前言

天啊，這本書竟是如此輕薄短小。我真想大叫一聲，哇歐！*C++ Primer* 加上索引、扉頁、謝詞之後，厚達 1237 頁，而此書卻薄薄只有 276 頁。套句拳擊術語，這是一本「羽量級」作品。

每個人都會好奇地想知道這究竟是怎麼回事。裡頭的確有一段故事。

過去數年來，我不斷纏著華德迪士尼電影動畫公司（Disney Feature Animation）的每一個人，要求讓我親身參與一部電影的製作。我纏著導演，甚至 Mickey 本人（如果我可以說出來的話），要求一份管理工作。我會如此瘋狂，部份原因是深陷於好萊塢大螢幕那令人神往的無盡魔力而難以自拔。除了電腦科學方面的學位，我還擁有藝術碩士的頭銜，而電影工作似乎可以爲我帶來個人專長的某種整合。我要求管理工作，爲的是從製片過程中獲取經驗，以便提供實際有用的工具。身爲一個 C++ 編譯器撰寫者，我一直都是自己最主要的用戶之一。而你知道，當你是自己軟體的主力抱怨者時，你就很難再爲自己辯護或覺得受到不公平的責難。

狂想曲兩千（*Fantasia 2000*）片中有一段火鳥（Firebird）特效鏡頭。其電腦特效指導對於我的加盟頗感興趣。不過，爲了掂掂我的斤兩，他要求我先寫個工具，讀入爲某段場景所攝的原始資料，再由此產生可嵌入 Houdini 動畫套件中的攝影機節點（camera node）。我當然用 C++ 順利把它搞定。他們愛死它了，我也因此得到了我夢寐以求的工作。

有一次，在製片過程中（在此特別感謝 Jinko 和 Chyuan），我被要求以 Perl 重寫那個工具。其它的 TDs 並非編程高手，僅僅知道 Perl、Tcl 之類的程式語言。（TD 是電影工業中的術語，指的是技術導演。我是這部片子的軟體 TD，我們還有一位燈光 TD [嗨 Mira]，一位模型 TD [嗨 Tim]，以及電影特效動畫師 [嗨 Mike, Steve, Tonya]。）而且，喔，天啊，我得趕著點，因爲我們想要獲得一些觀念上的實證，而導演（嗨 Paul 和 Gaetan）及特效總監（嗨 Dave）正等著這個結果，準備呈給公司大頭目（嗨 Peter）。這雖然不是什麼緊急要務，可是，你知道的…，唉。

這令我感到些許爲難。我可以自信滿滿地以 C++ 快速完成，但我不懂 Perl。好吧，我想，我去找本書抱抱佛腳好了 — 前提是這本書不能太厚，起碼此刻不能太厚。而且它最好不要告訴我太多東西，雖然我知道我應該知道每一樣東西，不過暫且等等吧。畢竟這只是一場表演：導演們需要一些經過證

實的概念，藝術家需要一些東西協助證實其概念，而製片（嗨 heck），她需要的是一天 48 小時。此刻我不需要全世界最棒的 Perl 大全，我需要的是一本能妥善引導我前進，並使我不致偏離正軌過遠的小書。

我找到了 Randal Schwartz 的 *Learning Perl*，它讓我立即上手並進展神速，而且頗具閱讀趣味。不過，就像其它有趣的電腦書籍一樣，它也略去了不少值得一讀的內容 — 雖然在那個時間點，我並不需要瞭解所有內容，我只需要讓我的 Perl 程式乖乖動起來。

我終於在傷感的心境中明白，*C++ Primer* 第三版其實無法扮演人們在初學 C++ 時的導師角色。它太龐大了。當然我還是認爲它是一本讓我驕傲的巨著 — 特別是由於邀請到 Josée Lajoie 共同完成。但是，對於想立刻學會 C++ 程式語言的人來說，這本巨著實在過於龐大複雜。這正是我動手撰寫本書的原因。

你或許會想，C++ 又不是 Perl。完全正確！本書也非 *Learning Perl*，它談的是如何學習 C++。真正的問題在於，誰能夠在散盡千頁篇幅之後，猶敢自稱教導了所有的東西呢？

