# 譯序（侯捷）

泛型程式設計（generic programming）和其第一份重要實作品 STL（Standard Template Library），對許多人而言並不陌生，但是對更多人而言卻非常遙遠。

自從被全世界軟體界廣泛運用以來，C++ 有了許多演化與變革。然而就像人們總是把目光放在豔麗的牡丹而忽略了花旁的綠葉，做爲一個廣爲人知的物件導向程式語言（Object Oriented Programming Language），C++ 的另一面 — 泛型編程思維 — 被嚴重地忽略了。不能說物件導向思維和泛型思維有什麼主從之分，但是紅花綠葉相輔相成，肯定能對程式開發有更大的突破。

泛型思維在 C++ 身上主要以 templates 及相關特性來表現。許多人以爲 template 是多麼高階的技巧，望之儼然；接觸後才發現，即之也溫。更深入研習泛型思維後又有另一層體會：其言也厲。

C++ templates 相關特性不難掌握，然而以此技術爲載具所發展出來的一套泛型程式庫：STL（現已納入 C++ 標準程式庫），背後卻另有一套嚴密邏輯。本書一語道破 STL 的精神：一個嚴謹的軟體概念（***concept***s）分類學。是的，這些 ***concept***s 和任何程式語言無關，但就像 OO 思維需要 OO 語言的協助才得以順利展現一樣，泛型思維需要泛型語言的協助，才得以順利展現。Ada 和 C++ 都支援泛型語法和語意，Java 也已跟進[1]。

換言之，泛型程式設計和 C++ templates 和 STL 不該併爲一談。然以現今發展看來，三者又幾乎被劃上等號。本書即以 C++ 爲載具，第一篇講述泛型思維的演進及 STL 的組織概念，第二篇整理 STL 所涵蓋的全部 concepts 的完整規格，第三篇整理 STL 所涵蓋的全部 components 的完整規格。

人們對於 STL 的最大誤解是效率。事實上 STL 提供的是一個不損及效率的抽象性。泛型編程和物件導向編程不同，它並不要求你透過額外的間接層來呼叫函式；它讓你撰寫完全一般化並可重複運用的演算法，其效率和「針對特定資料型別而設計」的演算法旗鼓相當。每一個演算法、每一個容器的操作行爲，其複雜度都有明確規範 — 通常是最佳效率或極佳效率。在接受規格書明定的複雜度之後，我想你不會認爲自己親手撰寫的碼，能夠更勝通過嚴格檢驗的世界通用程式庫。

[1] 請參考 "GJ : A Generic Java - java may be in for some changes". Philip Wadler, DDJ, Feb., 2000

人們對 STL 效率的誤解，有一大部份是把編譯期效率和執行期效率混爲一談了。的確，大量而巢狀地運用 templates，會導致編譯器在進行 template 引數推導（argument deduction）及具現化（instantiation）時耗用大量時間。但它絕不影響執行效率。至於對專案開發時程所帶來的影響，我要說，和 STL 爲我們節省下來的開發時間相比，微不足道。

STL 的嚴重缺點在於，它尚未支援 persistence（物件的永續性）。在良好的解決方案尚未開發出來之前，persistence 必須由使用者自行完成。

作者 Austern 在本書「前言」對泛型思維及其技術演化還有更多精譬討論，我建議你此刻（噢不，至少看完這篇序之後）立即翻看「前言」，一定能夠對整個技術的來龍去脈有所掌握。

永遠記住，面對新技術，工程師最大的困難在於心中的怯弱。To be or not to be, that is the question! 不要和哈姆雷特一樣猶豫不決，當你面對一個有用的技術，應該果斷。

我個人在元智大學開授一門「泛型程式設計」課程。這本書便是我指定給同學的高階參考書。本書在 (1) 泛型觀念之深入淺出、(2) STL 架構組織之井然剖析、(3) STL 參考文件之詳實整理 三方面有卓越的表現。可以這麼說，在泛型技術和 STL 的學習道路上，本書並非萬能（至少它不適合初學者），但如果你希望徹底掌握泛型技術和 STL，沒有此書萬萬不能 ☺

