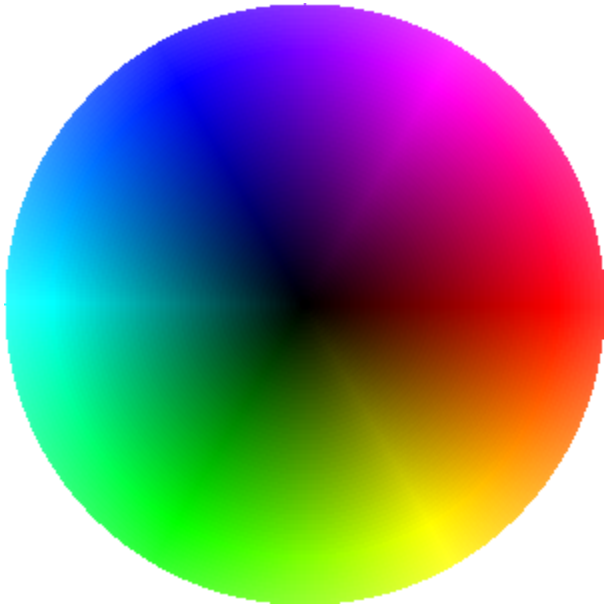
資訊、資訊、資訊

2015 資訊之芽語法班 suhorng

資訊科學與資訊工程



科學 $\Longleftrightarrow$ 工程像個光譜

~~好吧這個不是光譜~~

部份基礎 (看狀況)

- 演算法與資料結構（algorithm, data structures）
- 基本程式設計觀念
- 瞭解系統上到下的運作
  - 應用程式(常見關鍵字)
    - 視窗程式設計
    - 網路程式設計
  - 作業系統
    - 系統程式: 編譯器 連結器 assembler loader
    - 驅動程式
    - 基本的處理序管理 記憶體管理
    - 檔案系統
  - CPU / 記憶體
    - 快取設計 多核心CPU
    - 硬體知識 (e.g. 硬體描述語言), 形式驗證
    - 電路設計! 邏輯閘...

工程(與科學)

- 軟體工程，例如 design patterns
- 程式風格 元件設計 API 設計
- 產品開發管理(?)
- 前後端程式開發(我不熟這塊)

  #前端 樣式&版面設計, LESS, SASS, 程式 backbone.js anguler.js react.js ...

  #後端 伺服器 Nodejs, Ruby {Ruby on Rails}, PHP {CodeIgniter}, Python {Django, Tornado, ...}, Go, ...

  #資料庫 MongoDB, NoSQL, Neo4j ...

  #(?)端 Jade

科學(與工程)

- 影像處理、訊號處理、音訊、通訊（e.g. 無網）
- 機器學習、資料探勘、資訊檢索、（人工智慧？）
- 自然語言處理
- 機器人、電腦視覺
- 人機介面、普及計算
- 物連網？(工程?)
- 系統程式、作業系統、嵌入式系統
- 編譯器、計算機結構
- 資訊理論、程式語言、計算理論
- 平行計算、分散式系統、（雲端）
- ...

App 好像很紅, 不過我沒寫過

1. 會動就好?
2. 好的想法比較重要?
3. UI 跟 UX 設計
4. 能夠結合不同領域
5. 實作上對虛擬機有概念, 瞭解優缺點與雷區?

資訊與社群

- 一些會議
  - COSCUP
  - OSDC.tw (停辦: https://hcchien.hackpad.com/OSDC--39isdFVjjwc)
  - SITCON, Students' Information Technology Conference
- 各個語言的社群: Python, Ruby, Javascript, Haskell...
  - 各個語言的聚會/會議如 PyCon APAC/Taiwan
- 其他社群: http://g0v.tw/, ...
- 社群平台(?) GitHub https://github.com/

就算純粹只討論寫程式

- 語言與寫法...
  - 不同的編程典範
    http://en.wikipedia.org/wiki/Programming_paradigm :
    - procedural
    - object-oriented 物件導向
    - logic
    - functional
  - 不同的設計嘗試
    - generic programming
    - metaprogramming

C++ 是超級大坑, 你確定要跳?

