C++ 程式設計

丁 培 毅

106/02 – 106/06

http://squall.cs.ntou.edu.tw/cpp/

http://sirius.cs.ntou.edu.tw/cppBB/

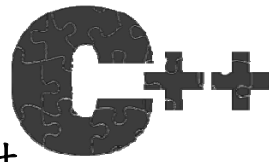
Contents

C++ 這個課到底要學些什麼?

- 好多朋友在其他學校都從 C++ 開始學, 為什麼我們到大二下才有這個課? 為什麼不在一年級就上 C++ 呢?
- 以程式設計線上平台的題目來說, 像是UVa, Codeforce, POJ, e-tutor, 看到很多人用 C++, Java, Python 解題, 好像都有很多工具, 答案可以比較簡短
- 對好多人來說, C++ 就是 cin, cout, vector, list, deque map, reference, overloading, 外加 C 本來就有的陣列/迴圈/函式與指標的東西
- 如果設定成這樣, 那麼我們可以在兩個星期裡把語法介紹完,就結束這學期的課程了, 大家爽爽過一個學期
- 不... 這個 ++ 不是只把輸入輸出換掉, 多了 STL 函式庫
- 真的有關鍵影響力的是提供其它的設計典範 – **OOP**, FP

程式設計思維的改變

程序式程式設計
Procedural Programming



物件導向程式設計
Object-Oriented Programming

函數式程式設計
Functional Programming

設計思維典範的改變 Paradigm Shift

好的**工具**, **使用方法**錯了,
是看不到效果的!

物件導向應用軟體開發過程

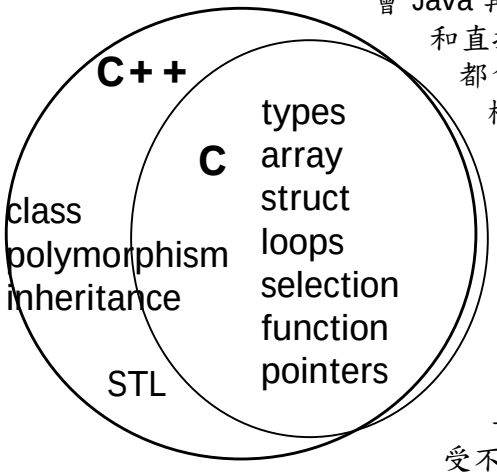
物件導向分析
Object-Oriented Analysis, OOA

物件導向設計
Object-Oriented Design, OOD

面向对象程序设计

物件導向程式設計
Object-Oriented Programming, OOP

學習 OOP 的媒介



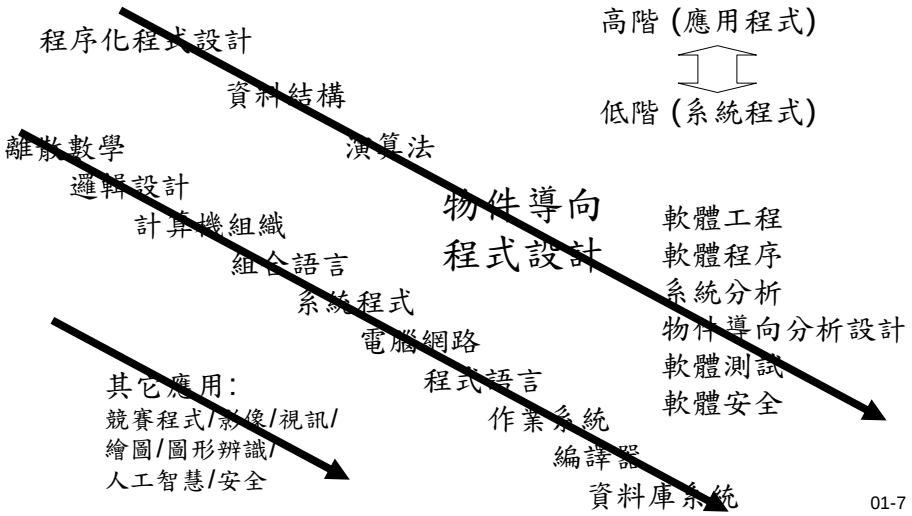
會 Java 再學 C++, 需要的時間大概和直接學 C++ 差不多, 自學大概都會放棄, Java 簡化到很多物件概念你都看不到

會 C++ 再學 Java 時咦, 語法看一下就夠了, 工具類別和圖形界面類別的設計都是物件和設計模式的應用

為什麼不用 Java???

大家都說 Java 學習曲線平緩而且可以製作圖形化界面受不了 C/C++ 單調的小黑框啦!

學習軟體設計的過程



花了一年半的時間學寫程式

- 學了程式設計, 也學了資料結構、演算法、組合語言, 程式好像寫得稍微大了一些, 可是說真的離你現在天天使用的軟體的規模和功能好像還有點.....距離，更糟糕的好像看到成長的曲線放緩了, 看不太到該怎麼進步下去!!
- 別說再繼續擴大程式的規模了, 就算是現在的 300^+ 行程式, 你自己或是同學一定常常會被裡面的 bugs 弄得想要「打掉重練」, 好像很容易有拿不太掉的錯誤, 怎麼繼續增加功能呢? 寫軟體真的一定會要常常打掉重練嗎? 如果蓋房子常常要蓋蓋拆拆, 應該不會有建商存活得下去吧! 那麼寫軟體呢? 不會要寫寫刪刪吧? 這樣的工作不是人做的吧!!
- 有看過因為大門不美要拆掉整棟樓房的嗎? 有看過因為輪圈壞了要換掉車子的嗎? 有因為硬碟不夠大、滑鼠不靈敏要換掉機器的嗎? 適當的設計下什麼東西都可以修理, 軟體呢?! 如果讓你覺得不太能修理是不是架構設計有問題?

