

下載範例檔案

- https://github.com/ck1001099/cpp_course_2019summer
- lesson10 -> Lecture -> Object_template.cpp

C++程式設計基礎

lesson 10

陳毅

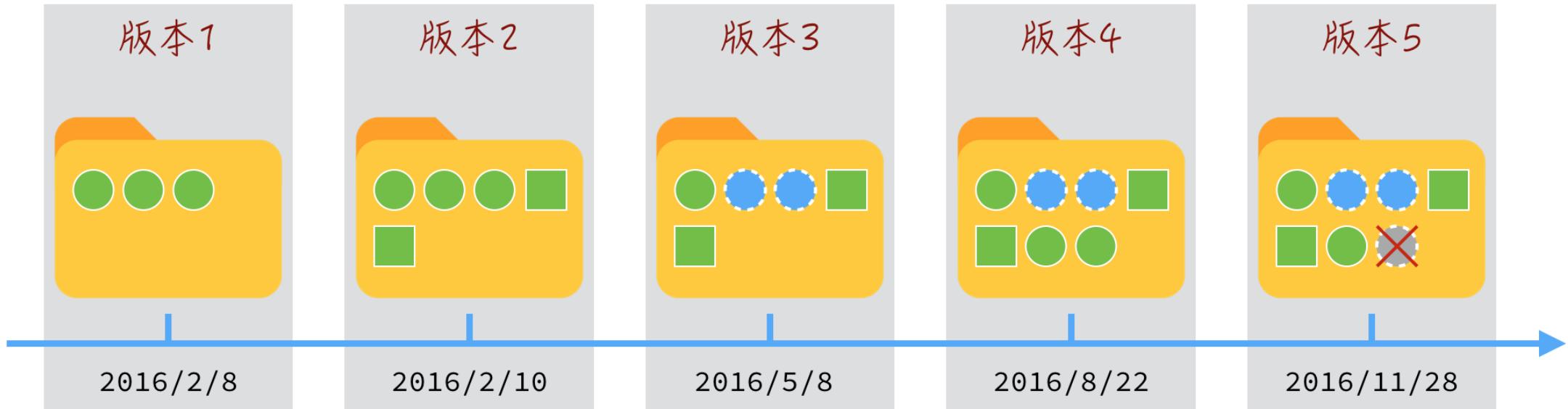
本週概要

- 專案管理：Sourcetree + GitHub
- 物件導向補充 – 靜態成員(static member)
- 多載函數（Ch12）
- 進階演算法
 - 更多的排序演算法：Insertion sort、Quick sort
- 其他程式語言簡介
- 課程回顧

專案管理：Sourcetree + GitHub

Git是什麼？

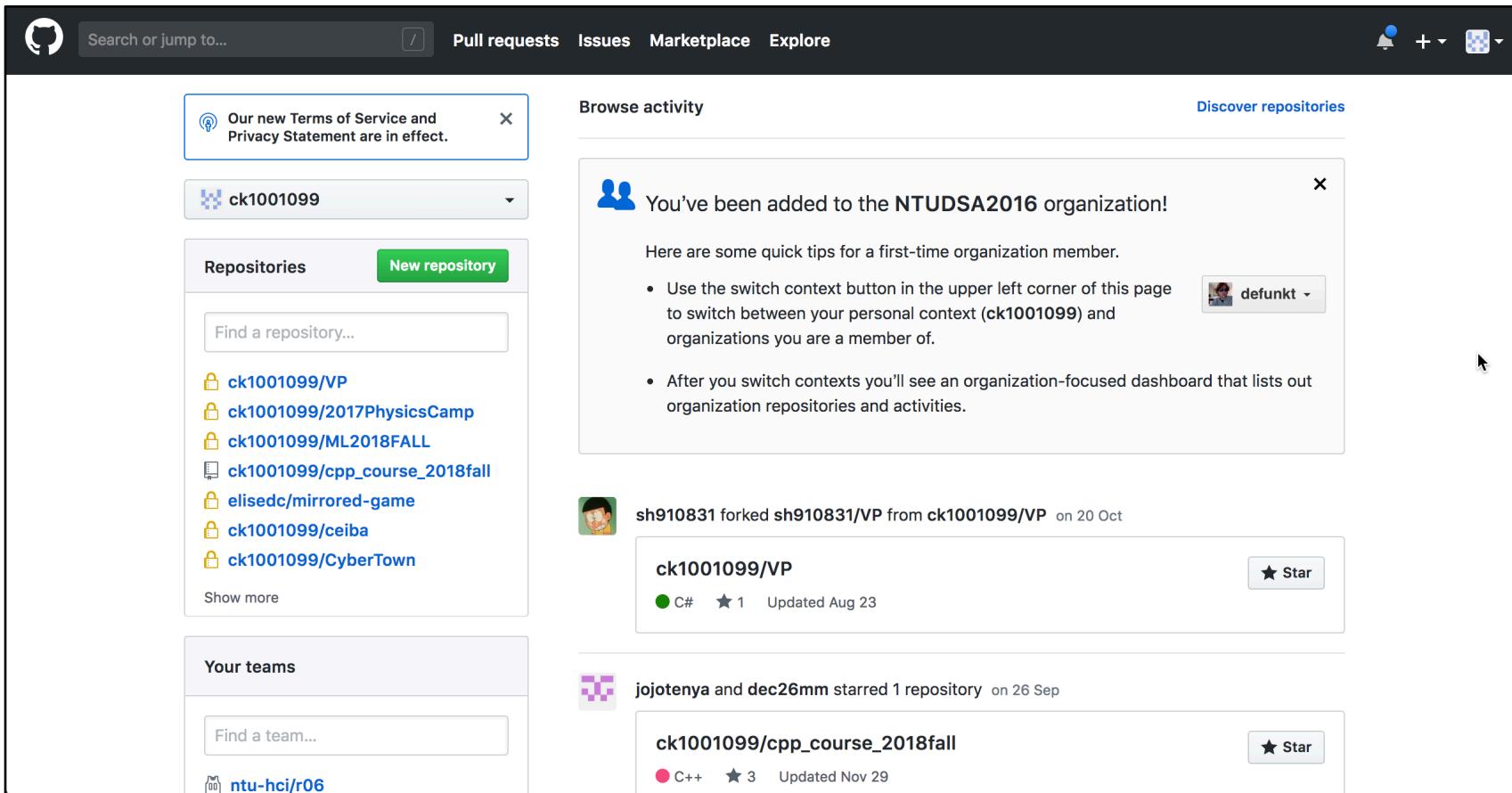
- Git是一種分散式版本的**版本控制系統(Version Control System)**。



