

實驗項目-include User_defined(lab1 延伸)

一、本節目的：

- 學習開發 C 語言程式
- 學習如何設計自訂的標頭檔

二、設計重點：

- C 語言的前置處理敘述指令

三、設計步驟：

1. 建立專案

方法 A. 透過 Github Classroom 下載並開啟專案

Step1. 請參考實驗 Lab0-2 的章節 0.1.1 連接 Github Classroom 將實驗專案 clone 至本地。

Step2. Step2. 開啟專案檔案

開啟專案資料夾

資料夾名稱會是 ch1-lab-{你的 Github 帳號 ID}

The screenshots show the following directory structure:

- Root directory: ch1-lab-template (2022/9/28 下午 0...)
- ch1-lab1 (2022/9/28 下午 0...)
- ch1-lab2 (2022/9/28 下午 0...)
- ch1-lab3 (2022/9/28 下午 0...)
- .gitignore (2022/9/28 下午 0...)
- lab1.sh (2022/9/28 下午 0...)
- lab2.sh (2022/9/28 下午 0...)
- lab3.sh (2022/9/28 下午 0...)
- Makefile (2022/9/28 下午 0...)
- README.md (2022/9/28 下午 0...)

Inside ch1-lab2:

- include (2022/9/28 下午 0...)
- Lab2 (2022/9/28 下午 0...)
- source (2022/9/28 下午 0...)
- .gitignore (2022/9/28 下午 0...)

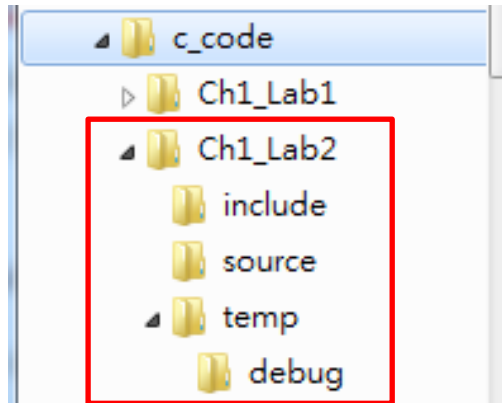
Inside Lab2:

- Lab2 (2022/9/28 下午 0...)
- x64 (2022/9/28 下午 0...)
- Lab2.sln (2022/9/28 下午 0...)

注意：透過方法 A 建立專案後，直接跳至步驟 3. 撰寫 C 語言程式

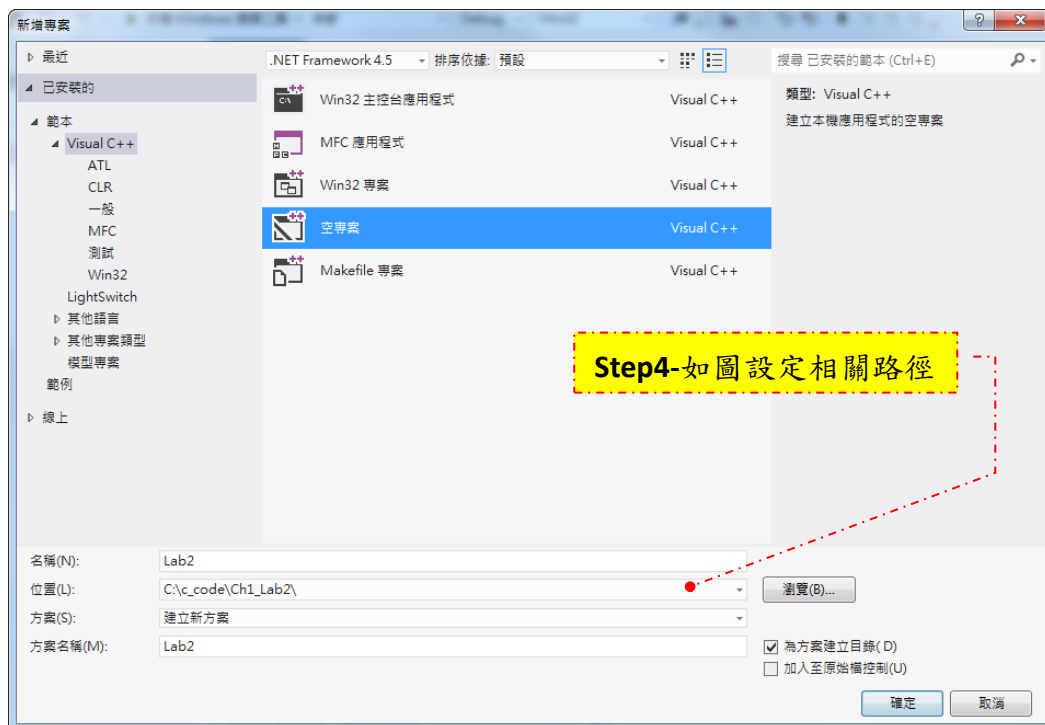
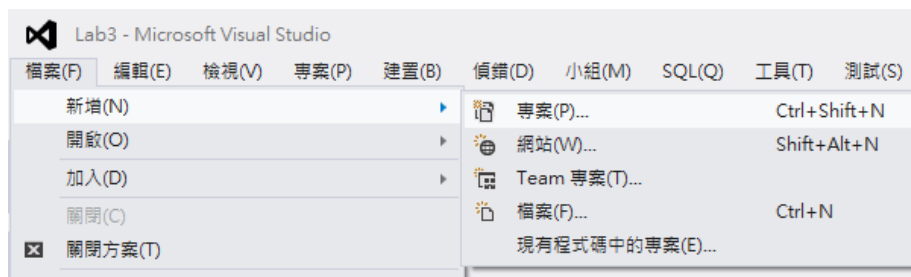
方法 B. 透過 Visual Studio 新建專案

