

## 實驗項目 - 製作階層函式(factorial 函式)

### 一、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

### 二、設計重點：

- C 語言的函式應用

### 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. 開啟專案檔案

名稱	修改日期	類型	大小
ch1-lab-template	2022/9/28 下午 0...	檔案資料夾	
ch2-lab-template	2022/10/6 下午 0...	檔案資料夾	
ch3-lab-template	2022/10/20 下午 ...	檔案資料夾	

開啟專案資料夾

資料夾名稱會是 ch3-lab-{你的 Github 帳號 ID}

名稱	修改日期	類型	大小
.git	2022/10/20 下午 ...	檔案資料夾	
ch3-lab1	2022/10/20 下午 ...	檔案資料夾	
ch3-lab2-1	2022/10/19 下午 ...	檔案資料夾	
ch3-lab2-2	2022/10/19 下午 ...	檔案資料夾	
.gitignore	2022/10/19 下午 ...	文	
a.out	2022/10/20 下午 ...	Out	
lab1.sh	2022/10/19 下午 ...	Shell Script	5 KB
lab2.sh	2022/10/20 下午 ...	Shell Script	2 KB
Makefile	2022/10/20 下午 ...	檔案	1 KB
README.md	2022/10/19 下午 ...	Markdown 來源...	2 KB
test-template.sh	2022/10/19 下午 ...	Shell Script	1 KB

開啟 ch3-lab2-1 資料夾

名稱	修改日期	類型	大小
Lab2-1	2022/10/19 下午 ...	資料夾	
source	2022/10/20 下午 ...	資料夾	
.gitignore	2022/10/19 下午 ...	文字文件	5 KB

開啟 Lab2-1 資料

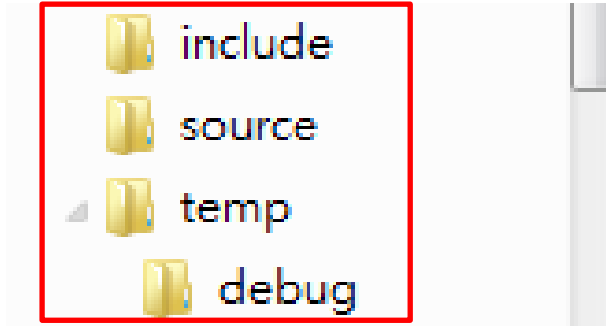
名稱	修改日期	類型	大小
.vs	2022/10/19 下午 ...	檔案資料夾	
Lab2-1	2022/10/19 下午 ...	檔案資料夾	
x64	2022/10/19 下午 ...	檔案資料夾	
Lab2-1.sln	2022/10/19 下午 ...	Visual Studio Sol...	2 KB

點擊 Lab2-1.sln 專案檔

**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 “Ch3\_Lab2-1” 的資料夾，再於 Ch3\_Lab2-1 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



