前言

C++ Primer 的第二版和第三版之間，發生了一些不小的變動。最引人注目的是，C++ 有了國際標準，這不只爲語言本身增加了新特性如 exception handling, run-time type identification, namespaces, 內建 Boolean 型別, 以及新的轉型表示法，而且也大幅修改並擴充了原有的特性如 templates、*object-oriented* 和 *object-based* 程式方法所支援的 class 機制、巢狀型別、以及多載函式（overload function）的決議方式（resolution）。最具意義也最重要的或許是，Standard C++ 終於涵蓋了一個多面向的程式庫（library），其中包含原來的 Standard Template Library（STL）。新的 `string` 型別、相關並可聯合運作的 container 型別（如 `vector`, `list`, `map`, `set`）、以及可用以操作那些型別並容許擴展的泛型演算法（generic algorithms），都是這個新的 standard library 的一部份。除了新素材的增加，有些新的想法也對 C++ 程式的撰寫方式帶來影響。

不只 C++ 有如此的大變革，*C++ Primer* 第三版也是。

在此第三版中，不僅語言的處理方式有本質上的改變，作者也有了變動：我們的成員增加了一倍。此外，我們現在是國際化作業了（儘管我們其實都源自北美大陸）：Stan 是美國人，Josée 是加拿大人。這個雙人小組各自代表 C++ 的兩大族群：Stan 目前正在華德迪士尼動畫公司以 C++ 從事 3D 電腦繪圖和動畫應用，Josée 正從事 C++ 的定義和實作，並且是 C++ 標準委員會中的 Core Language 次委員會主席，她同時也是 IBM 加拿大實驗室的 C++ 編譯器小組成員之一。

Stan 是貝爾實驗室中與 Bjarne Stroustrup（C++ 創造者）共同合作的原始成員之一。他從 1984 年起就與 C++ 長相左右。Stan 參與了 cfront（最原始的 C++ 編譯器）的各個版本開發工作，從 1986 的 1.1 版到後來的 3.0 版，無役不與，並領導其中的 2.1~3.0 版。隨後，他參與了 Stroustrup 所領導的 Foundation Research Project 專案中的 Object Model 部份；整個專案的目標是要完成一個程式開發環境。

Josée 任職 IBM 加拿大實驗室的 C++ 編譯器小組已有 8 年之久。1990 年開始她成爲 C++ 標準委員會的成員之一。她擔任這個委員會的副主席三年，目前已連續擔任 Core Language 次委員會的主席有四年之久。

C++ Primer 第三版呈現的是一個大幅修訂版本，它所反應的不只是語言本身的改變與擴充，也反應出作者洞察力和經驗的變化。

本書架構

C++ Primer 對 Standard C++ 提供了廣泛的介紹。這是一本入門書籍，提供的是一個富思考性的 C++ 個人學習方針，但它並不是一本最簡單或最溫和的 C++ 語言書。C++ 語言的程式設計概念，諸如異常處理（exception handling）、container 型別、物件導向程式設計…等等，都呈現在字裡行間，隨時準備爲你解決某特定問題或程式工作。語言規則如「多載函式呼叫的決議方式」（resolution of an overloaded function call）或物件導向程式設計中所支援的型別轉換…等等，亦都有廣泛的探討。雖然某些高階主題對初學者而言難度頗高，但我相信，如果要培養對 C++ 語言的確實認識，這樣的涵蓋範圍有其必要性。面對這些素材，讀者應該不時回頭消化咀嚼，而不是一次就想把它囫圇下嚥。如果一開始你覺得有點無法接受，或是呼吸急促、口乾舌燥，那麼不妨暫時放下，稍後（日後）再回頭重看。我把這樣的章節以 A 標示出來。

閱讀此書並不需要以 C 語言做爲基礎。不過，熟悉某些結構化語言可以讓你前進得更輕鬆些。這本書是以「你的第一本 C++ 書籍」爲目標而寫，但它不應該是你的第一本程式設計書籍！當然，我總是從普通而基礎的語彙談起。事實上最初數章涵蓋的一些基本觀念如迴圈和變數等等，可能會讓某些讀者覺得太過淺白。不要擔心，有深度的東西很快便接踵而來 ☺。