Step1-在 C:\c_code 資料夾內新增名為“Ch1_Lab2”的資料夾，再於 Ch1_Lab2 資料夾內分別建立 include、source、temp 等資料夾，建立後需要在 temp 資料夾內新增名為“debug”的資料夾，建立完成後如下圖

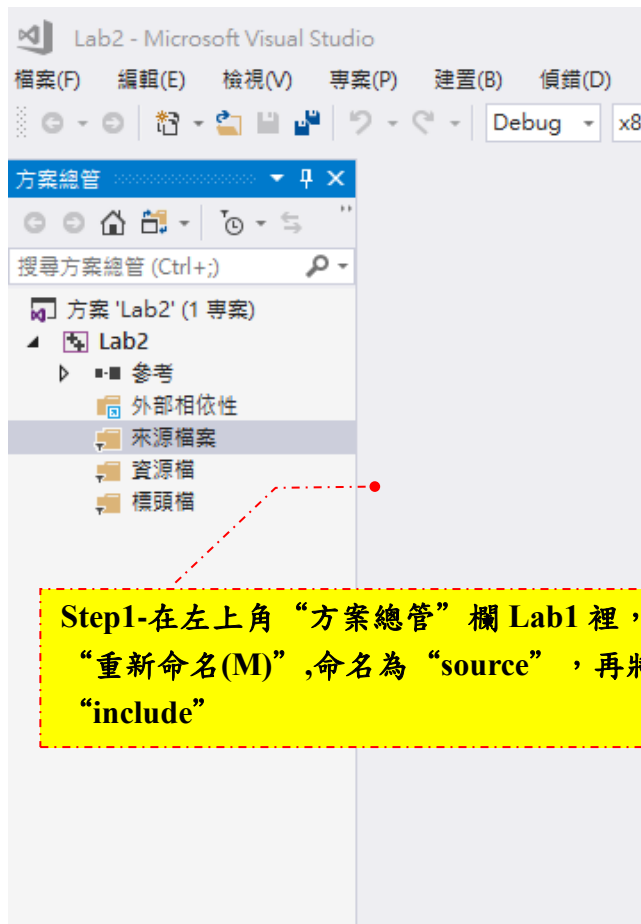


Step2-開啟 Microsoft Visual Studio 視窗畫面後點選左上角“開始(F)”

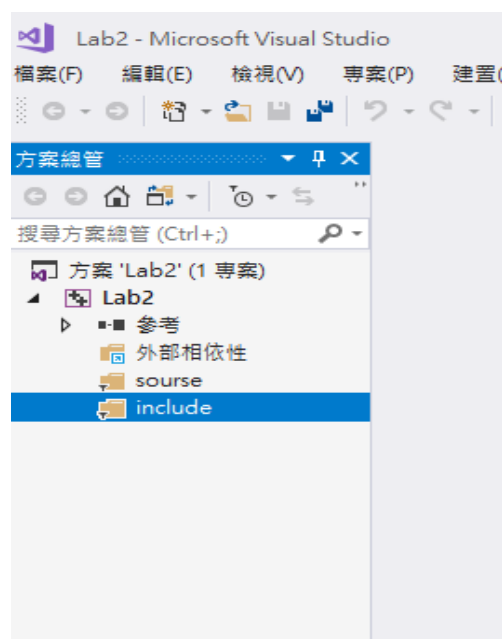
Step3-點選“新增(N)”再點選“專案(P)”

