1131-1 NTOU CSE 程式設計

113 學年第 1 學期

國立台灣海洋大學 資訊工程學系一年級C班

課號: B5701M33

上課教室與時間: INS 101 (星期二 13:10-16:00)

上機授課與實習：INS 203/201 (星期三 13:10-16:00)

補強時間: INS 203/201/205/301 (由助教另外安排)

授課教師: 丁培毅

- 丁培毅 (Pei-yih Ting)
  - 專長: 密碼學與資訊安全、深度學習、訊號處理
  - email: pyting@mail.ntou.edu.tw
  - facebook/messenger/line

- 非同步教材/實習網頁
  http://squall.cse.ntou.edu.tw/cprog
- https://tronclass.ntou.edu.tw/course/149984/conter
- 實習程式線上繳交與評測
  http://140.121.198.92:9487/
- fb群組:「**NTOUCSE 1131 程式設計 1C**」
  https://www.facebook.com/groups/486362080792668
- 課程/實習**助教**: 邱敏傑、朱庭君、龔浩慈、陳品睿、岳紀伶

課本與參考書籍

- 課本:
  - 洪維恩, C 語言教學手冊, 第四版, 旗標
- 參考閱讀:
  - C Programming: A Modern Approach, 2nd Ed., K. N. King, 2008

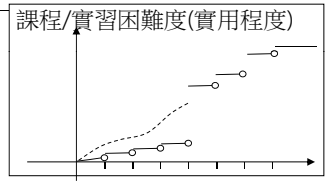
如果你已經有任何一本 C 的中文或是英文書籍相信應該都是足夠的，如果你擔心的話，可以拿過來我幫你看一下

課程目標

- 熟悉*程序式*程式語言的基本架構與計算機運算的基本模型
- 熟悉以 Top-down 方式分解程式的功能要求、切割問題、簡化問題，設計結構化、模組化的程式
- 以 C 語言為基礎，練習設計演算法解決程序複雜、需要耗費時間的運算問題

本學期課程大綱

1. C 語言、程式設計概觀、第一個 C 程式、與程式開發介面 (week 1)
2. C 的基本語法、關鍵字 vs. 識別字、程式碼的錯誤 (week 2)
3. 變數之概念、變數型態、資料表示法、與資料的轉換 (week 3)
4. 函數庫、格式化輸出與輸入函數 (week 4)
5. 運算式與運算子、運算子的優先順序、強制的資料型態轉換 (week 5)
6. 條件控制 (邏輯判斷與條件運算式) (week 6)
7. 結構化程式設計與迴圈 (week 7)
8. 區塊、函數、與參數傳遞 (week 7)
9. 期中考 (week 8)
10. 陣列與字串 (week 9)
11. 字串處理 (week 9, 10)
12. 指標運用 (week 10)
13. 遞迴 (week 11, 12, 13)
14. 使用者自定型態與基礎資料結構 (week 14)
15. 指標與動態記憶體使用 (week 15)
16. 期末考 (week 16)
17. 檔案輸出與檔案輸入 (week 17)
18. 大型程式開發、位元處理、物件導向設計 (week 18)



課程評量與要求

- Office Hour: 星期二、三、五 11:00 – 13:00, 其它時間請 email/fb messenger/discord 另約
- 課程評量方式

期中考	30%
期末考	30%
實習	30%
課程參與	10% (簽到/隨堂考試/上課回答/助教詢問/ Office Hour / 補強/ fb線上/ fb群組或email提問/ discord線上)

程式設計課程和其它課程的差別

- 程式設計是一種技術, 控制電腦用它的速度和準確性幫你完成你指派的工作,你需要**不斷地練習**把情境**抽象化**, 把資料**抽象化**
- 電腦能夠用很快的速度遵循指示來運作, 沒有特別的思考與推理邏輯, 所以聰明的你需要多負擔一些責任, 你需要練習到可以組合完全正確的語法下命令並且可以完全預期電腦在執行這個命令時的動作
- **不要**把它當成一般的科目來學習, 很多科目考前看一看都有一定的效果, 會了七成大概有七成的成績; 但是這個科目比較特別, 如果你只會七成, 記住了某些片段, 知道一些大概的法則, 基本上你寫不出正確運作的程式, 甚至你知道了九成, 也和能夠靈活運用有一段很大的距離
- 程式沒有辦法執行出你希望的結果, 常常是因為某一個語法你以為電腦會有 X 的效果但是實際上它卻有 Y 的效果, 某一個敘述你預期的效果和實際上電腦執行時的動作有落差, 你需要不斷地使用電腦寫程式, 觀察電腦執行的結果來驗證你所看到或認知的概念

聰明的漢斯 (Clever Hans)

1891

- 會算術的馬

  很多人不信, 想了很多方法來測試牠, 牠都[illegible]
  尤其是透過牠的訓練師提問, 回答更是絕對[illegible]
  換成其他人來提問, 漢斯還是能夠答對, **直到**
  提問者<u>不曉得答案</u>或是**漢斯<u>看不到</u>提問的**
  漢斯就再也無法答對了