- 僅僅學 "寫程式" 的概念不用把 C++ 學到很好, 也可學其他語言, C++ 其實設計也有頗多歷史遺跡&缺陷
- 沒教到的小東西: union（聯合）、reference（參考）、常用函式庫
- 沒教到的大東西
  - class，C++ 中物件導向的那一面
  - exception handling
  - namespace management
  - template (and metaprogramming!)
  - resource&ownership management
  - C++ 的各種 idiom 等等、C++11, C++14, C++17 ...
  - 每一項都是大坑, 很多很難弄懂的地方和地雷

你不會在任何課學到的 100 種 C++

```
auto curry = [](auto f) {
  return [f](auto x) { return [f,x](auto y) { return f(x,y); }; };
};
```

```
// http://cpptruths.blogspot.tw/2014/05/fun-with-lambdas-c14-style-part-2.html
template <class... F>
struct overload : F... {
  overload(F... f) : F(f)... {}
};

template <class... F>
auto make_overload(F... f) {
  return overload<F...>(f...);
}
```

```
// http://en.cppreference.com/w/cpp/thread/future
// future from a promise
std::promise<int> p;
std::future<int> f3 = p.get_future();
std::thread( [](std::promise<int> p){ p.set_value_at_thread_exit(9); },
             std::move(p) ).detach();
```

stackoverflow Questions Tags Users Badges Unanswered Ask Question

The Definitive C++ Book Guide and List

833

Reference Style - All Levels

1. *A Tour of C++* (Bjarne Stroustrup) The "tour" is a quick (about 180 pages and 14 chapters) tutorial overview of all of standard C++ (language and standard library, **and using C++11**) at a moderately high level for people who already know C++ or at least are experienced programmers. This book is an extended version of the material that constitutes Chapters 2-5 of The C++ Programming Language, 4th edition.

2. *The C++ Programming Language* (Bjarne Stroustrup) (**updated for C++11**) The classic introduction to C++ by its creator. Written to parallel the classic K&R, this indeed reads very much alike it and covers just about everything from the core language to the standard library, to programming paradigms to the language's philosophy. (Thereby making the latest editions break the 1k page barrier.) [Review] The fourth edition (released on May 19, 2013) covers C++11.

3. *C++ Standard Library Tutorial and Reference* (Nicolai Josuttis) (**updated for C++11**) *The* introduction and reference for the C++ Standard Library. The second edition (released on April 9, 2012) covers C++11. [Review]

4. *The C++ IO Streams and Locales* (Angelika Langer and Klaus Kreft) There's very little to say about this book except that, if you want to know anything about streams and locales, then this is the one place to find definitive answers. [Review]

C++11 References:

1. *The C++ Standard (INCITS/ISO/IEC 14882-2011)* This, of course, is the final arbiter of all that is or isn't C++. Be aware, however, that it is intended purely as a reference for *experienced* users willing to devote considerable time and effort to its understanding. As usual, the first release was *quite* expensive ($300+ US), but it has now been released in electronic form for $60US

2. *Overview of the New C++ (C++11/14) (PDF only)* (Scott Meyers) (**updated for C++1y/C++14**) These are the presentation materials (slides and some lecture notes) of a three-day training course offered by Scott Meyers, who's a highly respected author on C++. Even though the list of items is short, the quality is high.

Beginner

Introductory

If you are new to programming or if you have experience in other languages and are new to C++, these books are highly recommended.

1. *C++ Primer* * (Stanley Lippman, Josée Lajoie, and Barbara E. Moo) (**updated for C++11**) Coming at 1k pages, this is a very thorough introduction into C++ that covers just about everything in the language in a very accessible format and in great detail. The fifth edition (released August 16, 2012) covers C++11. [Review]

2. *Accelerated C++* (Andrew Koenig and Barbara Moo) This basically covers the same ground as the *C++ Primer*, but does so on a fourth of its space. This is largely because it does not attempt to be an introduction to *programming*, but an introduction to *C++* for people who've previously programmed in some other language. It has a steeper learning curve, but, for those who can cope with this, it is a very compact introduction into the language. (Historically, it broke new ground by being the first beginner's book using a modern approach at teaching the language.) [Review]

3. *Thinking in C++* (Bruce Eckel) Two volumes; second is more about standard library, but still very good

4. *Programming: Principles and Practice Using C++* (Bjarne Stroustrup) (**updated for C++11/C++14**) An introduction to programming using C++ by the creator of the language. A good read, that assumes no previous programming experience, but is not only for beginners.

* Not to be confused with *C++ Primer Plus* (Stephen Prata), with a significantly less favorable review.

Best practices

1. *Effective C++* (Scott Meyers) This was written with the aim of being the best second book C++ programmers should read, and it succeeded. Earlier editions were aimed at programmers coming from C, the third edition changes this and targets programmers coming from languages like Java. It presents ~50 easy-to-remember rules of thumb along with their rationale in a very accessible (and enjoyable) style. [Review]

2. *Effective STL* (Scott Meyers) This aims to do the same to the part of the standard library coming from the STL what *Effective C++* did to the language as a whole: It presents rules of thumb along with their rationale. [Review]

Intermediate

1. *More Effective C++* (Scott Meyers) Even more rules of thumb than *Effective C++*. Not as important as the ones in the first book, but still good to know.