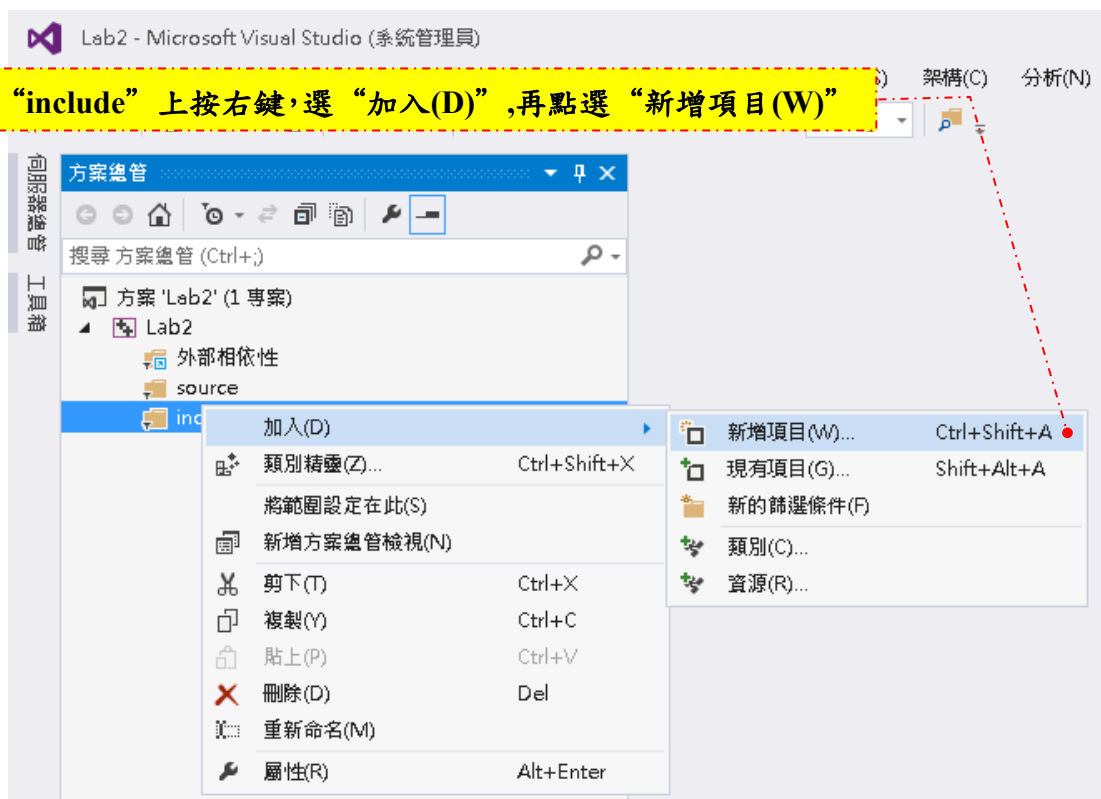
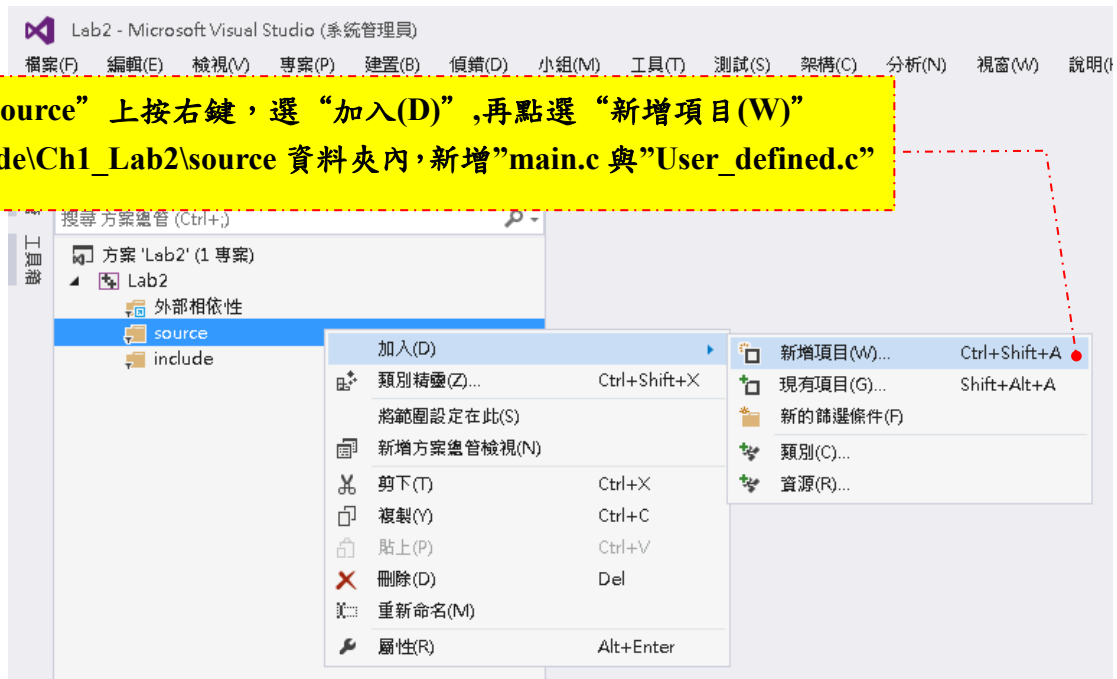