- 常見的線上版本控制平台：GitHub、BitBucket等。

GitHub 介紹

- GitHub: <https://github.com/>



Sourcetree 介紹

- Sourcetree: <https://www.sourcetreeapp.com/>

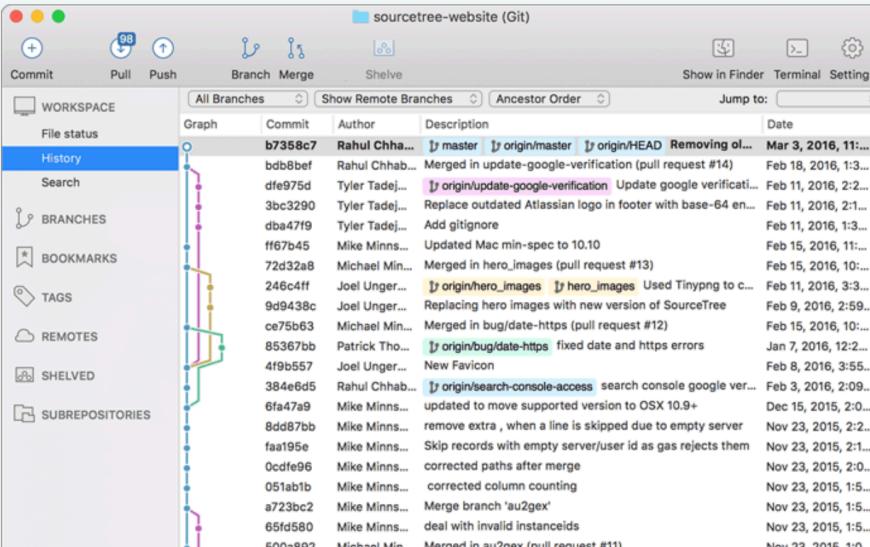
 Sourcetree

[Download free](#)

Simplicity and power in a beautiful Git GUI

[Download for Mac OS X](#)