侯捷 2000.10.05 于新竹

jjhou@ccca.nctu.edu.tw
http://www.jjhou.com
http://jjhou.readme.com.tw

p.s. 在此我要對本書另一位譯者黃俊堯先生表示感謝與讚美。俊堯從學生時代起即大量運用 STL，對於其中的技術、演進、效率均有深刻體會，是本書最合適的翻譯人選。一般所謂合譯是「你譯一半，我譯一半，兩不相干」，這是我最詬病的一種方式。本書由俊堯完成全部初稿，再由我完成全部的技術檢閱與文字修潤。我們都希望將書籍做到完美，但人世間沒有完美。請上本書網站（見封底）觀看後續的討論、勘誤、程式範例。

p.s. 本中文版係根據原書第四刷（4th printing）製作。其間已修定前三刷之錯誤。翻譯方式採頁頁對譯，俾得以完全保留原文索引。

# 譯序（黃俊堯）

泛型程式設計（generic programming）的觀念已經有一段歷史了，而 STL（Standard Template Library）是目前最重要的一個實作品。自 ANSI C++ 考慮採納 STL 之後，STL 漸漸地吸引 C++ 程式員的目光。然而，時至今日，雖然已經有許多人開始使用 STL，能夠掌握 STL 及泛型程式設計精髓的人卻不多。

《泛型程式設計與 STL》是一本討論泛型程式設計並深入 STL 學理的書。Matthew H. Austern 和 STL 創始者 Alexander Stepanov 共事多年，以其豐富的 STL 發展經驗撰成此書，是當前最具威信的 STL 鉅著。如果你使用過 STL，本書可以釐清你的觀念，並解答你對 STL 的許多疑惑。如果你已經熟悉 STL，想要對 STL 抽絲剝繭一番，甚至擴充 STL，本書更是不可或缺。

STL 背後是一套嚴謹的軟體概念（***concept***s）分類學，STL 的組件便是以此設計的。它不只是一套功能強大的程式庫，同時也是一套程式設計思維方式。學習 STL，不單是學習如何使用一套類別程式庫，也是學習泛型程式設計的精神。本書的第一篇至爲重要，敘述泛型程式設計思維及 STL 組織概念。詳細閱讀並多加思考，有助於瞭解泛型程式設計及 STL 的設計哲學。第二篇是 STL 所涵蓋之所有 concepts 的參考手冊；第三篇整理 STL 所定義之演算法及各種組件規格，與第二篇交互參照。

然而，世界上不可能有完美的事物，有時我們還得對 STL 加以擴充，甚至自行撰寫新的程式庫。STL 輔以 C++ template，提供前所未有的彈性，在不失效率的情況下，讓你以現有的 STL framework 爲基礎來擴充 STL。

程式庫的設計，往往攸關整個軟體設計的優劣，並深深影響將來的維護成本。如果能夠學習良好的程式庫設計經驗，便能把我們推往軟體設計的更高層次。STL 是 C++ Standard 所採納的程式庫，可以做爲我們學習程式庫設計哲學的典範。作者以本書詳述了 STL 的設計哲學，對於通用性、高效率、可攜性、擴充性等相關議題都有豐富的討論，是很珍貴的資料。

## 致謝

對我而言，翻譯是個特別的經驗，能與侯捷大哥合作更是件難得的事。侯大哥對於國內電腦技術書籍的貢獻令人欽佩，對於著、譯、評工作的專業水準及嚴謹態度更是大家有目共睹。翻譯期間與侯大哥討論請教，實在受益匪淺。我是這本譯作的第一個受惠者，侯大哥是我首先要感謝的人。我也要感謝葉秉哲先生，他是國內少數幾位很早便熟悉並使用 STL 的人。他所發表的《話說 STL》三篇文章，帶我一窺 STL 堂奧，之後的往來討論更釐清了我許多觀念。這幾個月來，家人及女友對於我的鼓勵和關懷，是我努力的動力。感謝他們。最後，謝謝所有關心我的朋友。

黃俊堯 2000.10.6 于新竹

jyhuang@wizs.org