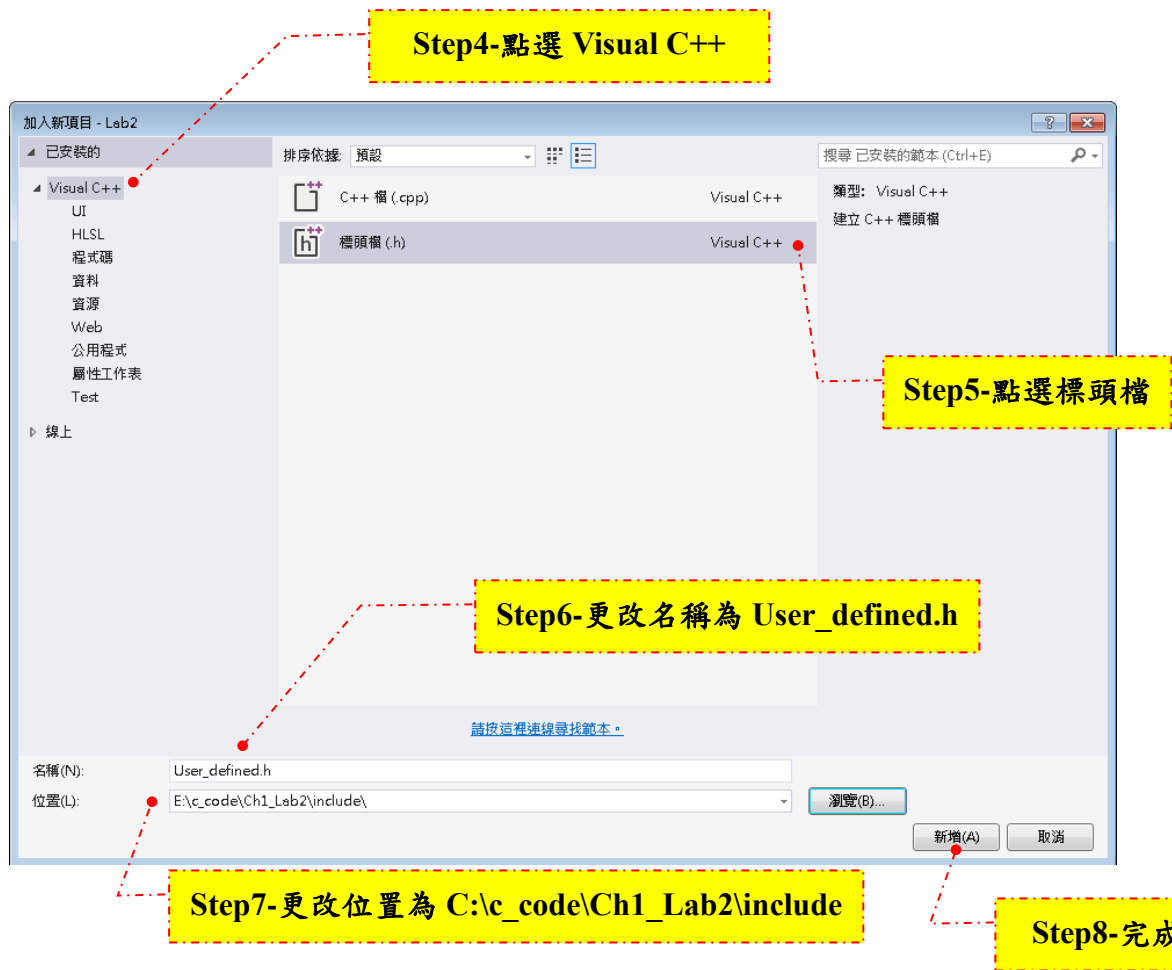
2. 新增 .c 檔、路徑設定

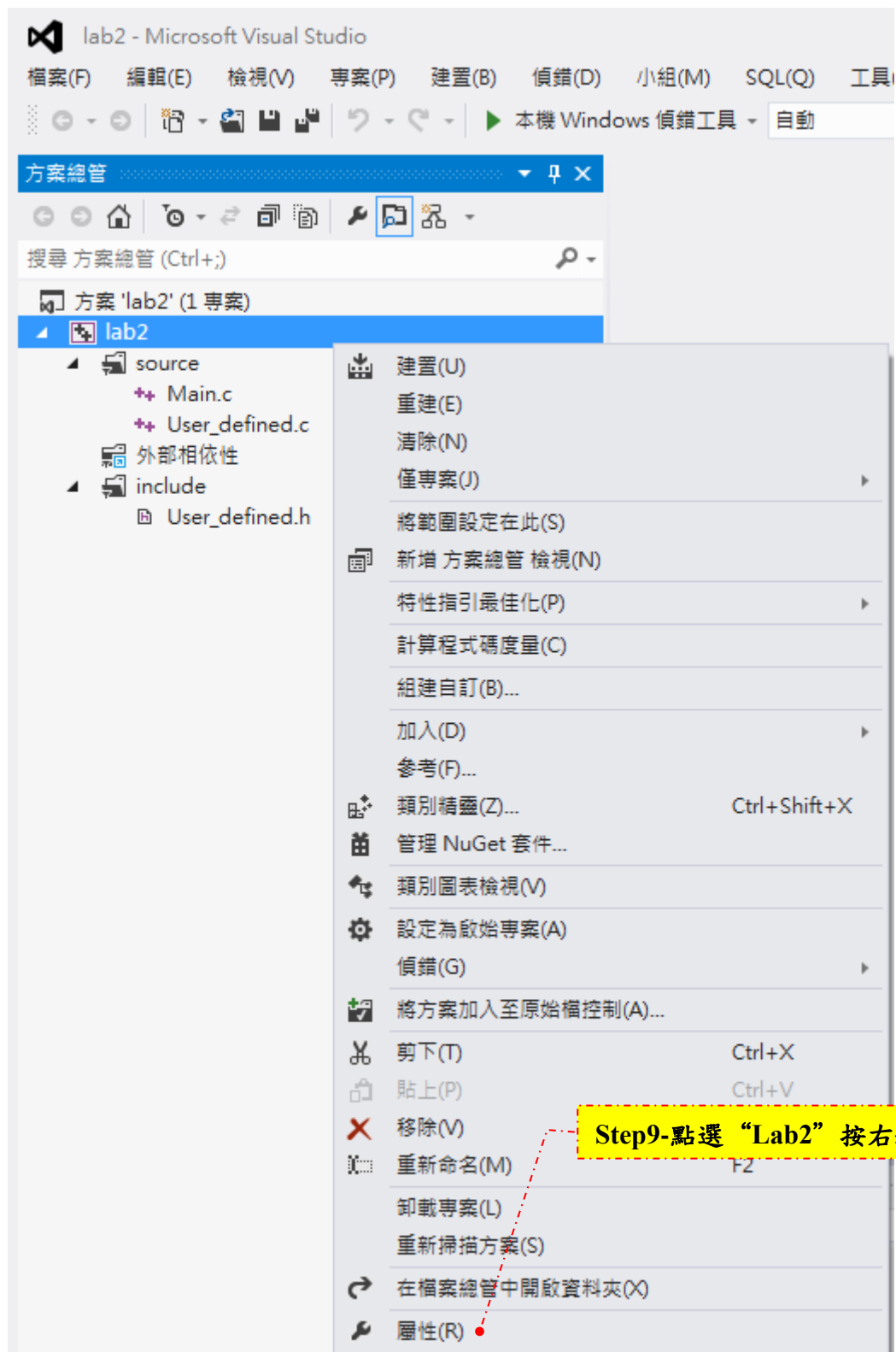


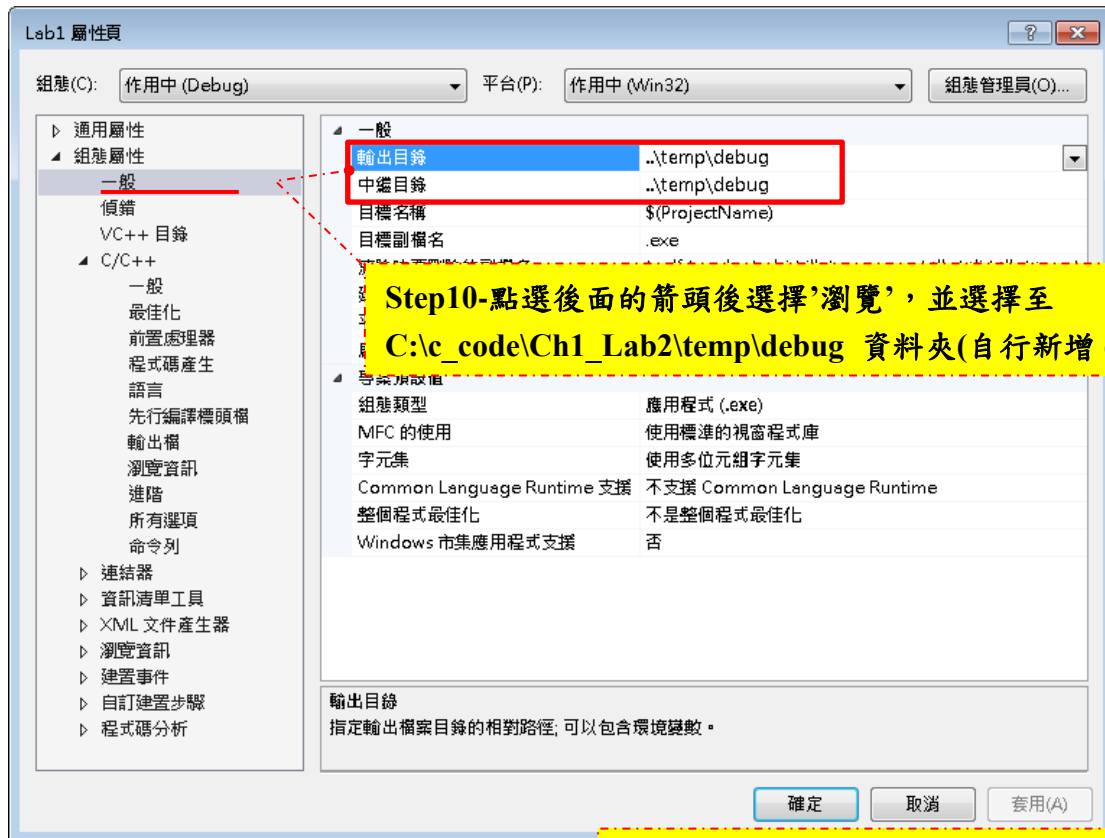
點選“資源檔”，按右鍵選“刪除(D)”，完成後方案總管如下圖



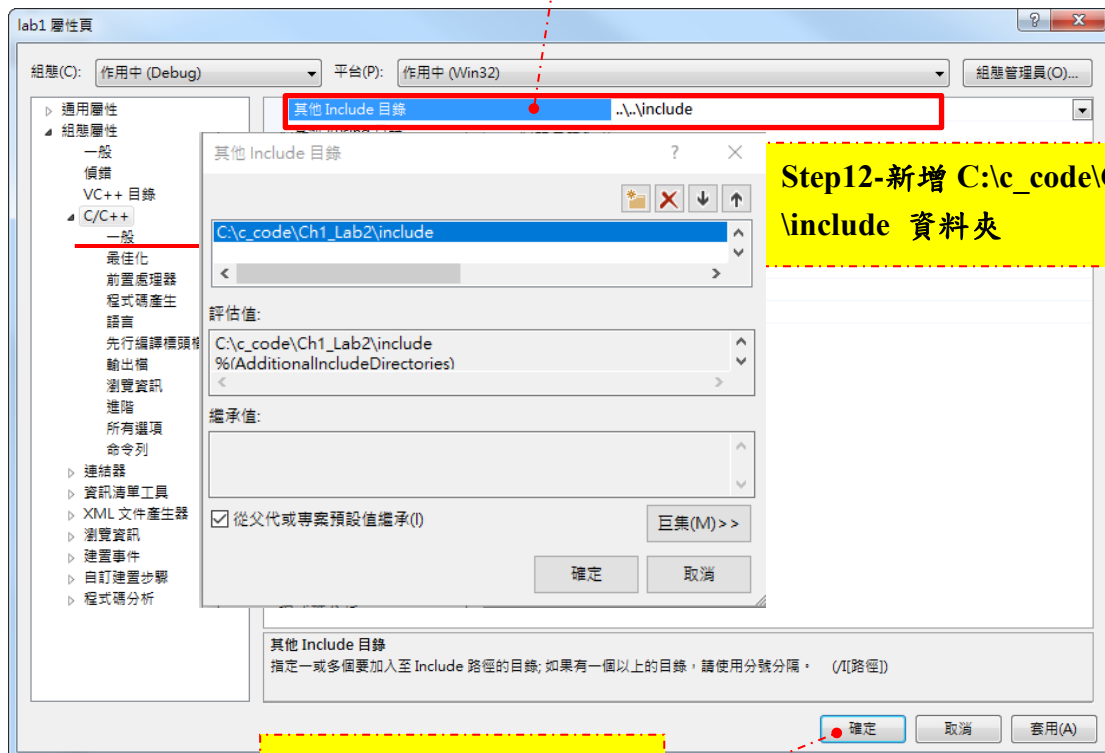