1. 精細度。在電腦繪圖領域中，精細度指的是影像被描繪出來的鮮明程度。畫面左上角那位騎在馬背上的匈奴人，需要一張看得清楚眼睛的臉、頭髮、五點鐘方向的影子、衣服…。匈奴人的背後 — 不，不是那塊岩石，老天 — 唔，相較之下無關緊要。因此我們不會以相同的精細度來描繪這兩個影像。同樣道理，本書的精細度調降了相當程度。依我看，*C++ Primer* 除了在運算子多載化（operator overloading）方面的實例討論稍嫌不足之外，可說極其完備了（我敢這麼說是因爲 Josée 也有一份功勞）。但儘管如此，*C++ Primer* 還花了 46 頁篇幅加以討論，並附上範例，而這本書卻僅以 2 頁帶過。

2. 語言核心。當我還是 *C++ Report* 的編輯時，我常說，雜誌編輯有一半工作花在決定哪些題材應該放入，哪些不要。這句話對本書一樣成立。本書內容環繞在程式設計過程中所發生的一系列問題。我介紹程式語言本身的特性，藉此來爲不同的問題提供解決之道。書中並未述及任何一個多重繼承或虛擬繼承可解決的問題，所以我也就完全沒有討論這兩個主題。然而，爲了實作一個 iterator class，我必須引入巢狀型別（nested types）。Class 的型別轉換運算子很容易被錯用，解釋起來也很複雜，所以我不打算在書中提到它。諸如此類。我對題材的選擇以及對語言特性的呈現順序，歡迎大家指教批評。這是我的選擇，也是我的職責。

3. 範例的數量。*C++ Primer* 有數百頁程式碼，鉅細糜遺，其中甚至包括一套物件導向的文本檢索系統，以及近十個左右的完整 classes。雖然本書也有程式碼，但數量遠不及 *C++ Primer*。爲了彌補這項缺憾，我將所有習題解答都置於附錄 A。誠如我的編輯 Deborah Lafferty 所言，『如果你想提高教學速度，垂手可得的解答對於學習的強化，極有助益。』

本書的結構與組織

本書由七章和兩份附錄構成。第一章藉著撰寫一個具有互動性質的小程式，描繪 C++ 語言預先定義的部份。這一章涵蓋了內建的資料型別、語言預先定義好的運算子（operators）、標準程式庫中的 `vector` 和 `string` 類別、條件述句和迴圈述句、輸入和輸出用的 iostream 程式庫。我之所以在本章介紹 `vector` 和 `string` 兩個 classes，因爲我想鼓勵讀者多多利用它們取代語言內建的陣列和 C-style 字串。

第二章解釋函式的設計與使用，並針對 C++ 函式的多種不同風貌一一檢視，包括 inline 函式、多載化（*overloaded*）函式、function template，以及函式指標（pointers *to functions*）。

第三章涵蓋所謂的 Standard Template Library（STL）：一組容器類別（包括 `vector`, `list`, `set`, `map` 等等）、一組作用於容器身上的泛型演算法（包括 `sort()`, `copy()`, `merge()` 等等）。附錄 B 依字典順序列出最常被廣泛使用的泛型演算法，並逐一附上使用實例。

身爲一個 C++ 程式員，你的主要任務便是提交 classes 以及物件導向的 classes 階層體系。第四章帶領你親身走訪 classes 機制的設計與使用過程。在這個過程中，你會看到如何爲自身的應用系統建立起專屬的資料型別。第五章說明如何擴展 classes，使多個相關的 classes 形成族系，支援物件導向的 classes 階層體系。以我在夢工廠動畫電影公司（Dreamworks Animation）擔任顧問的經驗爲例，那時候我們設計了一些 classes，用來進行四個頻道影像合成之類的工作。我們使用繼承和動態繫結（*dynamic binding*）技術，定義影像合成所需的 classes 階層體系，而不只是設計八個相互獨立的 classes。

第六章的重頭戲是 class templates，那是建立 class 時的一種先行描述，讓我們得以將 class 用到的一個（或多個）資料型別或資料值，抽離並參數化。以 vector 爲例，可能需要將其元素的型別加以參數化。buffer 的設計不僅得將元素型別參數化，亦得將其緩衝區容量參數化。本章的行進路線圍繞在二元樹（binary tree）class template 實作上。

最後一章，第七章，說明如何使用 C++ 提供的異常處理機制（exception handling facility），並示範如何將它融入標準程式庫所定義的異常體系中。附錄 A 是本書習題解答。附錄 B 提供最被廣泛運用的一些泛型演算法的相關討論與使用實例。