C++ 的威力，絕大部份在於它支援新的程式設計方法，以及解決問題的新思維方式。因此，欲學習如何有效使用 C++，不能只是學習一組新的語法和語意而已。爲了幫助你做更大的學習，本書列有許多實例。這些例子不但用來介紹各種語言特性細節，也用來刺激讀者的思考。如果我們能夠從一個完整範例中學習到某個語言特性，我們就比較清楚這個特性爲什麼有用，也就比較清楚未來解決真實世界的問題時應該如何運用它們。此外，將焦點放在範例上，可以使本書初期所運用的觀念，在讀者的知識基礎建立起來之後，進行更完整的解釋。本書初期所舉的例子內含 C++ 基礎觀念的簡單運用，讓你在不需完整瞭解底層的設計細節和實作細節之前，可以先體會一下 C++ 的程式設計風貌。

第一篇意圖帶領你快速瀏覽 C++ 所支援的觀念和語言設施（language facilities），以及快速瀏覽撰寫並執行一個程式時所需要的基礎。第 1 章和第 2 章對整個 C++ 語言做了一個完整的簡介。讀過這

些章節之後，你應該對 C++ 的性質有了一些認識，但是還談不上徹底的瞭解（那正是其餘各章的目的）。

第 1 章帶領我們認識 C++ 語言的基本元素：內建資料型別、變數、算式（expressions）、述句（statements）以及函式。其中有一個極小而合法的 C++ 程式，概要探討了程式的編譯過程，讓我們瞭解所謂的前置處理器（preprocessor）以及輸入和輸出。本章另有數個簡單而完整的 C++ 程式，讀者可試著編譯並執行之。第 2 章透過 class 機制，介紹 C++ 對以物件爲基礎（*object-based*）和物件導向（*object-oriented*）程式設計的支援，並以一個陣列抽象體的演化來說明。此外它也概略介紹了 templates, namespaces, exception handling, standard library 所支援的泛型容器（general container types）和泛型程式設計（generic programming）。**這一章的配速有點快，部份讀者可能會覺得無法負荷**。如果是這樣，建議你不妨跳著讀，稍後再回頭仔細看。

C++ 的本質是提供各式各樣的設施，讓使用者得以定義新的資料型別，具備「與原有內建型別相同」的彈性和易用性，以此方式來擴充語言本身。支配語言的第一個步驟就是要先能夠瞭解語言的基本性質。第 3 章至第 6 章（第二篇）在這個層面上介紹 C++ 語言。

第 3 章介紹 C++ 語言預先定義好的內建資料型別和複合資料型別，以及由 C++ standard library 供應的 `string`, `complex`, `vector` 等 class 型別。這些型別架構出程式的基石。第 4 章對算式（expressions）提供了詳細的討論，像是算術運算式、大小比較運算式、指派動作等等。第 5 章討論述句（statements），那是 C++ 程式中的最小獨立單元。第 6 章焦點放在 standard C++ library 所供應的 container 型別。這一次不再是一段段小小的動作示範，我們要歷練一個文字查詢系統的實作過程，以此說明 container 型別的設計和運用。

第 7 章至第 12 章（第三篇）把焦點放在 C++ 所支援的「以程序爲基礎（*procedural-based*）」的程式設計方法上。第 7 章介紹 C++ 函式機制。函式（functions）封裝了一組動作，一般而言它們完成單一任務，像是 `print()`。我以「符號名稱之後加上一對空的小括號」代表一個函式。第 8 章探討變數的 scope（生存空間）和 lifetime（生命週期）觀念，以及 namespace（命名空間）。第 9 章延伸第 7 章對函式的討論，進一步介紹所謂的函式多載化（function overloading），這可以讓相同演算行爲的多個不同的函式實體共享同一個名稱。例如，我們可以定義一堆 `print()` 函式，用來輸出不同型別的資料。第 10 章介紹 function template 並示範其運用。所謂 function template 係提供一種預先表述，以備編譯器根據不同的參數型別，自動產生無窮多組函式實體，而函式實作碼（演算法）本身並沒有改變。