Also available for Windows

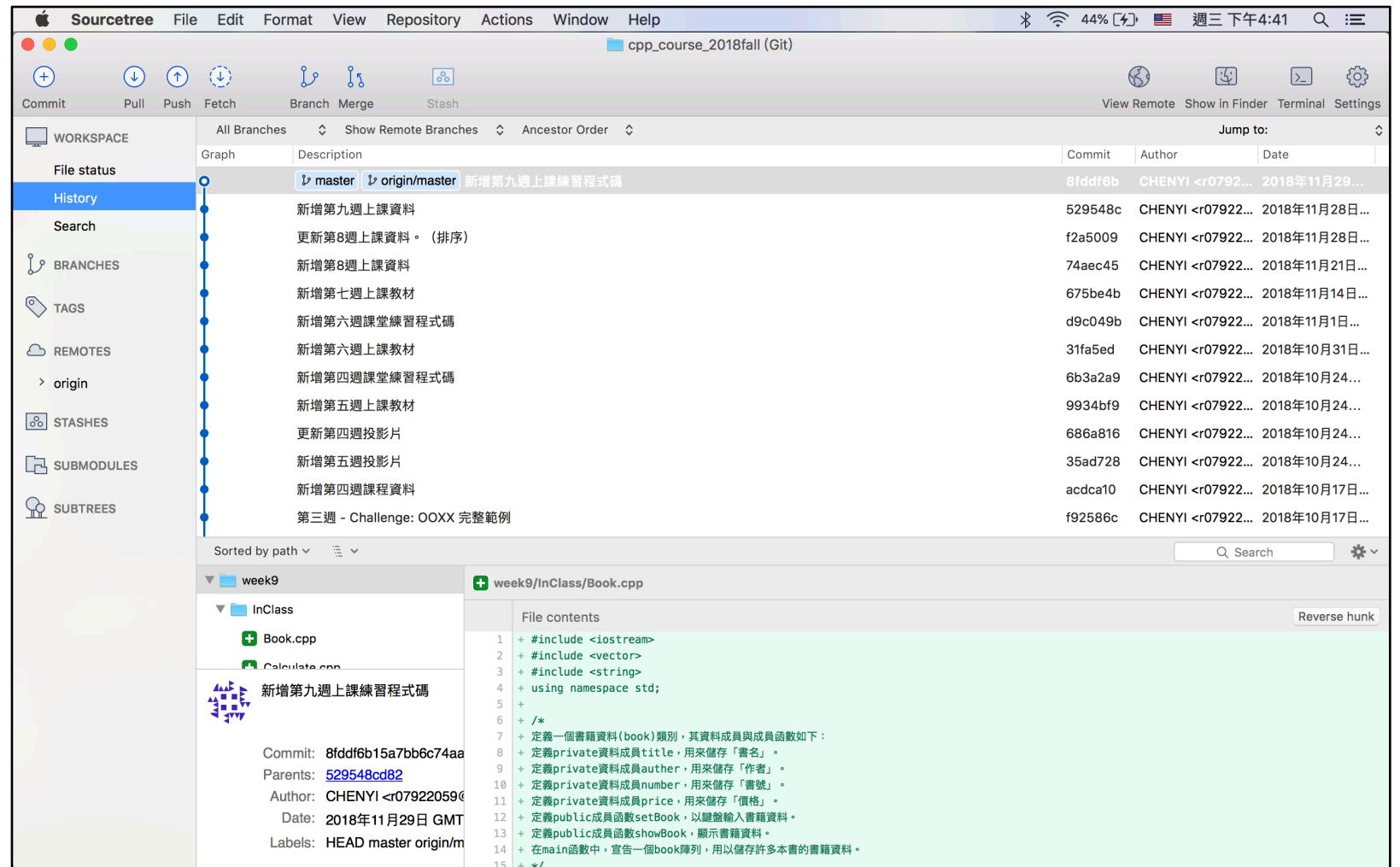
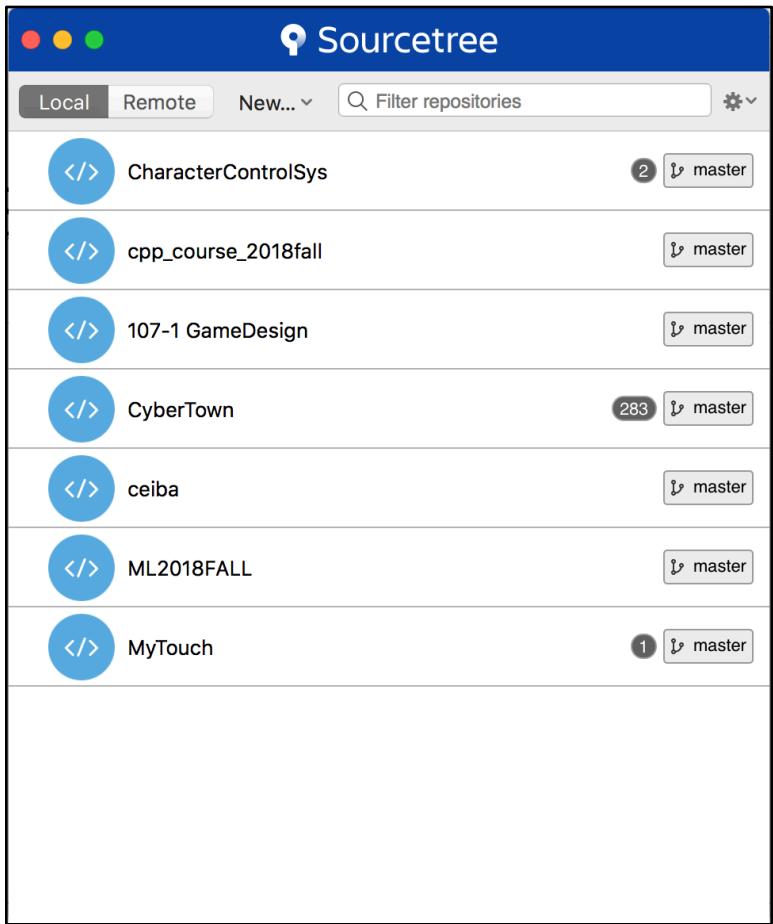