物件和物件導向

- 這個課程的目的就是要把物件化、物件導向的軟體方法介紹給你，提昇你組織架構軟體的能力，讓你能夠設計更大規模的程式，讓你能夠直接配合很多**軟體框架** (framework) 一起設計各種功能 –例如 Graphic User Interface (MFC, Qt, Swift, ...), Web application frameworks (Ajax, .net, Wt, ...)
- 不是人家寫好下層的工具給你用喔!! 是你要寫適合的介面來接上現成的**框架**，你需要和設計好的框架密切合作!! 要能夠了解整個系統的設計概念你才能夠做出正確的設計，要能夠活用物件導向的概念，你才能夠在不修改原本框架的情況下快速擴充它的功能，突破這個瓶頸之後，要寫出 3000^+, 5000^+, 10000^+ 行正確運作的程式就指日可待了

物件導向程式設計 - C++

- 在前三個學期裡大家會掌握 ***程序式 C 程式設計***、***資料結構***、***演算法*** 這三個科目
- 已經滿足許多 **科學與工程應用** 的要求，輔以適當的 **領域知識**，電腦的速度與記憶容量就可以協助自動化以及設計系統參數

可是也還得很多專業課程你要到各個不同系所去學，你才會有足夠的知識去解決… 數學系、物理系、電機系、機械系、土木系、化工系、財金系、生科系… 不要忘記… 的內容其實門檻不高，基本上自學也可以…

– 多媒體展現、網路設計、UI/UX、聲音、影像、視訊、通訊、網路、控制、訊號處理、圖形識別、語言處理、資訊處理、統計、電腦圖學、深度學習、大數據
– 生物資訊、財務金融運算、雲端與網路安全、防毒防駭
– 介面裝置驅動與微處理機控制、作業系統、網路系統
– ASIC, FPGA, VLSI 設計

還有很多應用領域… **不能畢業** (還好，這些大家通常不會很想畢業)

那麼你讀資訊系的優勢在哪裡?

I. hand holding honeymoon　學　Job Ready

很快的你發現只會 ***程序式*** 的程式，和很多商務/業界的

https://www.ptt.cc/bbs/Soft_Job/M.1433309760.A.9D5.html
時間**Wed Jun 3 13:35:58 2015**
職務名稱: **C++ 程式** 演算法 工程師Software Engineers & Quantitative Programmers
職務類別: 資訊助理人員、**軟體設計工程師**、統計學研究員
Software Engineer/Developer & IT Support (Full-time)
公司名稱: 俊一研發有限公司 Alluvium Global Research http://www.alluviumgr.com
公司位置: 靠近101
薪資範圍: NTD $80,000-$380,000(月) 視能力而定(面議)
職務內容:
1) 程式工程師
Requirements: **C++**, C, linux
Degree in any of the following: computer science, engineering, math/statistics, operations research, c
Preferred: **Knowledge of design patterns,** systems programming, kernel, networking TCP/UDP
Duties: Develop/debug software and support servers/networks, Implement algorithms for data ma
Maintain documentation of procedures, models, and programming work
2) 演算法工程師
Requirements: Familiar with R or Matlab (or NumPy/SciPy)
Degree in any of the following: engineering, math/statistics, econometrics, operations research, or ph
Preferred: Knowledge of **probability theory**, **stochastic processes**, or t**ime series analysis**
Duties: Perform statistical analysis/modeling on large data sets; prepare research reports
連絡方式: 應徵請寄一頁履歷表至 human.resources@alluviumgr.com, 請在標題註明，從PTT來

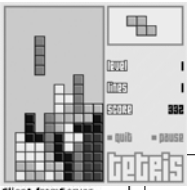
希望你走出去

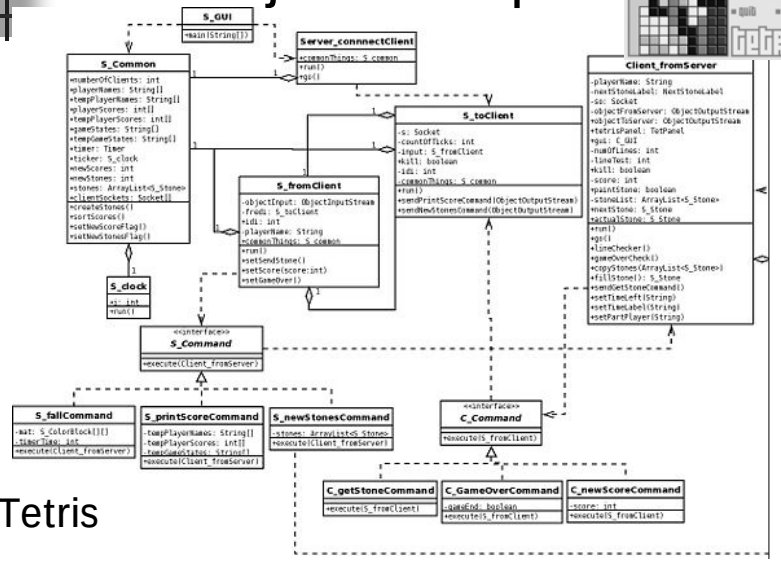
在各種環境壓力下都可以愉快地生存

- 有沒有同學希望留在這兒慢活的??
- 可以舉手... 雖然是必修課
  可是還是要給你一點選擇的權利
  我可以考慮...