C++ 支援 exception handling（異常處理）設施。所謂 exception 代表的是一個非預期的程式行爲，例如「所有可用的記憶體耗盡」。發生 exception 的程式段落，應該丟出（*throw*）一個 exception，然後程式中的某些函式應該捕捉（*catch*）這個 exception 並執行必要的處理。Exception handling 的相關討論橫跨兩章，第 11 章利用一個「捕捉及丟出 class 型別之 exception」的簡單例子，介紹異常處理的基本語法和運用。由於我們的程式所處理的實際 exceptions，通常是物件導向 class 階層體系中的某個 class objects，所以我會在第 19 章再繼續討論如何丟出及處理 exceptions（那時我已介紹過物件導向程式設計觀念）。

第 12 章介紹 standard library 供應的各種泛型演算法（generic algorithms），並檢視它們如何與第 6 章的 container 型別以及語言內建的陣列型別互動。這一章首先以一個運用泛型演算法的簡單程式做爲起點。第 6 章提過的 iterators 在本章有更進一步的探討，它們像一種膠著劑，將泛型演算法和實際的 containers 繫結起來。本章也說明並示範 function object 的觀念，這個東西允許我們對泛型演算法所使用的運算子（例如 equality 運算子或 less-than 運算子）提供另一種語意。本書附錄有各演算法的詳細介紹及使用範例。

第 13 章至第 16 章（第四篇）專注在以物件爲基礎（*object-based*）的程式設計身上 -- 也就是 class 的定義和運用，以求產生出獨立的抽象資料型別。利用「產生新型別以描述題域（problem domain）」的方式，C++ 允許程式員撰寫應用程式時不必太在意各式各樣的煩瑣簿記工作 -- 那會使程式工作沉悶而令人生厭。程式所需的基本型別可以實作一次然後重複使用，讓程式員把心力集中在問題身上，而不是在實作細節身上。「將資料封裝起來」的設施可以戲劇性地簡化後續的維護工作和程式演化流程。

第 13 章專注於一般的 class 機制：class 的定義、資訊的隱藏（也就是將公開的 class 介面和私有的實作碼分隔開來）、class object 的定義和操控。第 13 章並討論 class scope、nested classes，以及如何以 classes 做爲 namespace 的成員。

第 14 章詳細描述 C++ 對於 class objects 的初始化（initialization）、解構（destruction）、指派（assignment）等動作的特殊支援。針對這些動作，C++ 所依恃的是一種特殊的 member functions，分別稱爲 constructor, destructor, 和 copy assignment 運算子。我們也會看到 memberwise initialization 和 memberwise copy 這兩個主題，其主旨是以一個 class object 做爲「隸屬同一 class」之另一個 object 的初值或新值。我們還會看到，爲了讓 memberwise initialization 和 memberwise copy 更有效率，編譯器支援的 named return value（NRV）最佳化動作。

第 15 章的主題是與 class 相關的運算子多載化（operator overloading）動作。首先呈現一般觀念和設計考量，然後便針對特定的運算子如 assignment, subscript, call, 以及 class 專屬的 new 運算子和 delete 運算子做討論。本章也討論了所謂的 friend class（它有特殊的存取權限），以及它的必要性。

本章也討論了使用者自定的轉換行爲，包括底層觀念以及一個廣泛的運用實例。本章還細部討論了函式多載化的決議行爲（function overload resolution），並以程式實例附上廣泛的說明。

第 16 章的主題是 class templates。所謂 class template 是一種用來產生 class 的預先描述體，其定義式內的一個或多個型別或數值，都被參數化了。例如 `vector` class，就是將其元素型別給參數化了。做爲儲存緩衝區的 class，可參數化的東西不只是元素型別而已，還包括緩衝區的大小。在較爲精巧的運用中，例如在分散式電算環境（distributed computing）中，其 IPC（多工行程通訊）介面、定址介面、以及同步控制（synchronization）介面，都可能被參數化。本章的討論包括如何定義一個 class template，如何產生特定型別的 template 實體，如何定義一個 class template 的成員（包括 member functions, static members, 以及 nested types），以及如何對使用 class template 的程式進行檔案編組。其中亦包含了一個廣泛的 class template 實例。