Commit	Author	Description	Date
b7358c7	Rahul Chhab...	master origin/master origin/HEAD Removing ol...	Mar 3, 2016, 11...
bdb8bf	Rahul Chhab...	Merged in update-google-verification (pull request #14)	Feb 18, 2016, 1:3...
dfe975d	Tyler Tadej...	origin/update-google-verification Update google verificati...	Feb 11, 2016, 2:2...
3bc3290	Tyler Tadej...	Replace outdated Atlassian logo in footer with base-64 en...	Feb 11, 2016, 2:1...
dba47f9	Tyler Tadej...	Add gitignore	Feb 11, 2016, 1:3...
ff67b45	Mike Minns...	Updated Mac min-spec to 10.10	Feb 15, 2016, 11...
72d32a8	Michael Min...	Merged in hero_images (pull request #13)	Feb 15, 2016, 10...
246c4ff	Joel Unger...	origin/hero_images hero_images Used TinyPNG to c...	Feb 11, 2016, 3:3...
9d9438c	Joel Unger...	Replacing hero images with new version of SourceTree	Feb 9, 2016, 2:59...
c675b63	Michael Min...	Merged in bug/date-https (pull request #12)	Feb 15, 2016, 10...
85367bb	Patrick Tho...	origin/bug/date-https fixed date and https errors	Jan 7, 2016, 12:2...
419b557	Joel Unger...	New Favicon	Feb 8, 2016, 3:55...
384e6d5	Rahul Chhab...	origin/search-console-access search console google ver...	Feb 3, 2016, 2:09...
6fa47a9	Mike Minns...	updated to move supported version to OSX 10.9+	Dec 15, 2015, 2:0...
8dd87bb	Mike Minns...	remove extra , when a line is skipped due to empty server	Nov 23, 2015, 2:2...
faf195e	Mike Minns...	Skip records with empty server/user id as gas rejects them	Nov 23, 2015, 2:1...
0cde96	Mike Minns...	corrected paths after merge	Nov 23, 2015, 2:0...
051ab1b	Mike Minns...	corrected column counting	Nov 23, 2015, 1:5...
a723bc2	Mike Minns...	Merge branch 'au2gex'	Nov 23, 2015, 1:5...
65fd580	Mike Minns...	deal with invalid instanceids	Nov 23, 2015, 1:5...
500a892	Michael Min...	Merged in au2gex (pull request #11)	Nov 23, 2015, 1:0...