2. *Exceptional C++* (Herb Sutter) Presented as a set of puzzles, this has one of the best and thorough discussions of the proper resource management and exception safety in C++ through Resource Acquisition is Initialization (RAII) in addition to in-depth coverage of a variety of other topics including the pimpl idiom, name lookup, good class design, and the C++ memory model. [Review]

3. *More Exceptional C++* (Herb Sutter) Covers additional exception safety topics not covered in *Exceptional C++*, in addition to discussion of effective object oriented programming in C++ and correct use of the STL. [Review]

4. *Exceptional C++ Style* (Herb Sutter) Discusses generic programming, optimization, and resource management; this book also has an excellent exposition of how to write modular code in C++ by using nonmember functions and the single responsibility principle. [Review]

5. *C++ Coding Standards* (Herb Sutter and Andrei Alexandrescu) "Coding standards" here doesn't mean "how many spaces should I indent my code?" This book contains 101 best practices, idioms, and common pitfalls that can help you to write correct, understandable, and efficient C++ code. [Review]

6. *C++ Templates: The Complete Guide* (David Vandevoorde and Nicolai M. Josuttis) This is *the* book about templates as they existed before C++11. It covers everything from the very basics to some of the most advanced template metaprogramming and explains every detail of how templates work (both conceptually and at how they are implemented) and discusses many common pitfalls. Has excellent summaries of the One Definition Rule (ODR) and overload resolution in the appendices. A second edition is scheduled for 2015. [Review]

Advanced

1. *Modern C++ Design* (Andrei Alexandrescu) A groundbreaking book on advanced generic programming techniques. Introduces policy-based design, type lists, and fundamental generic programming idioms then explains how many useful design patterns (including small object allocators, functors, factories, visitors, and multimethods) can be implemented efficiently, modularly, and cleanly using generic programming. [Review]

2. *C++ Template Metaprogramming* (David Abrahams and Aleksey Gurtovoy)

3. *C++ Concurrency In Action* (Anthony Williams) A book covering C++11 concurrency support including the thread library, the atomics library, the C++ memory model, locks and mutexes, as well as issues of designing and debugging multithreaded applications.

4. *Advanced C++ Metaprogramming* (Davide Di Gennaro) A pre-C++11 manual of TMP techniques, focused more on practice than theory. There are a ton of snippets in this book, some of which are made obsolete by typetraits, but the techniques, are nonetheless, useful to know. If you can put up with the quirky formatting/editing, it is easier to read than Alexandrescu, and arguably, more rewarding. For more experienced developers, there is a good chance that you may pick up something about a dark corner of C++ (a quirk) that usually only comes about through extensive experience.

Classics / Older

Note: Some information contained within these books may not be up-to-date or no longer considered best practice.

1. *The Design and Evolution of C++* (Bjarne Stroustrup) If you want to know *why* the language is the way it is, this book is where you find answers. This covers everything *before the standardization* of C++.

2. *Ruminations on C++* - (Andrew Koenig and Barbara Moo) [Review]

3. *Advanced C++ Programming Styles and Idioms* (James Coplien) A predecessor of the pattern movement, it describes many C++-specific "idioms". It's certainly a very good book and still worth a read if you can spare the time, but quite old and not up-to-date with current C++.

4. *Large Scale C++ Software Design* (John Lakos) Lakos explains techniques to manage very big C++ software

各種你可能會碰到的 open source project（library）

- 每天都在用的 g++/gcc 系列，以及威脅它的地位的 clang + LLVM
- OpenGL：寫小遊戲時已經碰到了，圖形處理函式庫
- OpenAL：音效處理函式庫
- OpenCL：heterogeneous computing
- OpenCV：影像（視覺）處理的函式庫
- OpenMP：平行程式設計
- C++ 很有名的函式庫：boost、Loki

其他語言

- Python、Ruby
- Rust、Go
- Java 統治世界
- 老牌的 Perl、很容易有漏洞的 PHP，etc

- 微軟的作品：C#、F# 等以 .Net 平台為後端(?)的語言

- 各種函數式（functional）語言...個人私心XD
    - Haskell, OCaml, Scheme, Racket, Scala, ...

- 新時代的組合語言，javascript，還有各種前端，CoffeeScript、LiveScript 等

- 前面很多語言看似類似(雖然有微妙的差別), 但有些東西看了可以改變你的一生與看法