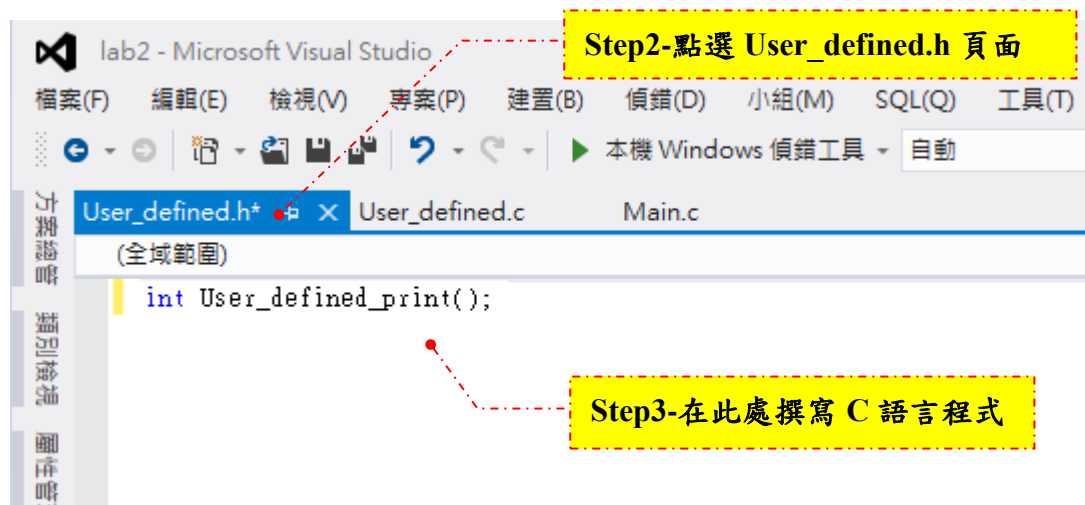
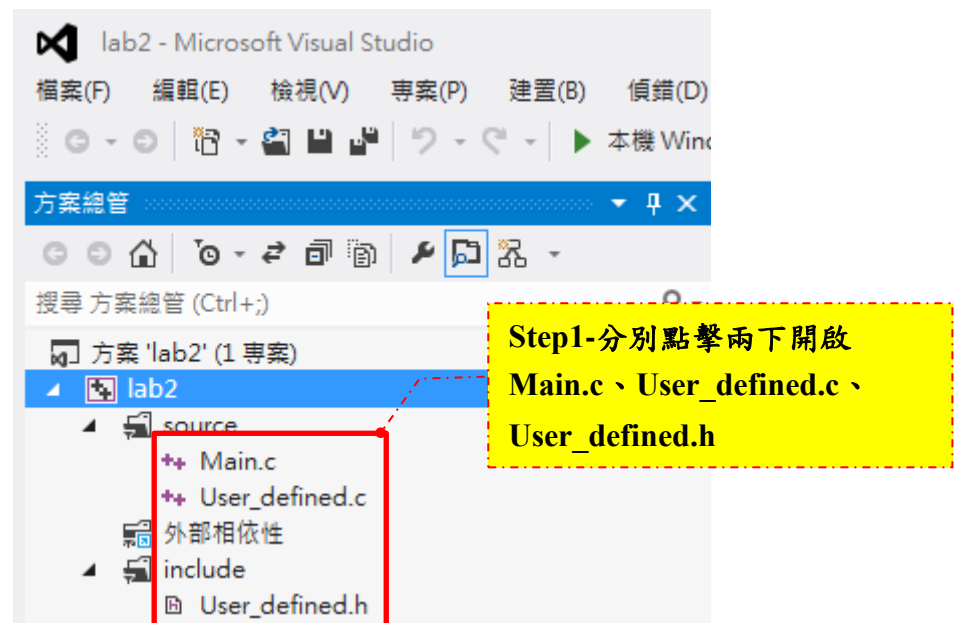
Step11-點選後面的箭頭選擇編輯



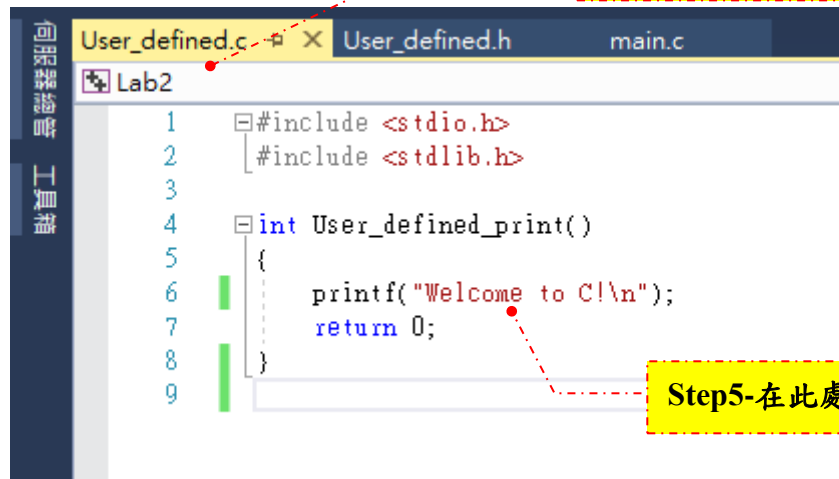
Step13-修改完成後按確定

3. 撰寫 C 語言程式

如果透過「方法 A. 透過 Github Classroom 下載並開啟專案」建立專案，直接從此步驟繼續實驗。



