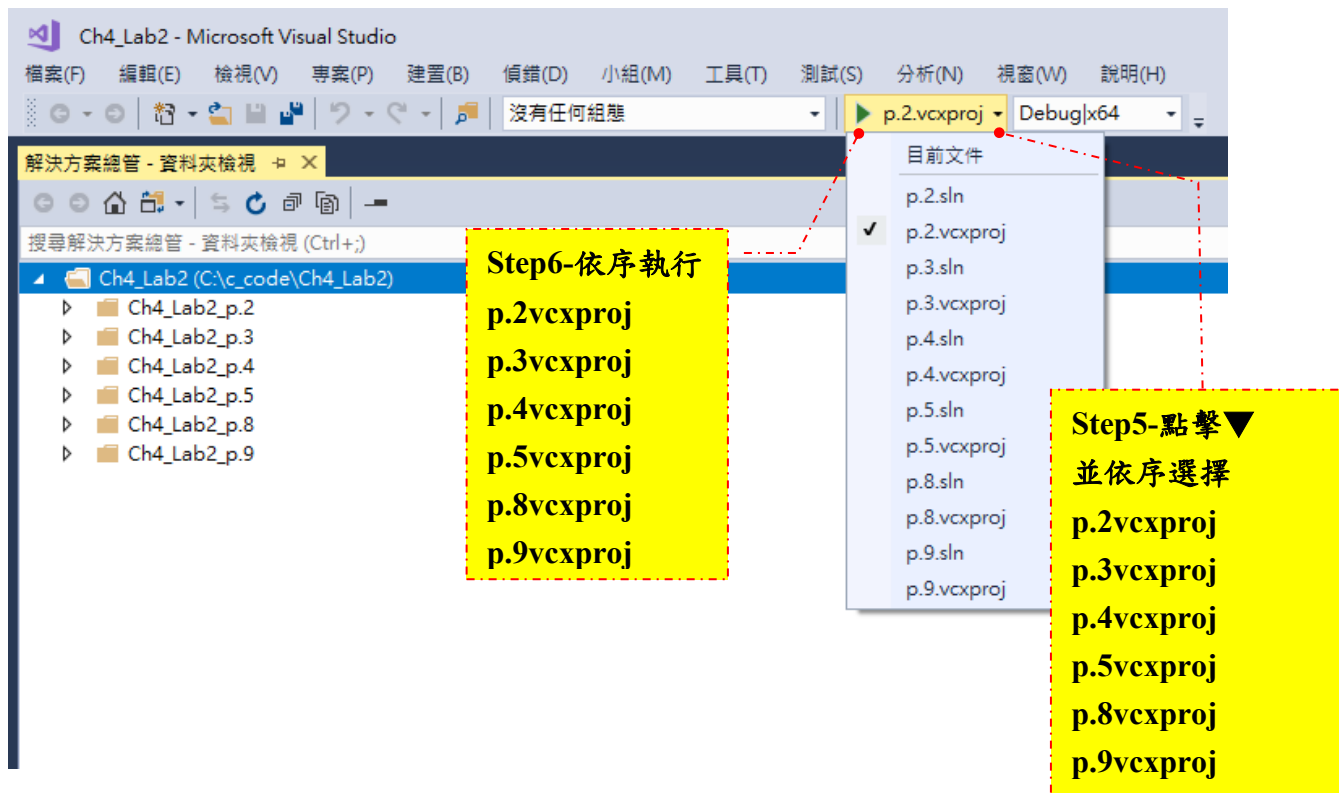
```
git commit -m "[修改的內容描述]"
```

```
git push origin master
```

● 最後將結果呈現給助教檢查



Step4-點擊 選擇資料夾



- 本周課程 Lab 到此結束，謝謝各位同學參閱