第五篇的 17, 18, 19, 20 四章主題放在物件導向（*object-oriented*）程式設計以及 C++ 對此的支援設施上面。第 17 章介紹一個用以支援物件導向程式設計的最主要設施：繼承（inheritance）與動態繫結（dynamic binding）。在物件導向程式設計中，parent/child 的關係（或說 type/subtype 的關係）係用來定義「分享共同行爲的各個 classes」。一個 class 可以繼承其 parent class 的資料和行爲，而不需要重新實作出共同特徵。child class（或說 subtype）只需負責它與 parent class 之間的差異即可。例如我們可以定義一個 parent `Employee` class 和兩個 children：`TemporaryEmpl` 和 `Manager`。這些子型別（subtypes）繼承了 `Employee` 的所有行爲，它們只需再完成自己的獨特行爲即可。

繼承的第二個概念，稱爲多型（polymorphism），也就此是「以一個 parent type 取用其所衍生的任何一個 subtypes」的能力。例如 `Employee` 可以表現自己所屬型別，亦可表現 `TemporaryEmpl` 或 `Manager`。所謂動態繫結（dynamic binding）便是「根據 polymorphism object 的真實型別，在執行時期決議由哪一個操作行爲起而執行」的能力。在 C++ 中，此係透過虛擬函式機制來完成。

第 17 章介紹物件導向程式設計的基本性質。此章完成一個 `Query` class 階層體系的設計和實作，用來支援第 6 章的文字查詢系統。

第 18 章介紹比較複雜的繼承體系：多重繼承與虛擬繼承。此章將第 16 章的 template class 實例加以擴充，成爲一個運用多重繼承和虛擬繼承的三層 class template 體系。

第 19 章介紹 run-time type identification（RTTI，執行時期型別確認）設施。RTTI 讓我們的程式得以在執行期間查詢一個具多型性質的（所謂 polymorphic）class object 的真實型別。例如我們可以查詢一個 `Employee` object，看它是否其實表現的是一個 `Manager` 型別。此外，本章重新檢討 exception

handling，討論 standard library 的 exception class 階層體系，並示範如何定義和處理我們自己的 exception class 階層體系。本章並提供一個深度觀察，看看在繼承情況下，編譯器如何支援多載函式的決議過程。

第 20 章詳細說明 C++ iostream input/output library 的使用。本章提供說明並示範資料的一般性輸出和輸入、定義 class 專屬的 input/output 運算子實體、認知及設定 input/output 狀態、以及資料的格式化。iostream library 是一個 class 階層體系，以虛擬繼承和多重繼承實作出來。

本書最後以一份附錄做爲結束。附錄中對每一個泛型演算法（generic algorithm）都提供有討論和程式實例，並以字母順序排列，以便查閱。

最後我要說，當一個人寫了一本書，他決定略去什麼東西，往往和他決定涵蓋什麼東西一樣重要。這個語言的某些風貌，例如 constructors 的運作細節、什麼條件下編譯器會產生內部暫時性物件、對效率的廣泛考量，雖然對於寫出真正有用的應用程式很是重要，卻不很適合一本語言入門教本。在進行 *C++ Primer* 第三版之前，Stan 寫了一本 *Inside the C++ Object Model*（請看稍後所附的參考書目中的 [LIPPMAN96a]），涵蓋大部份這一類題材。通常當我們推測讀者可能希望獲得更進一步的討論（特別是針對 object-based 以及 object-oriented 程式設計）時，本書會參考 *Inside the C++ Object Model* 中的相關解釋。

C++ standard library 中的某些部份被我刻意略去，例如對於國別（本土，locale）的支援，以及數值運算函式。C++ standard library 的範圍非常廣泛，如果要呈現它的所有風貌，會超過這本入門書的預設範疇。稍後所附的參考書目中，有一些書籍對此 library 有較詳細的討論，例如 [MUSSER96] 和 [STROUSTRUP97]。我相信一定會有許多展現 C++ standard library 各色風貌的各種書籍，在本書問世之後蓬勃出現。