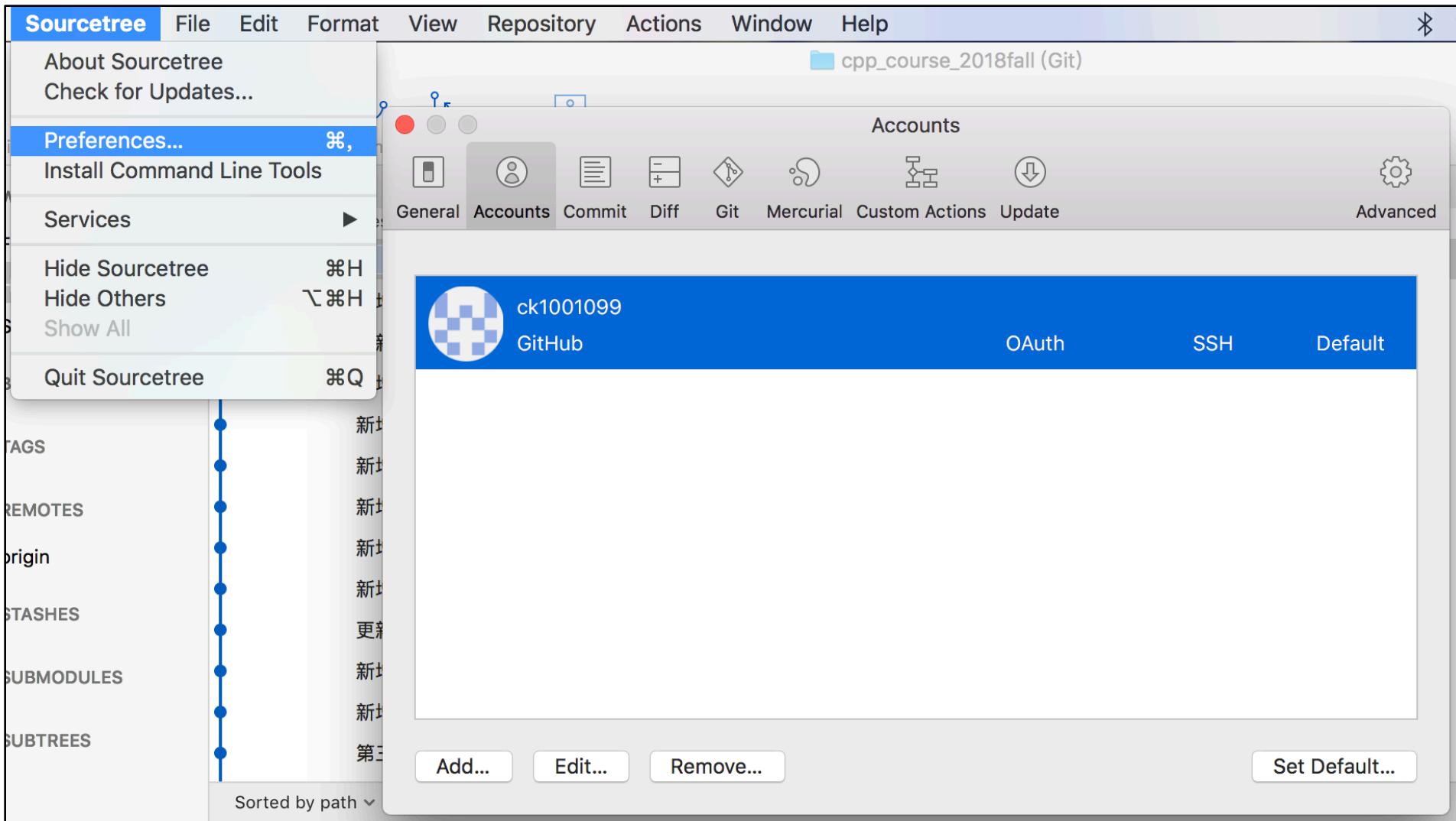
A free Git client for Windows and Mac

Sourcetree simplifies how you interact with your Git repositories so you can focus on coding. Visualize and