- 有困難度喔!
- 要投入時間喔!
- 會不會跳太快了啊?
  - 老師你以前學程式設計, 資料結構, 演算法之後, 就直接把目標朝向 10000^{+} 了嗎?
- 不用額外繳$$$的喔
  - 把同仁送到資策會花 20000/五天課程
  - 在國外上課時, 很多其他領域轉來的學十個星期就完全會了

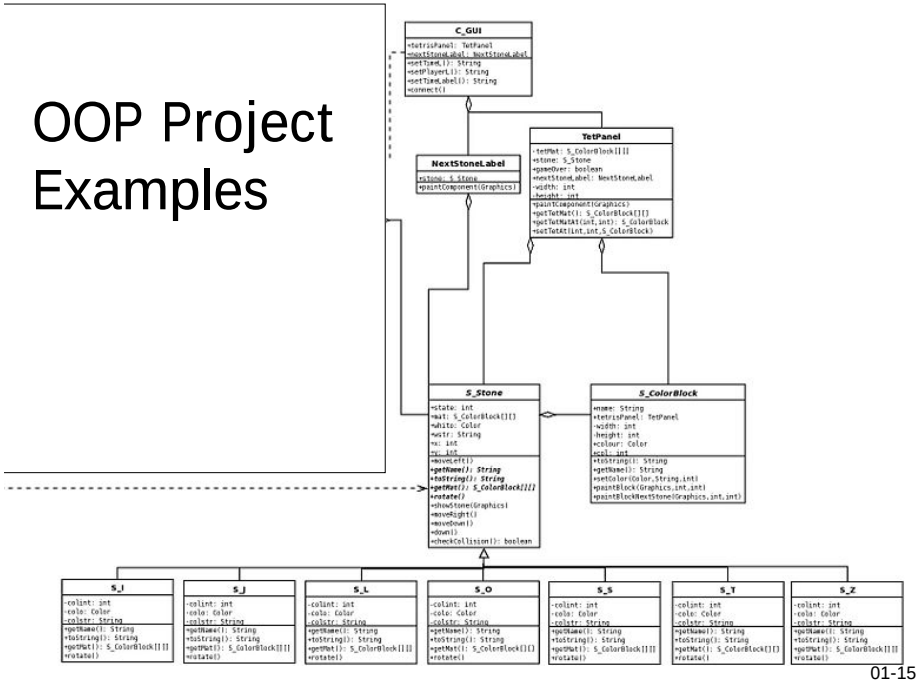
OOP Project Examples



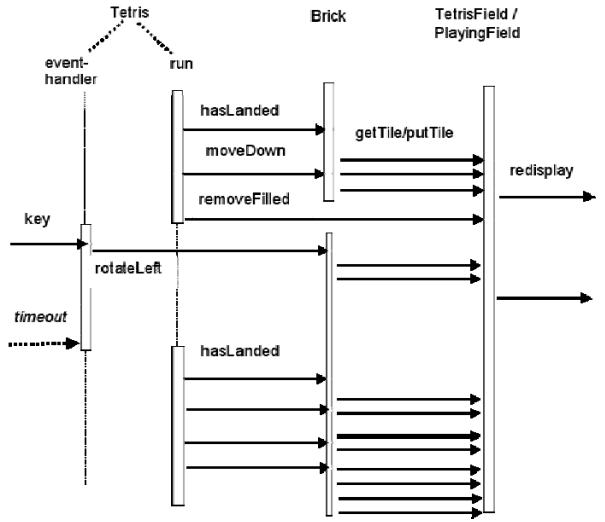


- Tetris

OOP Project Examples

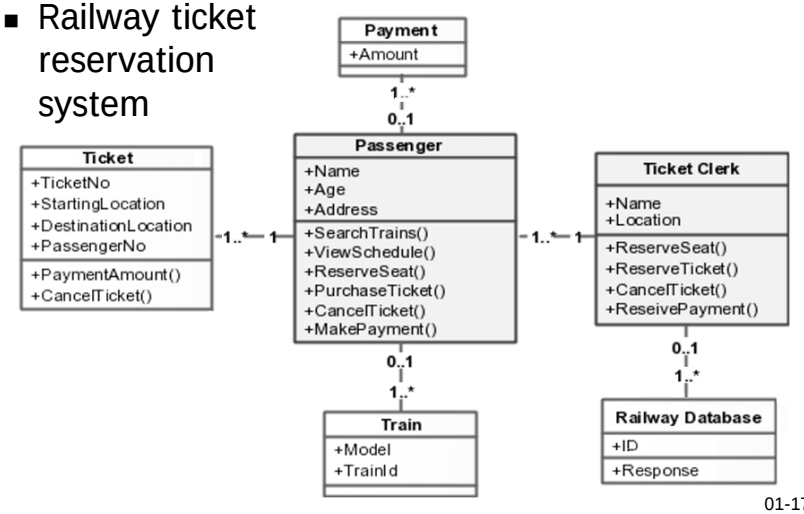


OOP Project Examples



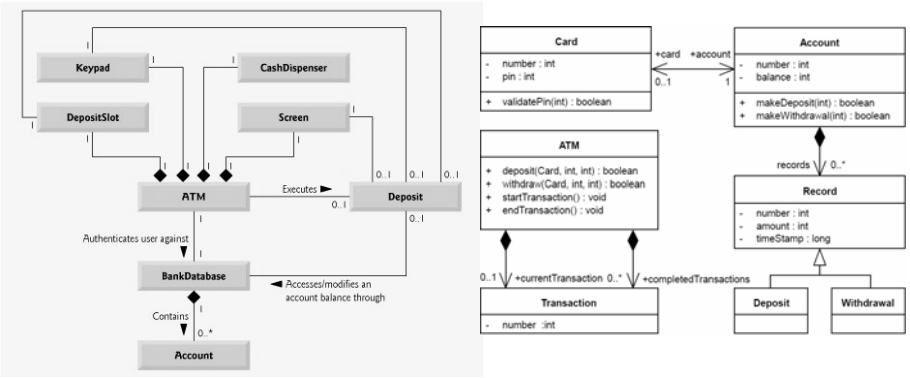
OOP Project Examples

- Railway ticket reservation system



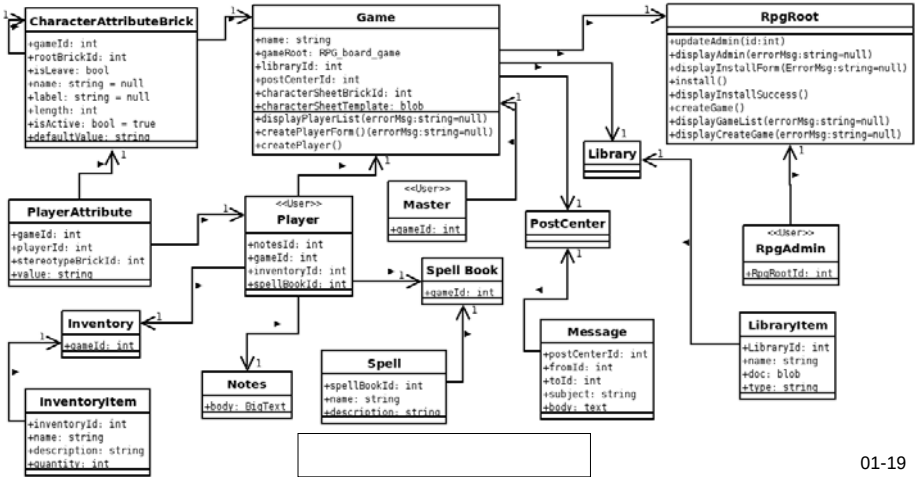
OOP Project Examples

- ATM



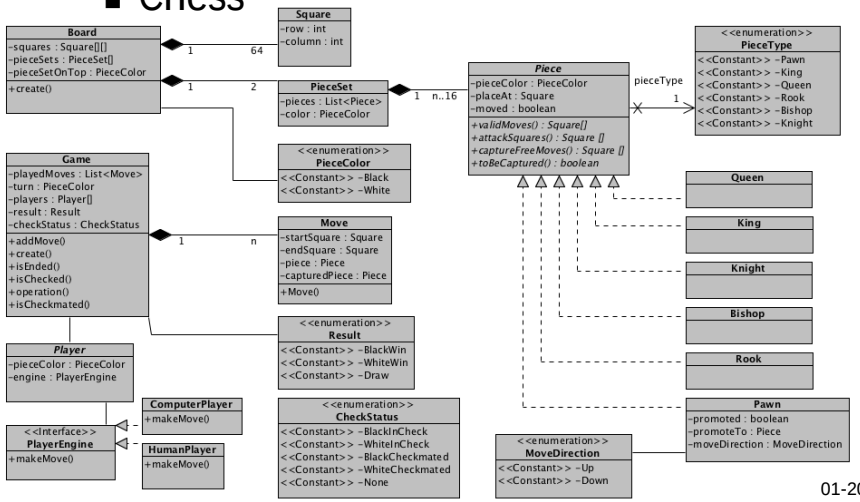
OOP Project Examples

- an RPG game

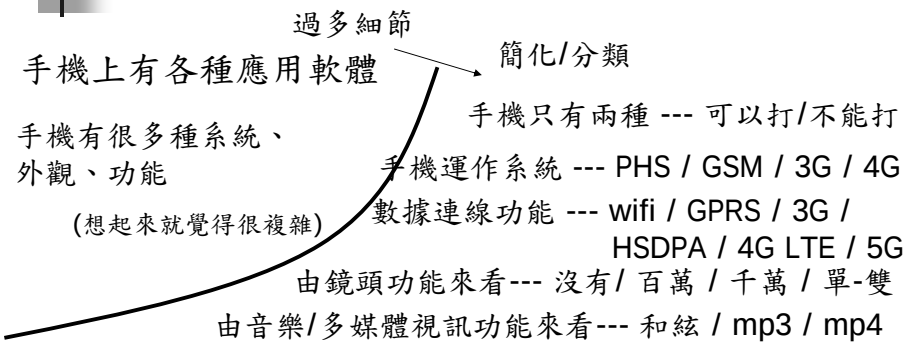


OOP Project Examples

- Chess



抽象化 (簡化) 過程



昆蟲有很多種 --> ...　　會計報表 --> 簡報圖表

抽象化 (簡化) 過程 (cont'd)

- 抽象化 (Abstraction):
  去掉不在意的細節, 專注於所在意的性質
- 在 OOP 中以 介面 (interface)
  來規範抽象化以後軟體物件的功能
  介面可以看成是一組操作物件的方法
- 抽象化是實現軟體開發時 "Open-Closed Principle" (Bertrand Meyer, 1985) 的關鍵

OCP { Software entities (classes, modules, functions, etc.) should be *open for extension* but *closed for modification*