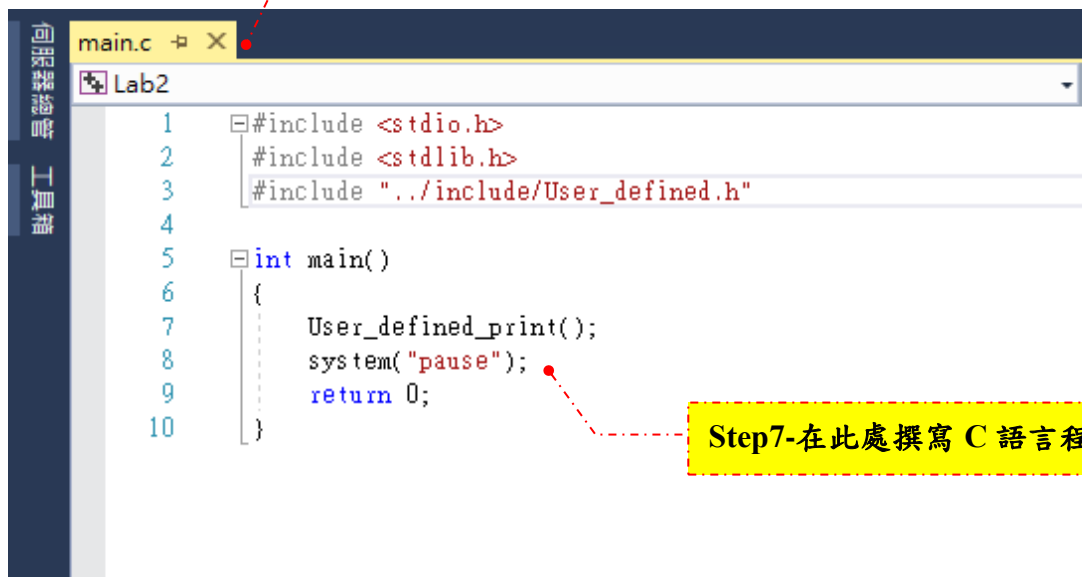
Step4-點選 User_defined.c 頁面



```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int User_defined_print()
5  {
6      printf("Welcome to C!\n");
7      return 0;
8  }
```