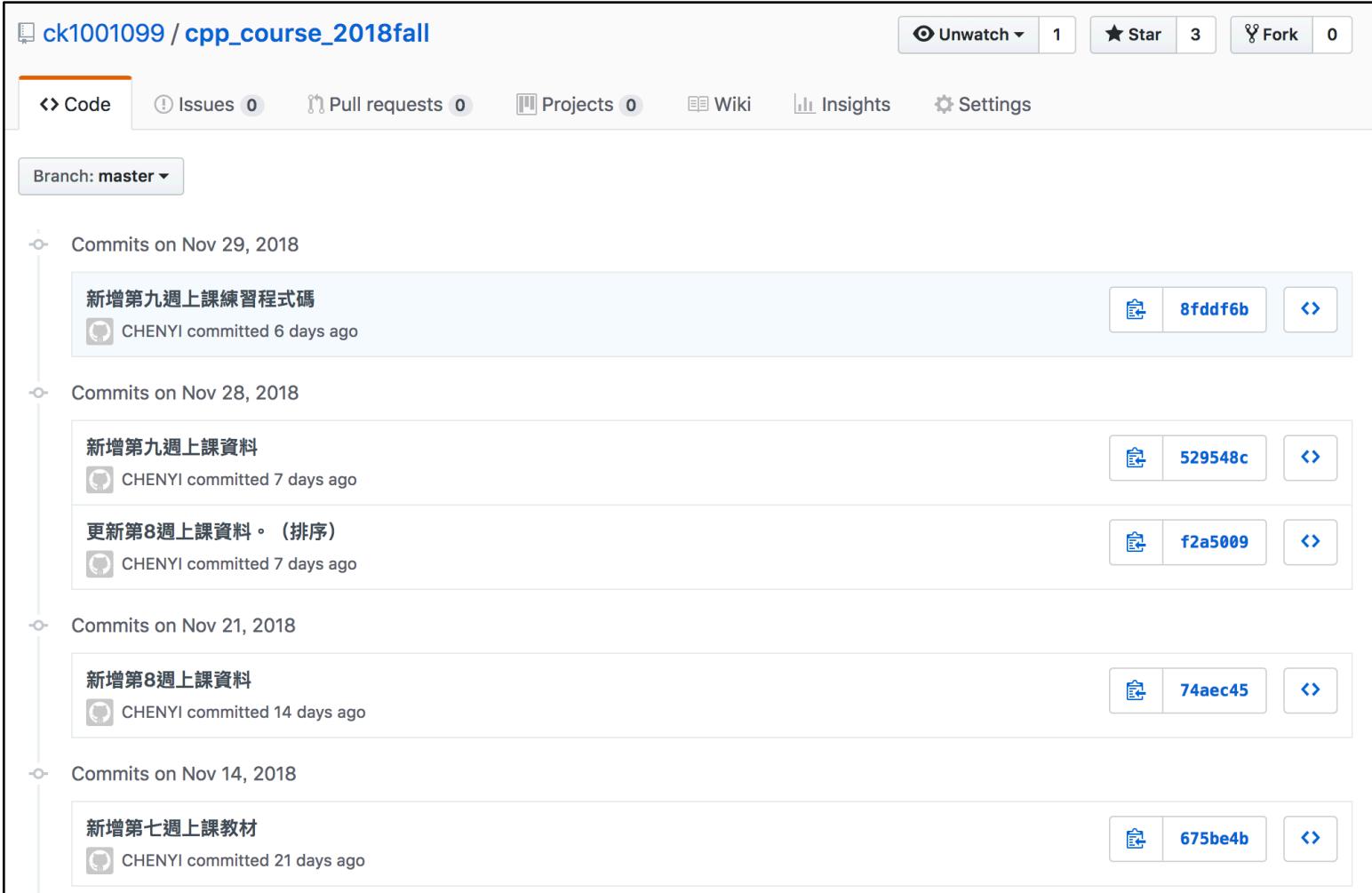
Sourcetree + GitHub



Sourcetree + GitHub



Sourcetree + GitHub



補充資料

- 30 天精通 Git 版本控管
<https://github.com/doggy8088/Learn-Git-in-30-days>
- GitHub 版本控制
<https://hackmd.io/s/Bk2AaU6o>
- Git 的基本介紹
https://backlog.com/git-tutorial/tw/intro/intro1_1.html
- 一步一步教你用 GitHub + SourceTree 做版本控制
<https://hackmd.io/s/Bymbf3y2>

物件導向補充

靜態成員(static member)

- 若某個資料成員被宣告為static，則其他同類別的物件皆可共享該靜態成員的資料。
- 若某個成員函數被宣告為static，則該static成員函數，不可呼叫非static的成員函數。
- 使用方法：宣告的型態前，加上「static」。

static 資料型態 變數名稱；

static 函數型態 **函數名稱(參數列){ 敘述區； }**

靜態成員(static member)

- 靜態的資料成員，必須要進行初始化，而且也只能被初始化一次。

```
class Test{
public:
    Test(int k){ this->k = k; }
    void Add1(){ number = number + 1; }
    int Get(){ return number+k; }
private:
    static int number;
    int k;
};

int Test::number = 0;
```

練習

- 定義一個物體(Object)類別，其資料成員與成員函數如下：
 - 定義一個private資料成員name，儲存該物體名稱。
 - 定義一個private資料成員index，儲存該物體編號。
 - 定義一個static private資料成員count，記錄已經建立多少個物體。
 - 定義一個static private資料成員objList，用於儲存被建立的物體的指標。
 - 定義建立者(constructor)函數，初始化name，將count值加1，並將指標存入objList。
 - 定義破壞者(destructor)函數，將count值減1，並根據編號從objList中移除指標。
 - 定義一個static public成員函數info，輸出總物體數目以及各物體編號及名稱。

多載函數

什麼是多載(overloading)？

- 四則運算也是函數的一種。
- $3+5$ ，實際上就是 *int operator+ (3, 5)*
- $6.3-1.8$ ，實際上就是 *double operator- (6.3, 1.8)*

什麼是多載(overloading)？

- 關於加法，加數與被加數有很多種組合。
- 整數 + 整數 : $3+5, 62+10, \dots$
- 整數 + 小數 : $7+1.3, 6+3.14, \dots$
- 小數 + 整數 : $5.32+13, 872.24+524, \dots$
- 小數 + 小數 : $34.43+235.73, 1.548+745.231, \dots$
- 那加法的函數雛形到底長什麼樣子呢？

什麼是多載(overloading)？

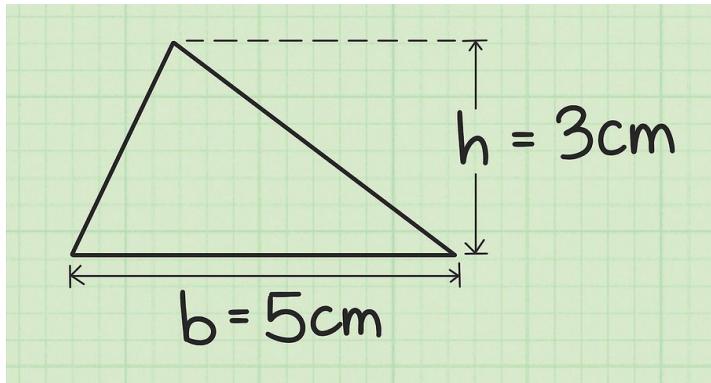
- Ans：加法的函數雛形其實有很多個！
- 整數 + 整數 : int operator+ (int, int)
- 整數 + 小數 : double operator+ (int, double)
- 小數 + 整數 : double operator+ (double, int)
- 小數 + 小數 : double operator+ (double, double)
- 同個函數名稱卻有不同的定義及功能，這就稱作**多載(overloading)**。

練習

- 想要計算一個三角形的面積有很多不同的算法，例如：

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

double area(int, int, int);



double area(int, int);

$$\text{Area} = \frac{bc}{2} \sin A$$

double area(int, int, double);



double area(int);