OOP 重用"軟體設計"

軟體物件的重複使用 (Reuse)
- 縮短應用軟體的生產過程
- 降低軟體更新功能時的阻力、縮短修改的流程
- 減少不必要的錯誤

- 封裝 (Encapsulation)
  確保實作介面功能的正確性, 降低軟體的成本
- 繼承 (Inheritance)
  重用軟體的架構 (程式碼與介面)
- 多型 (Polymorphism)
  基於抽象介面撰寫程式, 使用不同實體物件的機制

教科書

Object Oriented Programming in C++, R. Johnsonbaugh & M. Kalin, 2nd Ed.

- Thinking in C++, Bruce Eckel, 2nd Ed., http://www.bruceeckel.com/
- 課程網頁
  http://squall.cs.ntou.edu.tw/cpp/
  - 電子書, 參考書, 參考資料
  - 課程內容摘要, 投影片
  - 作業
  - 實習
  - 討論群組
  - 過去課程網頁

C++ 語言演進與標準

starting from **C**, **Simula**, **ALGOL 68**, **Ada**, **CLU**, **ML**

- 1979: **C with Classes** by Bjarne Stroustrup
- 1983: **C++** **這學期的課程**
- 1998: **C++98** – ISO/IEC 14882:1998, 1st standardization
- 2003: **C++03** – ISO/IEC 14882:2003, few corrections
- 2007: **C++TR1** – ISO/IEC TR 19768:2007
- 2011: **C++11** – ISO/IEC 14882:2011 (formerly C++0x)
- 2014: **C++14**, minor revision
- Scheduled: **C++17**, major revision

課程使用軟體環境

- Compiler: VC 2010
- 開發環境:
  - IDE: Visual Studio 2010
  - Command Line: VC 2010