第三版的變化

第三版的變化主要分爲四大部份：

1. 涵蓋語言新增特性：exception handling, run-time type identification, namespaces, 內建的 `bool` 型別，以及新式轉型（cast）表示法。

2. 涵蓋新的 C++ standard library，包括 `complex` 和 `string` 型別，`auto_ptr` 和 `pair` 型別，各種循序容器（sequence container）型別和聯合容器（associative container）型別（主要是 `list`, `vector`, `map` 和 `set`），以及泛型演算法（generic algorithms）。

3. 調整原來的文字，反應出 Standard C++ 對原語言性質的精鍊、修改、擴充。舉一個語言精鍊的例

子，我們現在可以前置宣告（forward declare）一個巢狀型別（nested type），這原本是不允許的。至於語言性質改變的實例是，虛擬函式在 derived class 中的函式實體可以傳回一個型別，該型別以 public 的方式衍生自 base class 的虛擬函式實體所傳回的型別。這項改變支援一種 class 操作型式，即所謂的 clone method，或說 factory method （17.5.7 節對於 `clone()` 虛擬函式有舉例說明）。至於語言性質的擴充實例，則是能夠明白對一個 function template 指定一個或多個 template 引數。事實上 templates 被大幅擴充，幾乎成爲一種全新性質！

4. 加強對 C++ 高階特性的對待和組織方式，特別是對於 templates, classes, 以及物件導向程式設計觀念。Stan 從一個相對小型的 C++ 供應者族群（譯註：指貝爾實驗室），跳到一個代表一般大眾的 C++ 使用者族群（譯註：指迪士尼動畫公司），引發的一個副作用就是，他相信，愈是能夠深刻洞察問題，便能夠愈是高明地運用 C++ 語言。所以，在此第三版中，許多時候我們將焦點移轉到如何更適切地說明語言特性下的重要觀念，如何以最理想的方式來運用它，並指出潛在的陷阱以趨吉避兇。

C++ 的未來

本書發行時，ISO/ANSI C++ Standards 委員會已經完成了 C++ 國際標準化的技術工作。這份標準規格書應該會在 1998 夏季由 International Standardization Organization（ISO）出版（譯註：已於 1998/09/01 出版，編號 ISO/IEC 14882）。

Standard C++ 公佈之後，符合標準之 C++ 編譯器應該很快會推出。此一標準之公佈，使 C++ 語言的演化終於塵埃落定。這使得以 Standard C++ 完成之精巧複雜的 libraries 終得蓬勃發展，爲工業問題的解決更助一臂之力。從此，C++ 世界中的主要成長，我們預期會在 libraries 領域中出現。

一旦標準完成，所謂的標準委員會仍然會繼續運作（當然速度會緩和下來），爲使用者的各項申請提供服務。這將導至 C++ Standard 細微的澄清、解釋、修正。如有需要，國際標準每五年會修訂一次，將技術上的改變以及工業界的需求納入考量。

C++ Standard 公佈後的五年內會發生什麼事，沒有人知道。說不定廣被相關工業所使用的新的 library components 會被納入 C++ standard library。以目前來說，配合著 C++ Standards 委員會所完成的工作，C++ 的命運正掌握在其使用者的手上。

致謝

（中譯本 略）

第二版致謝

（中譯本 略）

參考書目（Bibliography）

以下書籍，有些影響本書之撰寫，有些則是我所推薦的 C++ 重要資料。

[BOOCH94] Booch, Grady, *Object-Oriented Analysis and Design*, Benjamin/Cummings, Redwood City, CA (1994) ISBN 0-8053-5340-2.

[GAMMA95] Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns*, Addison Wesley Longman, Inc., Reading, MA (1995) ISBN 0-201-63361-2.

[GHEZZI97] Ghezzi, Carlo, and Mehdi Jazayeri, *Programming Language Concepts, 3rd Edition*, John Wiley and Sons, New York, NY (1997) ISBN 0-471-10426-4.

[HARBISON88] Samuel P. Harbison and Guy L. Steele, Jr, *C: A Reference Manual, 3rd Edition*, Prentice-Hall, Englewood Cliffs, NJ (1988) ISBN 0-13-110933-2.