Step2-參照 Ch1\_Lab3 中 “1.建立新的空專案” Step2~Step4，設定相關路徑位置為 C:\c\_code\ Ch3\_Lab2-1

### 2. 路徑設定、新增 .c 檔

Step1-參照 Ch1\_Lab3 中 “2. 路徑設定、新增 .c 檔” Step1~Step8，新增 main.c 檔與設定相關屬性設定。

### 3. 撰寫 C 語言程式

Step2-於 main.c 頁面下撰寫程式

Step3-在此處撰寫 C 語言程式

Step1-點擊兩下開啟 main.c

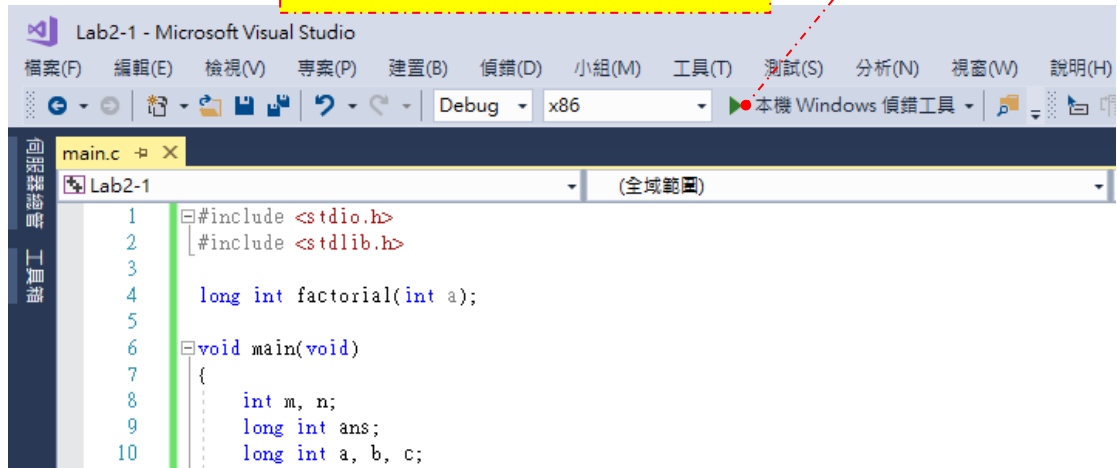
```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  long int factorial(int a);
5
6  void main(void)
7  {
8      int m, n;
9      long int ans;
10     long int a, b, c;
11
12     printf("求排列組合C(m,n)\n");
13     printf("m=");
14     scanf_s("%d", &m);
15     printf("n=");
16     scanf_s("%d", &n);
17
18     ans = a/(b*c);
19     printf("C(%d,%d)=%d\n", m, n, ans);
20
21     system("pause");
22 }
23
24 long int factorial(int p)
25 {
26     int count;
27     long int result = 1;
28
29     for (count = 1; count <= p; count++)
30     {
31         result = result * count;
32     }
33     return result;
34 }
```