如何上課

- 請預習
  - 課本
  - 中文參考書
  - 投影片, 講義
- 上課
  - 請隨時準備問題, 這和你未來是否會留在資訊業界有密切關係
- 請複習
  - 不定期測試
  - 期中考/期末考

我們生活在浩瀚的知識海洋中

在這個瞬息萬變的環境中，每個人都接觸到很多狀況，但是學習到的知識因你的**態度**而異

學校與課堂中希望能夠幫助你尋找一條你自己的路

你，同學，助教，和老師要充分合作才能幫助你迅速獲得你想要的

『學問』...要學也要問

- 問問題是有方法的
  - 會問問題的人是能夠掌握自己目標的人
  - 是隨時動腦的人
  - 是能夠 (願意) 和同僚互動的人 (心態常比能力還重要)
- 問得好，你很容易得到你想要的，解決你的困惑，沈澱自己的想法，被問的人甚至由你的問題中獲益
  - 大部分主管/老闆其實都不知道底下的員工整天在做什麼，可是常常幾個問題就可以掌握 80%
- 問得不好，隨便問，太籠統，吹毛求疵，被問的人不知從何答起，他的回答也幾乎對你沒有作用

發問

- 問問題需要練習嗎?
  - 當然需要, 很多同學遇到需要問問題時腦子裡一片空白
  - 在學校裡練習問問題你沒有損失, 只有獲得, 老師也沒有損失, 同學也會有其它獲得
  - 有些同學會用影響課程進度作為不問問題的藉口, 那要看你問什麼問題, 不問看看你怎麼知道? 沒有常常看到同學在問問題, 你怎麼能學到怎樣問問題才是有用的?
  - 怕老師答不出來 (真好心, 不過老師也有很多招數的啊)
  - Trick: 有的時候問問題可以取代答案...
- 要問得好, 需要了解問題的答案嗎?
  - 雞生蛋蛋生雞
  - 不需要, 也不可能... 不過有適當的預期是有幫助的

發問

- 怎麼會透過問問題來學習呢?
  - 要別人直接幫你思考, 這樣會不會太偷懶了啊?
  - 很多時候你在組織問題的時候就是強迫你自己在壓力下思考, 分析所有手邊的資料, 讓你的腦子裡一些不常用的部份一起發揮功能, 常常會有意想不到的結果
  - 這是增加探討問題深度的主要方法
- 方法
  - 需要從不同的面相去觀察一個主題
  - 可以由如何應用, 用途, 功能性來探討; 可以由動機來探討; 可以由本身的推理邏輯來探討; 可以由其他配合的環境來探討; 可以探討效率的問題; 可以由其他相類似的問題的解決方法來橫向的探討...