[ISO-C++97] Draft Proposed International Standard for Information Systems -- Programming Language C++ - Final Draft (FDIS) 14882.

[KERNIGHAN88] Kernighan, Brian W., and Dennis M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, NJ (1988) ISBN 0-13-110362-8.

[KOENIG97] Koenig, Andrew, and Barbara Moo, *Ruminations on C++*, Addison Wesley Longman, Inc., Reading, MA (1997) ISBN 0-201-42339-1.

[LIPPMAN91] Lippman, Stanley, *C++ Primer, 2nd Edition*, Addison Wesley Longman, Inc., Reading, MA (1991) ISBN 0-201-54848-8.

[LIPPMAN96a] Lippman, Stanley, *Inside the C++ Object Model*, Addison Wesley Longman, Inc., Reading, MA (1996) ISBN 0-201-83454-5.
侯俊傑譯《深度探索 C++ 物件模型》，碁峰 1998，ISBN 9575663179

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

[MEYERS98] Meyers, Scott, *Effective C++, 2nd Edition*, Addison Wesley Longman, Inc., Reading, MA (1998) ISBN 0-201-92488-9.

[MEYERS96] Meyers, Scott, *More Effective C++*, Addison Wesley Longman, Inc., Reading, MA (1996) ISBN 0-201-63371-X.

[MURRAY93] Murray, Robert B., *C++ Strategies and Tactics*, Addison Wesley Longman, Inc., Reading, MA (1993) ISBN 0-201-56382-7.

[MUSSER96] Musser, David R., and Atul Saini, *STL Tutorial and Reference Guide*, Addison Wesley Longman, Inc., Reading, MA (1996) ISBN 0-201-63398-1.

[NACKMAN94] Barton, John J., and Lee R. Nackman, *Scientific and Engineering C++, An Introduction with Advanced Techniques and Examples*, Addison Wesley Longman, Inc., Reading, MA (1994) ISBN 0-201-53393-6.

[NEIDER93] Neider, Jackie, Tom Davis, and Mason Woo, *OpenGL Programming Guide*, Addison Wesley, Inc., Reading, MA (1993) ISBN 0-201-63274-8.

[PERSON68] Person, Russell V., *Essentials of Mathematics, 2nd Edition*, John Wiley & Sons, Inc., New York, NY (1968) ISBN 0-132-84191-6.

[PLAUGER92] Plauger, P.J., *The Standard C Library*, Prentice-Hall, Englewood Cliffs, NJ (1992) ISBN 0-13-131509-9.

[SEDGEWICK88] Sedgewick, Robert, *Algorithms, 2nd Edition*, Addison Wesley Longman, Inc., Reading, MA (1988) ISBN 0-201-06673-4.

[SHAMPINE97] Shampine, L.F., R.C. Allen, Jr., and S. Pruess, *Fundamentals of Numerical Computing*, John Wiley & Sons, Inc., New York, NY (1997) ISBN 0-471-16363-5.

[STROUSTRUP94] Stroustrup, Bjarne, *The Design and Evolution of C++*, Addison Wesley Longman, Inc., Reading, MA (1994) ISBN 0-201-54330-3

[STROUSTRUP97] Stroustrup, Bjarne, *The C++ Programming Language, 3rd Edition*, Addison Wesley Longman, Inc., Reading, MA (1997) ISBN 0-201-88954-4.
葉秉哲譯《C++ 程式語言經典本》，儒林 1999，ISBN 9579815240

[UPSTILL90] Upstill, Steve, *The RenderMan Companion*, Addison Wesley Longman, Inc., Reading, MA (1990) ISBN 0-201-50868-0.

[WERNECKE94] Wernecke, Josie, *The Inventor Mentor*, Addison Wesley Longman, Inc., Reading, MA (1994) ISBN 0-201-62495-8.

[YOUNG95] Young, Douglas A., *Object-Oriented Programming with C++ and OSF/Motif*, 2nd Edition, Prentice-Hall, Englewood Cliffs, NJ (1995) ISBN 0-132-09255-7.