Step5-在此處撰寫 C 語言程式

Step6-點選 Main.c 頁面

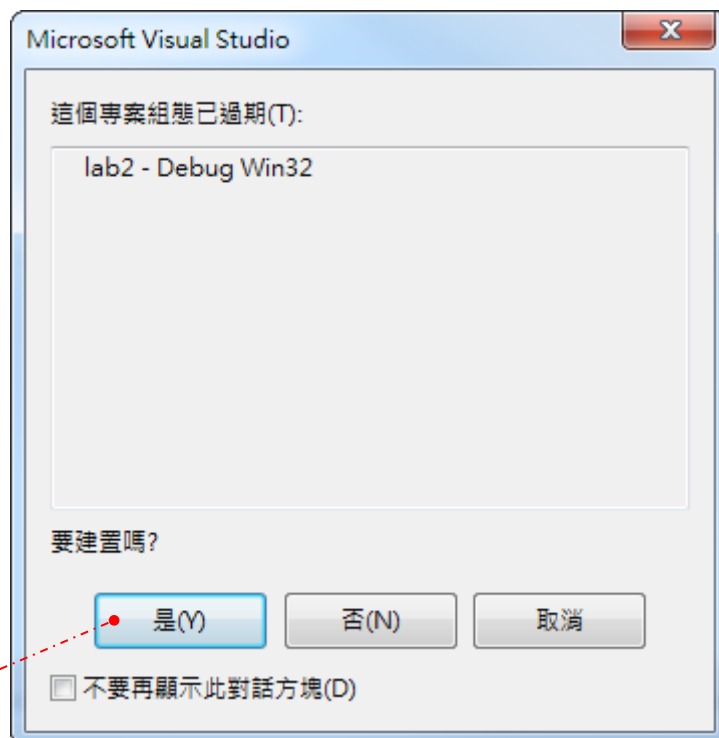
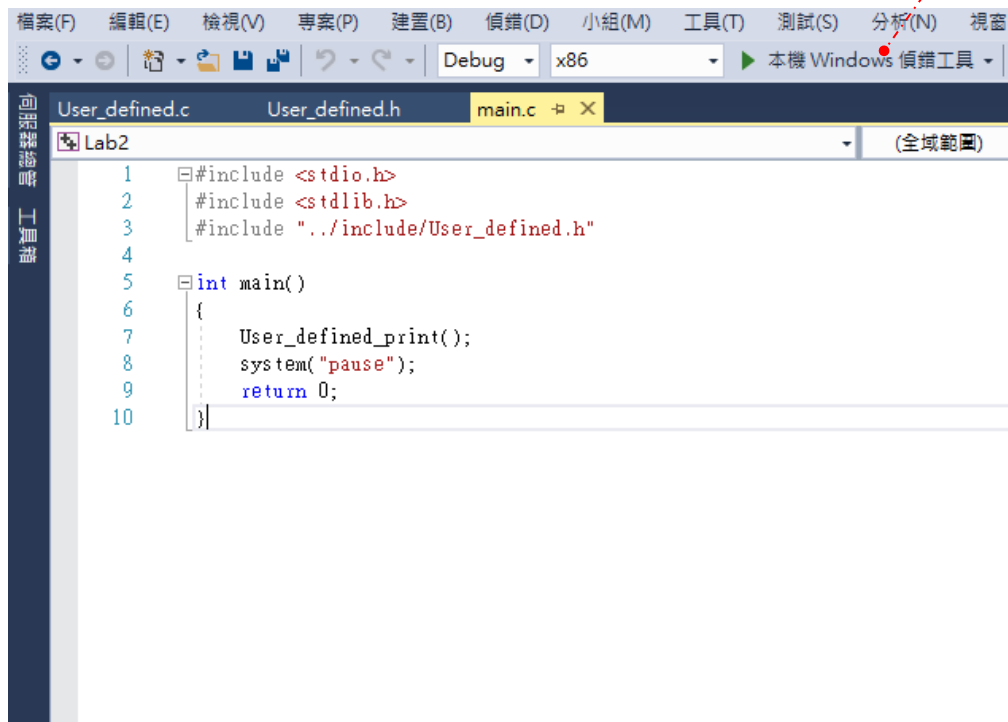


```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "../include/User_defined.h"
4
5  int main()
6  {
7      User_defined_print();
8      system("pause");
9      return 0;
10 }
```

Step7-在此處撰寫 C 語言程式

4. 執行與測試程式結果

Step1-點選開始偵測，進行偵測



Step2-如出現此視窗請按“是”



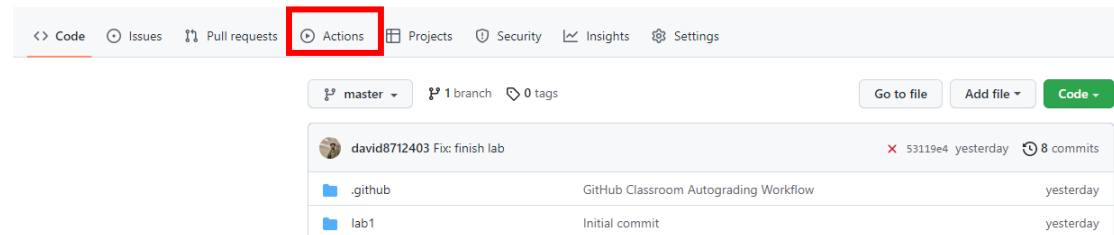
5. 上傳實驗至 Github Classroom

請參考從 Github Clone 專案中 README.md 檔案的**上傳專案說明**，將專案透過 Git 指令 push 到 Github classroom