發問

- 期勉

  當我們能夠不斷地討論問題的時候

  - 大家可以觀察到老師呈現知識的步驟, 可以觀察到同學在這個環境中如何思考, 解決問題的步驟
    -- 否則用一片教學 VCD 就可以達到更大的效果
  - 大家才能夠真正運用到上課的時間**深入**學習, 所獲得 ***深度的知識*** 與 ***探索知識的方法*** 常常可以幫助你探討其他沒有在課堂裡談到的知識 (不要嫌上課得到的東西太少, 老師教得太少...幫你釣再多的魚你沒辦法自己如法炮製終究是一種浪費... 見山不是山見水不是水也不過是好高騖遠, 終究無法體會)

程式學不會

沒興趣?!　　看你從什麼角度看

熱門音樂都是一樣的 --- 都很吵

古典音樂都是一樣的 --- 都很沈悶

籃球沒什麼好看的 --- 一個球十個人搶來搶去

物理、數學很無聊 --- 式子一堆太繁瑣, 我該不會用到吧

那麼程式設計呢??　　怎麼寫都有錯誤...

資訊系畢業不見得一定要寫程式吧?!

學不會?

- 嚴格來說, 學不會是一種藉口
  - 沒有付出足夠多的代價, 去把東西弄清楚
  - 你花的時間有比玩 Game 的時間多嗎? 比上網哈拉的時間多嗎? (和你有興趣的東西比較看看吧)
- 在大學裡這麼多科目的訓練, 希望你收穫到的是 "怎麼學會一種知識的方法", "建立怎樣學會一門知識的信心", 如果你真的對某樣東西沒有興趣時再換就是了, 如果從來沒有付出代價學會過什麼東西, 一直在尋找一個你可以不用付出太多就可以獲得很多的行業...

資訊業的特性

- 當然有這樣的行業, 不過通常也適合大部分的人, 也就是說競爭會很激烈, 手段會很殘酷, 鬥爭會很血腥
- 表面上看起來資訊業就是這樣的一個行業, 不管你是什麼系畢業的, 經過三個月、六個月的訓練你就是專家了...(不相信你去外面軟體公司或是公司的軟體系統部門看看)
- 看起來門檻很低,人員的汰換律和取代性很高...
- 既然如此,各位何必要那麼辛苦地學習呢?

資訊業的特性 (cont'd)

- 你被人家取代的機會高嗎? (22K vs 40^{+}K?)
  - 你對系統的瞭解深入嗎? (你能夠修改作業系統核心嗎? 你能夠設計驅動程式嗎? 有辦法設計嵌入式系統?)
  - 你能夠撰寫網路通訊應用程式嗎? 圖形介面?
  - 你能夠設計大型的物件化應用程式嗎?
  - 你能夠設計複雜的知識存取、檢索核心嗎?
  - 你能夠應用大數據?
  - 你能夠設計高階的 3D 圖形介面嗎? 3D 列印模擬?
  - 你能夠設計智慧型的應用嗎 (語言轉譯、人工智慧、深度學習、機器學習、影像、視訊、語音輸出入...)?
  - 你能夠設計安全的軟體和通訊系統嗎?

資訊業的特性 (cont'd)

- 簡單地說...你希望建立一道學習方法、知識與技術的防火牆...你希望透過幾年的苦讀突破瓶頸, 建立別人無法在短時間裡追上的屏障 ...
- 沒有一種知識有絕對困難的門檻...就算有,也不見得每一個人都會遇見相同的瓶頸...
- 回應先前的說法, 這個門檻其實是你自己學習與應用的瓶頸...是那個一遇到困難立刻退縮的本能反應
- 很多同學到了三、四年級看到系上選修課程, 只想要找簡單的, 營養的, 不用花太多時間的課程選修
- 於是就加入了容易被取代的一群, 甚至還沒有畢業就已經開始尋求第二專長了, 資訊業不是很缺人嗎?

條條大陸通羅馬

沒有一條路、一個固定的模式適合所有的人

你自己找到的方法 才是對你最有用的,

雖然該滿足的**標準**可能由不得你

不過切記 有準備的人才能掌握機會

準備什麼?

- 準備一種接受挑戰的心態
- 準備一種系統地分析、解決問題的方法
- 準備一種尋找知識、接收知識、轉化知識為工作能量的成功模式
- 準備一種隨時面對環境變化的信心
- 準備一種逐步驗收成果的工程信念

抽象嗎?