main.c 程式碼：

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  long int factorial(int);
5
6  int main(void)
7  {
8
9      printf("求排列組合C(m,n)\n");
10
11     int m, n;
12     long int ans;
13     long int a, b, c;
14     printf("m=\n");
15     scanf("%d", &m);
16     printf("n=\n");
17     scanf("%d", &n);
18
19     a = factorial(m);
20     b = factorial(n);
21     c = factorial(m - n);
22
23     ans = a / (b*c);
24     printf("C(%d,%d)=%ld\n", m, n, ans);
25     system("pause");
26     return 0;
27 }
28
29 long int factorial(int p)
30 {
31     int count;
32     long int result = 1;
33     for (count = 1; count <= p; count++)
34         result *= count;
35     return result;
36 }
37
```

#### 4. 執行與測試程式結果

Step1-點選開始偵測，進行偵測



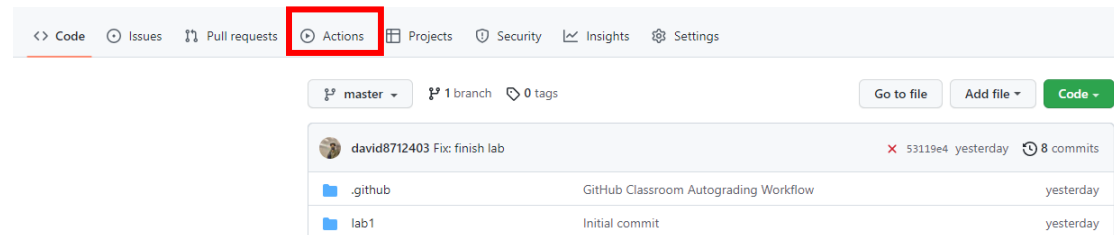
## 5. 上傳實驗至 Github Classroom

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

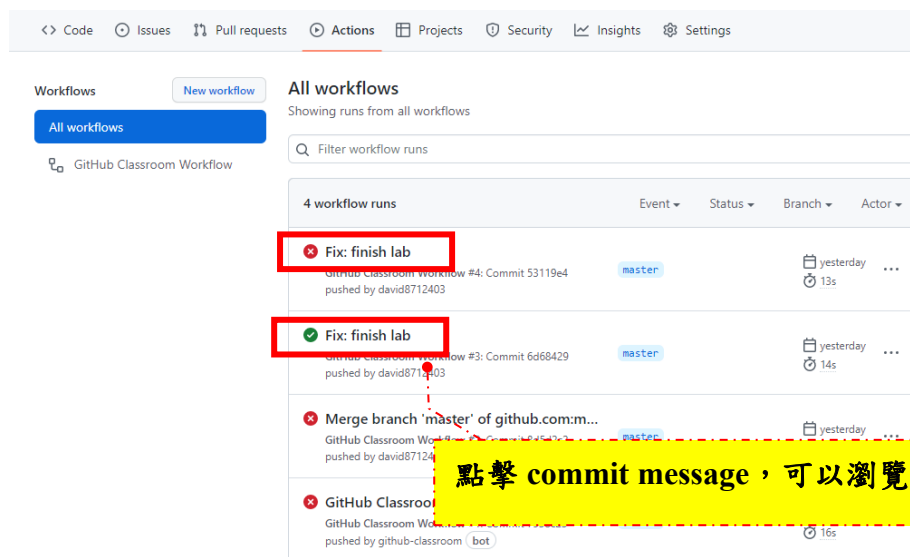
## 6. 觀察 Github Action 評分

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。

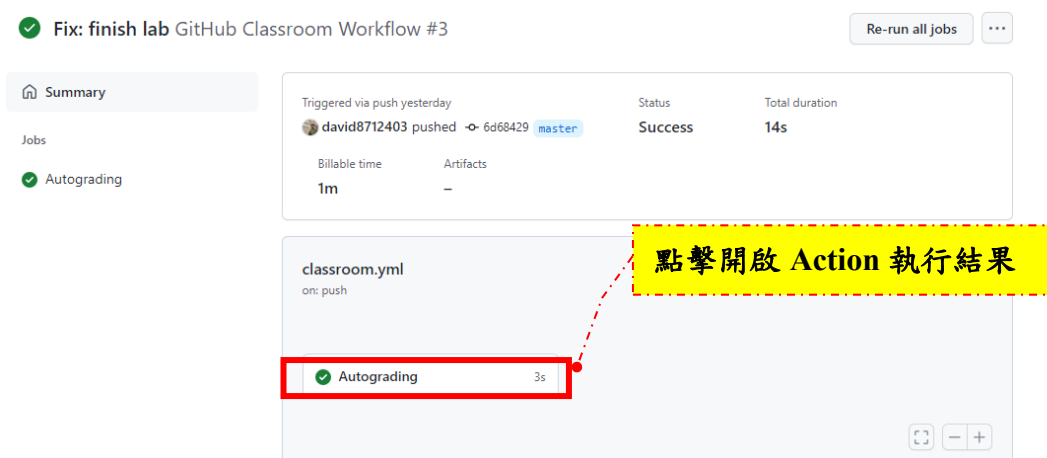
Step1. 進入 Github classroom 實驗的 repository，點擊 Action



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。本次 Lab 有 3 個 test case，全數通過會得到 100 分

succeeded 1 hour ago in 11s

**展開 Run education/autograding@v1**

Run education/autograding@v1

```

38 the relationships they satisfy:
39 10 is not equal to 20
40 10 is less to 20
41 10 is less than or equal to 20
42 Pass: Output is correct
43
44 All tests passed.
45
46  test2
47
48  test3
49
50
51 bash test3.sh
52 Running tests...
53
54 sh: 1: pause: not found
55 Pass: Program exited zero
56 Output:
57 Enter two integers, and I will tell you
58 the relationships they satisfy:
59 40 is not equal to 20
60 40 is greater to 20
61 40 is greater than or equal to 20
62 Pass: Output is correct
63
64 All tests passed.
65
66  test3
67
68
69 ::***::
70
71 All tests passed
72
73
74
75 Points 100/100

```

## 編譯及輸出結果