關於原始碼

本書的所有程式，以及習題解答中的完整程式碼，皆可線上取得。你可以在 Addison Wesley Longman 的網站（`www.awl.com/cseng/titles/0-201-48518-4`）或我的個人首頁（`www.objectwrite.com`）中取得。所有程式皆在 Visual C++ 5.0 環境中以 Intel C++ 編譯器測試過，並且也在 Visual C++ 6.0 環境中以 Microsoft C++ 編譯器測試過。你或許需要稍微修改程式碼，才能在自己的系統上編譯成功。

如果你需要做任何修改並且也做了，請將修改結果寄一份給我（slippman@objectwrite.com），我會將它們附上你的大名，附於習題解答程式碼中。注意，本書並未顯示所有程式碼。

致謝

在這裡我要特別感謝 *C++ Primer* 第三版的共同作者 Josée Lajoie。不僅因爲她爲本書初稿提供了許多深入見解，更因爲她在背後不斷地帶給我鼓舞。我也要特別感謝 Dave Slayton 以他那犀利的綠色鉛筆，徹底檢視了文本內容與程式範例。Steve Vinoski 則以同情但堅決的口吻，爲本書初稿提供了許多寶貴意見。

特別感謝 Addison-Wesley 編輯小組：Deborah Lafferty，本書編輯，從頭到尾支援這個案子；Besty Hardinger，審稿編輯，對本書文字的可讀性貢獻最大。John Fuller，產品經理，帶領我們把一堆文稿化爲一本完整的書冊。

撰寫本書的過程中，我同時還擔任獨立顧問工作，必須兼顧 *Essential C++* 和客戶之間的事務。感謝我的客戶對我如此地體諒和寬容。我要感謝 Colin Lipworth, Edwin Leonard, Kenneth Meyer，因爲你們的耐心與信賴，本書才得以完成。

更多資訊

內舉不避親，我要推薦 C++ 書籍中最好的兩本，那便是 Lippman 與 Lajoie 合著的 *C++ Primer*，以及 Stroustrup 著的 *The C++ Programming Language*。兩書目前皆爲第三版。我會在本書各主題內提供其他更深入的參考書目。以下便是本書的參考書目。（你可以在 *C++ Primer* 和 *The C++ Programming Language* 找到更廣泛的參考文獻）

[LIPPMAN98] Lippman, Stanley and Josée Lajoie, *C++ Primer*, 3rd Editoin, Addison Wesley Longman, Inc., Reading, MA (1998) ISBN 0-201-82470-1.

[LIPPMAN96a] Lippman, Stanley, *Inside the C++ Object Model*, Addison Wesley Longman, Inc., Reading, MA (1996) ISBN 0-201-83454-5.

[LIPPMAN96b] Lippman, Stanley, Editor, *C++ Gems*, a SIGS Books imprint, Cambridge University Press, Cambridge, England (1996) ISBN 0-13570581-9.

[STROUSTRUP97] Stroustrup, Bjarne, *The C++ Programming Language*, 3rd Editoin, Addison Wesley Longman, Inc., Reading, MA (1997) ISBN 0-201-88954-4.

[SUTTER99] Sutter, Herb, *Exceptional C++*, Addison Wesley Longman, Inc., Reading, MA (2000) ISBN 0-201-61562-2.

排版方式（ㄆㄞˊㄅㄢˇㄈㄤ ㄕˋ）

本書文字字型爲 10.5 pt Palatino。程式碼和語言關鍵字爲 8.5 pt lucida。書中出現的識別字如果後面緊接著 C++ 的 function call 運算子（也就是一對小括號 ()），即代表某個函式名稱。因此，foo 代表程式中的某個物件，`bar()` 代表程式中的函式。各個 classes 的名稱以 Palatino 呈現。

譯註：繁體中文版的排版方式是：內文中的一般英文字爲 9 pt Times New Roman。程式碼和語言關鍵字爲 8 pt `Courier New`。各個 classes 的名稱亦爲 8 pt `Courier New`。異常類別（exception classes）以 8 pt *Lucida Sans* 呈現。英文長術語（例如 template parameter list, by reference, exception safe）採用 8 pt Arial。運算子名稱採用 9 pt Footlight MT Light。