就在當下

- 有的人永遠看前面, 看哪一條路所需要付出的和所能夠獲得的比例最高 (如果是買東西的話就是 cp 值囉)
- 在萬象的世界裡, "尋找" 所需要花費的時間是很冗長的, 可以保證你有很大的機會會錯過 "最好的" 一條路徑, 沒有辦法做出"最好的" 選擇
- 於是你必須要學會下賭注, 做出選擇, 一旦下了決定就必須全力實現, 在有限的時間裡掌握所接觸的一切, 就是對你的未來做好最萬全的準備

怎麼調整你自己的學習方法?

- 很多接觸軟體開發設計的人都曾經萌生退意...(你呢?)
  - 開發工具變化好快
  - 系統好複雜, 好大, 好抽象
  - 設計的選擇性好多
  - 永遠有改不完的程式錯誤
  - 使用者的無理要求
  - 掌握不到軟體設計的精髓, 覺得好像不知道該要求些什麼, 有點不得其門而入

怎麼調整你自己的學習方法?

- 老師、同學、甚至網友都是你的鏡子
- 覺得沒有效率或是沒有成效的話, 自己思考一下, 整理一下, 和別人討論自己的心得
- 不要期待別人給你什麼實質的回饋, 自己整理問題和心得時一定會有新的體會
- 凡事總要自己提出自己的看法
- 再由同學、課本、課堂、實習中驗證

其它工程系所 vs. 資訊工程

- 機械、造船、土木
  - 將來做出來的東西很具體
  - 基礎學習過程比較辛苦, 不知道什麼時候可以真的踏進生產線
  - 模型
- 資工
  - 成品有點抽象
  - 學習的過程中不斷地可以作出成果出來
  - 模型?

軟體的特性與要求

- 軟體之所謂軟...因為沒有“硬性”不可變、不可挑戰的規則
  - 好處: 彈性很大, 山不轉路轉, 沒有標準答案, 正常運作就好...
  - 壞處: 很多小問題合在一起不斷放大, 到處藏污納垢, 沒有標準答案, 不知道到底對了沒有
- 解決方法

  - Coding styles, test-driven
  - 元件化
  - 模型化 (資料結構, 演算法, 物件化, 軟體模式)

OCP

對於大家的期許

- 過去二十年在海大資訊系, 看到很多同學從升上大二開始擔心自己的未來, 開始尋找必勝的捷徑, 於是看到有升研究所的補習班, 什麼都可以補, 既然看不到很明確的未來, 為何不去補習, 吸收一些考試的技巧, 也許將來藉由台大、清大、交大研究所的光環, 能夠少奮鬥幾年??
- 考試是“結果論”的, 考試成績不好似乎就代表一定的失敗, 可是會考試一定不代表能夠解決實際的問題, 反倒是把時間全放在準備考試上, 你一定無法兼顧正確的學習, 無法真正提昇自己的學習能力與思考能力, 而有實質的損失
- 你經歷了十年考試制度的摧殘, 你慢慢能夠抵抗這個怪獸了, 還打算再次投入這個錯誤制度的循環?!

對於大家的期許 (cont'd)

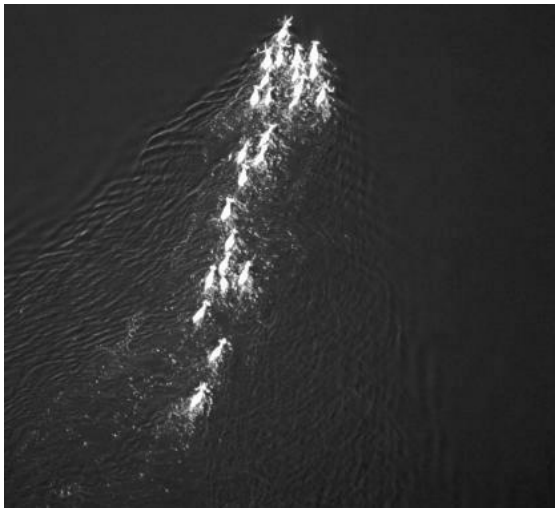
- 提醒你注意一件事，為什麼台清交成的大學畢業生沒有把他們的研究所填滿? 為什麼台清交成那麼多的畢業生，台灣的科技業還找不到人材? 為什麼這麼多優秀的同學都轉業了，為什麼在職場上不會很自豪?
- 會不會是這些「好學校」根本沒有辦法協助大部分同學突破學習瓶頸，變成對社會有貢獻的優秀工程師?
- 為了在考試制度裡維持突出，只能有競爭性的藏私而不能有合作式的心得分享，人類科技還能被這樣摧殘多久? 台灣有限的人力資源還能夠這樣浪費多久? 或許鬼島不是別人一手造成的，我們大家在這個循環裡都有相當程度的貢獻!!
- 把精力集中在專業科目上，集中力量跨過阻礙自己學習的門檻，建立自己學習的信心，機會來了你就一定能夠掌握

對於大家的期許 (cont'd)

- 常常聽到、看到一些實例，一些業界的老闆可以很快地就淘汰台清交的畢業生，現在國際性的競爭非常劇烈，業界需要的是實力，沒有實力的話，學歷光環是撐不了多久的
- 我也聽到一些我們系上進了台清交成的學長的抱怨: 學習資源獲得的困難，藏私性競爭的慘烈，所需要付出的適應，和爭取不到的合理重視
- 這兩年我也聽到這些錄取我們系畢業成績前茅同學的系所老師的抱怨，感嘆我們的畢業生程度越來越差了?! (話說這難道看不出來是因為你們用考試來錄取學生，考試又跳不出考古題範圍，又有補習班推波助瀾的結果嗎? 讓我們在這裡想要按步就班教學，一點一滴培養同學解決問題的能力，逐步建立專業能力都很困難! 這樣子教會考不上好學校喔!?)