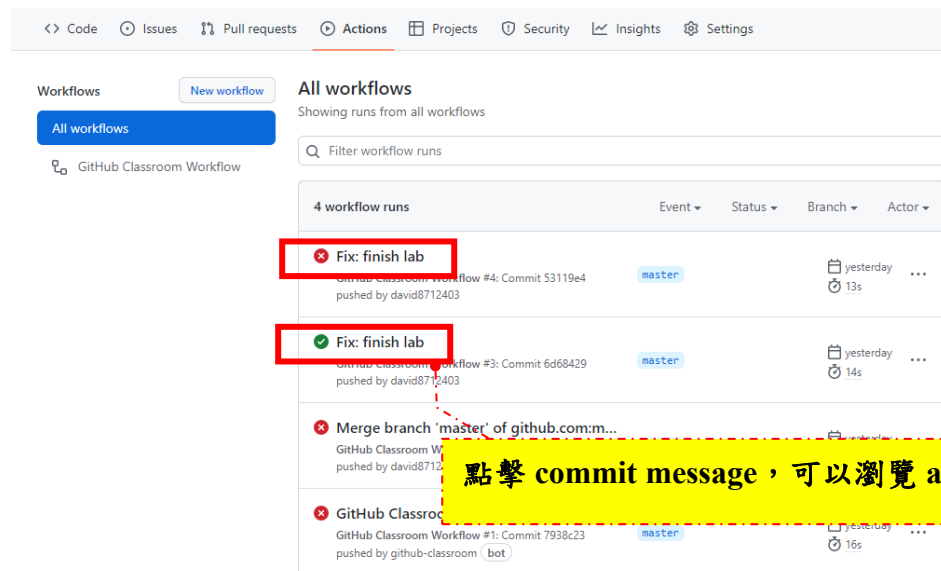
6. 觀察 Github Action 評分

在每次有新的 commit push 到 Github Classroom 時，會觸發定義好的 action 流程，會自動將程式編譯後執行，判斷是否執行正確。

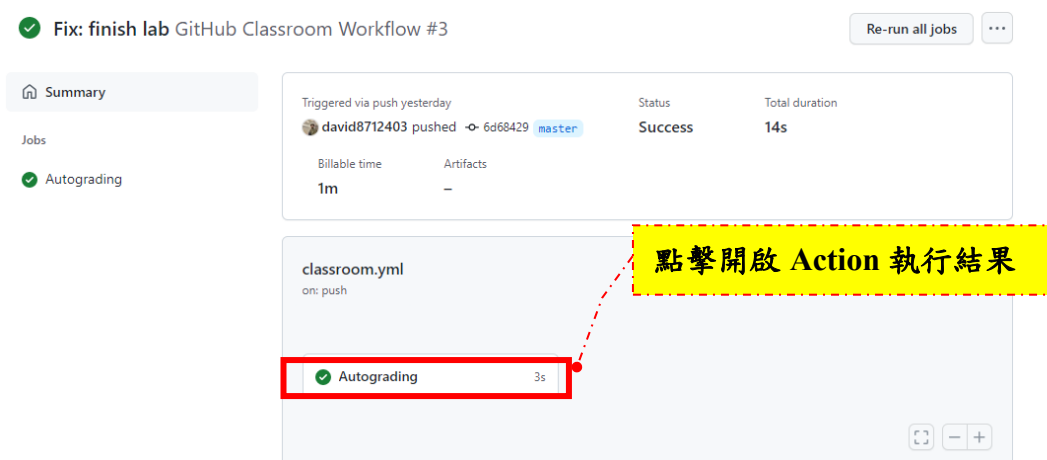
Step1. 進入 Github classroom 實驗的 repository，點擊 Action



Step2. 觀察自己每次 commit 時，action 的輸出及批改結果



Step3. 瀏覽 Action 執行結果



自動評分執行成功情況

✅ Fix: finish lab GitHub Classroom Workflow #3 Re-run all jobs ...

Summary

Jobs

✅ Autograding

Autograding
succeeded yesterday in 3s

Search logs

> Set up job

> Run actions/checkout@v2 0s

Run education/autograding@v1 0s

```
1 ▶ Run education/autograding@v1
4
5 ::stop-commands::***
6
7 test
8
9
10 gcc source/Main.c
11 bash test.sh
12 sh: 1: pause: not found
13 Running tests...
14
15 Pass: Program exited zero
16 Output:
17 welcome to c
18 Pass: Output is correct
19
20 All tests passed.
21
22 ✅ test
23
```

展開 Run education/autograding@v1

編譯及輸出結果

自動評分執行失敗情況

❌ GitHub Classroom Autograding Workflow GitHub Classroom Workflow #1 Re-run jobs ...

Summary

Jobs

❌ Autograding

Autograding
failed 1 hour ago in 5s

Search logs

> Set up job

> Run actions/checkout@v2 1s

Run education/autograding@v1 0s

```
1 ▶ Run education/autograding@v1
4
5 ::stop-commands::***
6
7 test
8
9
10 gcc source/Main.c
11 bash test.sh
12 Running tests...
13
14 sh: 1: pause: not found
15 Pass: Program exited zero
16 Output:
17 Expected 'Welcome to C!' but got: ''
18 make: *** [Makefile:5: test] Error 1
19
20
21 ❌ test
22 ::error::Error: Exit with code: 2 and signal: null
23
24 ::***::
25
```

展開 Run education/autograding@v1

編譯及輸出結果

失敗原因，輸出內容與預期不符
預期輸出: Welcome to C!
實際輸出: (空字串)