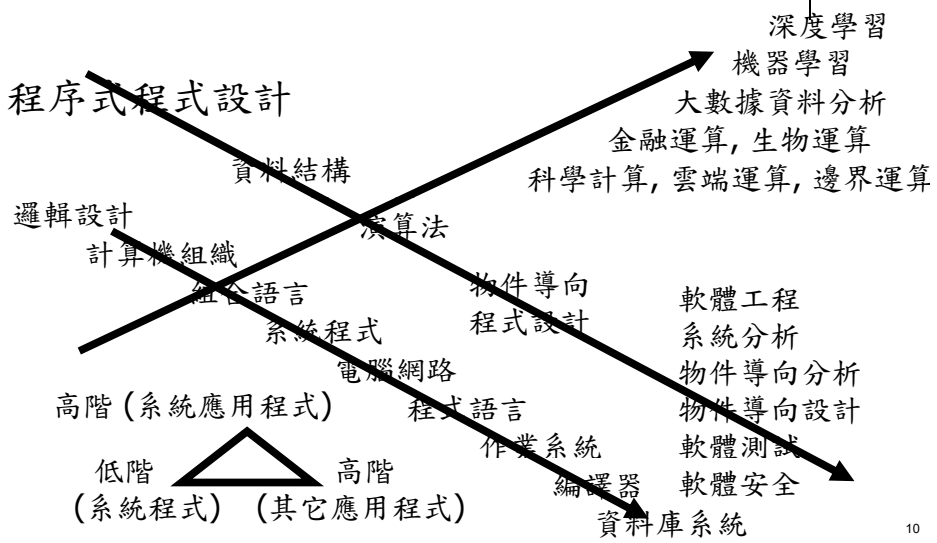
檢疫犬、緝毒犬

- 原來漢斯辨識出提問者<u>細微的肢體語言</u>與答案的關聯, 據以回答, **也許比會算術還更聰明**

差別 (cont'd)

- 常常看到有些同學很辛苦地完成實習和作業, 可是拿到一個新的問題時, 要不無從下手, 要不寫出來的程式錯誤百出, 練習寫程式的時候最怕: ❶**由別人已經對了的程式開始改** (本來應該遇見的錯誤都已經被同學解決了), ❷遇見程式錯誤的時候, 匆忙透過同學、助教或是Google尋找**替代方法避開錯誤**, (你遇到的每一個錯誤都是你**校正自己認知的關鍵**, 跳過去也許可以快速地完成作業, 可是下一次在類似的情境下你還是會遇見相同的錯誤)
- 遇見程式錯誤的時候, 最重要的是去**仔細觀察程式的表現**, 解釋為什麼某一個寫法會得到那樣的錯誤表現, 修改自己對那一個語法的認知與預期, 以後才能夠正確使用; 當然幫同學尋找程式錯誤的時候, 也要去針對目前程式的動作解釋, 不要為了速戰速決就直接拿另外一段程式來取代, 說 "反正這樣寫就會對", 這會讓小問題變大問題
- 上課講的概念與方法需要練習, 需要**自己在電腦上操作**, 才不會因為自己 "太多的**以為**" 或是誤會我的解釋或是誤會課本的解釋而遇見學習瓶頸, 這個課程幫大家準備了實習的題目和作業, 希望大家能夠按部就班地練習; **尤其是剛開學這一個月**, 不要因為比較忙而疏忽了

軟體設計與相關應用的學習過程



運用投影片上課

- 雖然教室前面要關燈, 午餐吃飽很好睡, 可是我們可以多一點時間來解釋投影片裡的內容, 可以多一些時間和同學互動
- 你發現我講的東西和先前講的不一致的時候, 發現我講的東西模稜兩可時, 請一定要隨時打斷我, 我可以很快回到前面的投影片跟你解釋, 不會跟你雞同鴨講隨便打發你
- 雖然速度快, 好像讓你沒有辦法抄筆記, 可是為什麼不在課程網站上下載投影片到平板, 或是印出來, 在上面註記呢…另外基本內容和課本一致, 你也可以考慮在課本上註記
- 上課難免和同學有一點點討論, 不過請盡量別影響其他同學, 真的精神很不好時, 考慮一下站起來到教室外面走一走…

Computer Science --- Coding

- **Programming** for a computer scientist is like *riding a horse* for a cavalry officer: It's not what the job is *about*, but if you can't do it, it will be a constant source of problems, will blind you to practical issues you need to consider, and you will look ridiculous.

Patrick Winston

1943-2019

AI 的時代不需要學寫程式了?!

- ChatGPT, CodeGPT
  - Based on a **large language model** with attention (transformer)
  - Magically performs better than a naïve programmer
  - Yann Lecun, Meta's AI chief , "Current artificial intelligence systems like ChatGPT do not have human-level intelligence and are not even as smart as a dog," June 2023
  - Demis Hassabis, CEO of Google DeepMind: "We're still not even at cat intelligence yet, as a **general** system," , July 2024
- coPilot – coding assistant

別急, 那個不需要 programmer 的時代還沒到, 還有得等

  - Robot vs. collaborative robot (cobot or co-robot)
  - Offers in-depth code suggestions for experienced programmers