對於大家的期許 (cont'd)

- 從小到大，各種考試只要會 60%，你就可以順利過關，記多一點，看多一點，練習多一點就會有 80 甚至是 100 分
- 很多時候 100 分又好像是個丟臉的目標，放棄很多有趣但是不會考的東西，仔細想想還真的不該是個目標...
- 不過，學習寫程式如果以會 60% 為目標，雖然你還是可以順利過關，但是你自己知道程式很多地方都不會順利運作，和其它程式合在一起時錯誤一定會出現 (墨菲定律)，你以後會不敢與軟體為伍...(雖然說其實軟體公司才真的不敢與你為伍...)
- 不要只因為考試而學習，更不要只把60分當成是目標，看到考試有了60分就放心了，100% 的了解課程內容是比較好的目標，課程之外還有很多很多東西呢? 不然為什麼要叫做 University 呢?



在叢林法則中，沒有到達
彼岸唯一的下場就是被淘汰

學寫程式的心態

- 學寫程式是學習怎樣由程式的功能要求轉換為可以正確運作的程式
- 找到轉換的基本規則以後就會覺得自己會寫程式
- 很多同學在前幾個學期的學習過程裡, 不小心跳過了那個轉換的過程, 也沒有整理出心得來
  - 比方說如果你的作業是由課本的範例為起點
  - 或是由google,學長/同學流傳的範例開始
  - 或是在還沒有想法的時候直接問人

  等你過了這段學習的過程以後, 運用找得到的資源來加速工作的進行是理所當然的事, 可是不是現在...
- 雖然參考別人的程式可以節省很多的時間, 可是你自己放棄練習那個轉換過程, 下次拿到稍微改變的要求, 還是不知道如何下手

學寫程式的心態 (cont'd)

- 不同的程式開發模型, 撰寫程式的方法就不一樣, 用C寫程序化的程式和用C++寫物件化的程式就相當不一樣, 我們以後會仔細分析這兩種方法的差異
- 在設計程序化的C程式(資料處理的模型)時你可以
  - 根據程式的要求, 用紙筆自己模擬嘗試完成程式要求的計算 (也許需要設計適當的演算法)
  - 設計一些標準的輸入資料, 設計其他可能的輸入資料
  - 把輸入的資料和輸出的資料用符合電腦模型的方式表示出來 (也許需要適當的資料結構)
  - 把每一個手動步驟換成符合語法、符合電腦操作模型的處理程序
  - 運用前面步驟中設計的資料驗證每一部份程式是否正確運作
  - 測試與除錯

學寫程式的心態 (cont'd)

- 前一頁把手動步驟轉換為程式的處理程序寫得好像很簡單...(實際上好像是常常構成阻礙的困難點!?)
- 有經驗的人都是這樣讓程式直接蹦出來的嗎? (如果是這樣的話那真的要一點天份了, 真的是 art of programming 了, 我們應該改名成資訊藝術...不要再叫做資訊科學/工程了)
- 解決問題時如果發現沒有辦法一下子完成, 應該有幾種基本的作法:

  ❶簡化問題: 要你處理1000筆資料, 可以先想5筆資料該怎麼做; 由網路讀取資料你可以先改成由鍵盤讀取; 有十道步驟可以先完成最重要的兩道; 適當地假設: 把前五個步驟的結果寫在程式裡, 就可以直接思考第六個步驟的做法

  ❷尋找類似問題的解決方法

學寫程式的心態 (cont'd)

- 當看到程式竟然對了, 別太興奮, 只要不在預期之內就要仔細分析, 程式功能不是亂試出來的, 需要是設計出來的
- 當你發現程式執行結果和你預期不一致的時候, 千萬不要假裝沒有看到, 想說下次不會再遇到它了吧! (程式語言的模型很制式化, 你一旦遇見一種錯誤沒有更正, 就會不斷地犯同樣的錯誤)
- 更不要想辦法隱藏, 隱藏程式裡的 bug 就像說謊一樣, 要用一連串謊言來遮掩, 最後還是會暴露出來的
- 每一個 "錯誤" 或是不如預期的 "程式表現" 都是一個矯正你所認知的 "程式運作模型" 的機會
- 當你能夠自由地運用程式的語法製作如同你所預期表現的程式時, 你就算跨過程式設計的學習門檻了, 你就再也不會耽心一直出現很多不同的語言了

這個學期裡

- 希望不要是

- 或是

- 希望課程實習與作業的安排比較像

- 後面很多單元裡你會看到有很多的東西需要學習，需要練習...
- 我當然知道你的時間有限，但是學多少是多少，一步一步往前走，所練習的學習方法是最重要的，不要因為吃不到那個蘿蔔而氣餒，真正的牛肉還不在這個課程裡...
- 在你以後資訊領域的生涯裡自然會發現你所建立的學習方法以及了解這些工具以後的好處