多載函數還能做到的事

- 若你定義了一個類別用來儲存複數，那要怎麼做複數四則運算呢？
 - 可自己定義屬於自己的四則運算(+, -, *, /)。

```
class Test{
public:
    Test(){ a = b = 0; }
    Test(int n, int m){ a = n; b = m; }
    Test operator+(Test);
    void display(){ cout << a << " " << b << endl; }
private:
    int a, b;
};

Test Test::operator+(Test obj){
    Test tmp;
    tmp.a = a + obj.a;
    tmp.b = b + obj.b;
    return tmp;
}
```

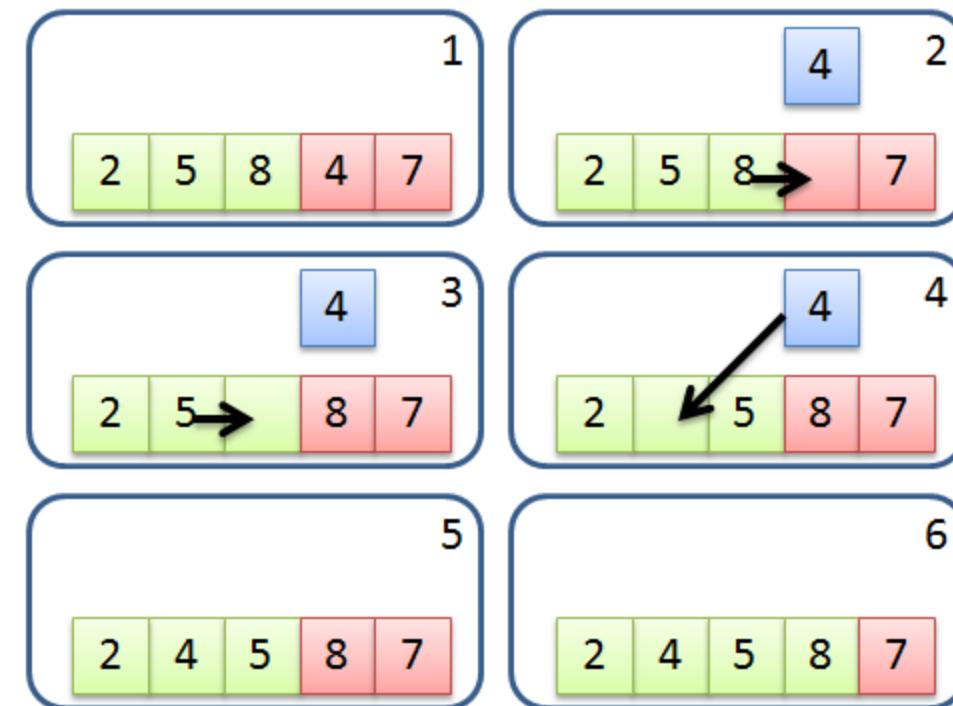
進階演算法

插入排序法(Insertion sort)

- 插入排序法(**Insertion Sort**)是排序演算法的一種，也是一種簡單容易理解的演算法。
- 對於未排序資料，在已排序序列中從後向前掃描，找到相應位置並插入。
- 在從後向前掃描過程中，需要反覆把已排序元素逐步向後挪位，為最新元素提供插入空間。

插入排序法(Insertion sort) – 實作

```
void insertion_sort(int arr[], int len){  
    for(int i=1; i<len; i++){  
        int key=arr[i];  
        int j=i-1;  
        while((j>=0) && (key<arr[j])){  
            arr[j+1]=arr[j];  
            j--;  
        }  
        arr[j+1]=key;  
    }  
}
```



插入排序法(Insertion sort) – 複雜度分析

- 最佳時間複雜度： $O(n)$
- 平均時間複雜度： $O(n^2)$
- 最差時間複雜度： $O(n^2)$
- 空間複雜度： $O(1)$

快速排序法(Quick sort)

- 快速排序法(**Quick sort**)是排序演算法的一種，使用*Divide and Conquer*的演算法來實作。
- 其概念是從數列中挑選一個基準點，大於基準的放一邊，小於的放一邊，如此循環最後可完成排序。
- 依照以下步驟進行(遞增為例)：
 1. 數列中選擇一元素作為基準點(pivot)。
 2. 小於此元素的移至基準的左邊，大於此元素的移至右邊，相等的任意放。
 3. 基準點左邊和右邊視為兩個數列，並重複做以上動作直到數列剩下一個或零個元素。

快速排序法(Quick sort) – 實作

```
function sort(list)
    if list.length < 2
        return list
    end if
    pivot = 從list取出一基準點
    var less, greater, result
    for i = 0;i < list.length;++i
        if list[i] > pivot
            greater.add(list[i])
        else
            less.add(list[i])
        end if
    end for
    result.add(sort(less))
    result.add(pivot)
    result.add(sort(greater))
    return result;
end function
```



快速排序法(Quick sort) – 複雜度分析

- 最佳時間複雜度： $O(n \log n)$
- 平均時間複雜度： $O(n \log n)$
- 最差時間複雜度： $O(n^2)$
- 空間複雜度： $O(n)$

其他程式語言簡介

常用程式語言

語言名稱	用途
C	數據處理、研究用
python	機器學習、爬蟲、數據處理
C#	遊戲開發（Unity主要使用語言）
swift	iOS開發
java	Android開發
R	統計專用
php	網頁後端
javascript	網頁前端

python (ML)

Show data

In [4]:

```
input_shape = (256,256,3)

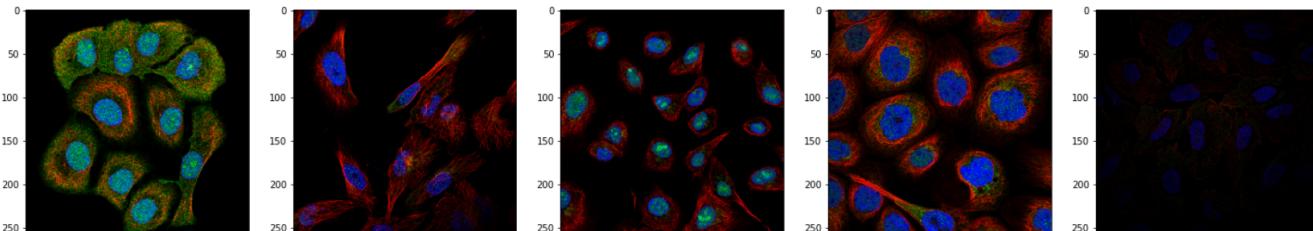
# create train datagen
train_datagen = data_generator.create_train(
    train_dataset_info, 5, input_shape, augment=True)
```

In [5]:

```
images, labels = next(train_datagen)

fig, ax = plt.subplots(1,5,figsize=(25,5))
for i in range(5):
    ax[i].imshow(images[i])
print('min: {}, max: {}'.format(images.min(), images.max()))
```

min: 0.0, max: 1.0



Featured Prediction Competition

Human Protein Atlas Image Classification

Classify subcellular protein patterns in human cells

Human Protein Atlas · 1,440 teams · a month to go (a month to go until merger deadline)

Overview Data Kernels Discussion Leaderboard Rules Team My Submissions Submit Predictions

Overview

Description

In this competition, Kagglers will develop models capable of classifying mixed patterns of proteins in microscope images. [The Human Protein Atlas](#) will use these models to build a tool integrated with their smart-microscopy system to identify a protein's location(s) from a high-throughput image.

Evaluation

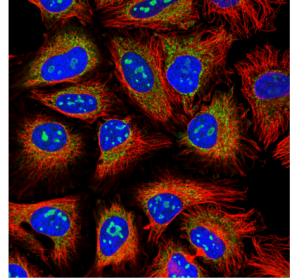
Prizes

Timeline

Special Prize Instructions

Proteins are “the doers” in the human cell, executing many functions that together enable life. Historically, classification of proteins has been limited to single patterns in one or a few cell types, but in order to fully understand the complexity of the human cell, models must classify mixed patterns across a range of different human cells.

Images visualizing proteins in cells are commonly used for biomedical research, and these cells could hold the key for the next breakthrough in medicine. However, thanks to advances in high-throughput



C# (Unity)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class GameController : MonoBehaviour {

    public static GameController _GameController;

    public GameObject loadingScreen;
    public float loadingTimeMin = 1.0f;

    public string chapterName;

    void Awake(){
        _GameController = this;
        DontDestroyOnLoad(this.gameObject);
        DontDestroyOnLoad(loadingScreen);
    }

    // Use this for initialization
    void Start () {
        Screen.SetResolution(1920, 1080, FullScreenMode.Windowed);
    }

    // Update is called once per frame
    void Update () {

    }
}
```



關於學習程式語言

- Google 上幾乎可以找到任何你想學的語言的教學。
- 我們使用過的一些網站也可以幫助你學習
 - Zerojudge
 - HackerRank
 - LeetCode
- 學程式語言不外乎就是：**多看，多想，多練習**。
- 實戰也可以大幅提升你對特定語言的熟悉度。

課程回顧

學習目標

- 能夠獨自撰寫簡易的C++程式
- 了解基礎的物件導向程式設計方法
- 閱讀程式碼的能力

時間表

- 7/9] • 資料型態、變數、基本輸入輸出
- 7/12] • 流程控制（條件控制、迴圈控制）
- 7/16] • 函式與陣列
- 7/16] • 字元與字串、結構
- 7/19]
- 7/23] • 指標
- 7/26] • 資料結構
- 7/30]
- 8/2]
- 8/6] • 基礎演算法
- 8/6] • 物件導向程式設計基礎
- 8/16]

作業

- Reading: 課本 Ch11.5.3, Ch12
- 若遇到程式相關問題，歡迎隨時寄信至：ck1001099@gmail.com

課程結束！