深入 C++ 系列

C++ STL 中文版

P.J. PLAUGER
ALEXANDER A. STEPANOV
MENG LEE
DAVID R. MUSSER
著

王昕 译

PH PTR

中国电力出版社
www.infopower.com.cn

深入 C++ 系列

C++ STL 中文版

P.J. PLAUGER
ALEXANDER A. STEPANOV
MENG LEE
DAVID R. MUSSER
著

王昕 译

中国电力出版社

内 容 提 要

本书对 C++ STL 进行了全面而深入的阐述。STL（标准模板库）是在惠普实验室中开发的，已纳入 ANSI/ISO C++标准。其中的代码采用模板类及模板函数的方式，可以极大地提高编程效率。本书由 P.J. Plauger 等四位对 C++ STL 的实现有着卓越贡献的大师撰写，详细讨论了 C++ STL 的各个部分。全书分为 16 章，其中的 13 章通过背景知识、功能描述、头文件代码、测试程序和习题，分别讲述了 C++ STL 中的 13 个头文件，其他章节介绍了 STL 中广泛涉及的三个主题——迭代器、算法和容器。本书附录列出了接口和术语表，最后列出了参考文献。

本书适合对 C++有一定了解的程序员及高等院校师生阅读。

图书在版编目（CIP）数据

C++ STL 中文版/（美）普劳格（P.J. Plauger）等著；王昕译.
-北京：中国电力出版社，2002.5
ISBN 7-5083-1058-6

Ⅰ.C… Ⅱ.①普…②王… Ⅲ. C 语言-程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2002）第 028461 号

著作权合同登记号 图字：01-2002-0709 号

本书英文版原名：The C++ Standard Template Library

中国电力出版社出版、发行
（北京三里河路 6 号 100044 http://www.infopower.com.cn）
北京地矿印刷厂印刷
各地新华书店经售
*
2002 年 5 月第一版 2002 年 5 月北京第二次印刷
787 毫米×1092 毫米 16 开本 34.5 印张 784 千字
定价 69.00 元

译 者 序

众所周知，C++是一门功能强大的编程语言，支持多种编程典范（paradigm），其间包括 PB（Procedure-Based）、OB（Object-Based）、OO（Object-Oriented）以及新近出现的 GP（Generic Programming）。

作为 GP 的第一个广为流传的实现，STL（Standard Template Library）自被 C++标准化委员会接纳以来，已经对整个 C++社区产生了极为深远的影响。程序员可以放心地使用 STL 所提供的常用数据结构和算法，避免“每次都重新发明一个轮子”的情况出现，有效地提高编程效率；再者，为了充分利用 STL 已有的资源，C++中 SL（Standard Library）的实现方式也有了很大的变化，一些常用的组件（如：IOStream、string 等）都提供了和 STL 交互的接口。

随着 STL 在 C++社区的逐渐普及，对于 STL 的教学也开始变得红火起来。在国外，早在 1995 年，就有 Alexander A. Stepanov（STL 之父）和 Meng Lee（STL 的第一个实现者之一）的文章面世，1996 年更有 David R. Musser（GP 理论的又一元老）所著的《STL Tutorial and Reference Guide》出版，之后更是出现了《The C++ Standard Library》（Josuttis，1999）、《Generic Programming and the STL》（Austern，1999，本书中文版将由中国电力出版社引进出版，详情请访问：http://www.infopower.com.cn）、《Effective STL》（Meyers，2001）、《The C++ Standard Template Library》（Plauger 等，2001）等大师级的作品。

而反观国内，在 2000 年前，市面上讲述标准 C++的书籍寥寥无几，而讲述 STL 的书籍更是有如凤毛麟角。这使得绝大多数 C++程序员对最新的 ISO/ANSI C++为开发提供的强大支持缺乏必要的了解，导致了诸如 list、binary_search 等常见数据结构和算法的重复实现，造成了人力物力的浪费。这不可不说是国内 C++教学的失败。

但是，从 2001 年开始情况有所好转。国内一些 ANSI C++学习的先行者开始在网络上向大家介绍 ISO/ANSI C++的一些新特性，以使国内的 C++学习能够跟得上 C++社区的最新发展，这其中自然少不了对 STL 的介绍。在这段时期内，出现了一些以 ANSI C++为主题的网站，如：STL 小组（http://www.smiling.com.cn/group/homepage.ecgi?group_id=11916）、dimension5 网站（http://www.dimension5.org ），还有 csdn（http://www.csdn.net ）讨论区的 C/C++板块等。另外，国内的各家出版社也开始有目的地从国外引入一些大师的著作（如我上面提及的四本大师级著作都将在近期出版中文译本），以更快地让国内的读者接触到来自国外主流的声音。

机缘巧合之下，我有幸从中国电力出版社得到了翻译由 P.J. Plauger、Alexander A.

Stepanov、Meng Lee 和 David R. Musser 四位大师所著的《The C++ Standard Template Library》一书的机会，从而得以更早地接触到大师们的言论。作为本书的译者，我有必要介绍一下四位作者的身份，以加重本书的砝码（古人有云：不看僧面看佛面，虽然我人轻言微，但这四人在国外可都是响当当的人物哦☺）。作为本书第一作者的 P.J. Plauger，对于国外 C/C++社区来说可是一个如雷贯耳的名字，他所著的《The Standard C Library》等书都早已成为经典著作，并且，目前使用人数最多的 C++编译器（MS VC++）中自带的 STL 就是出自他手，其实力可见一斑。第二作者 Alex Stepanov，则更是不得了，听听别人都叫他什么：GP 之父！哇……，STL 就是出自他那天才的思想，并且他还身体力行，连同本书的第三作者——Meng Lee，实现了 STL 的第一个实现（HP STL）。而本书的第四作者 David Musser，作为 Stepanov 的最初且时间最长的合作者，对 GP 理论的形成也称得上是功不可没。

由于 P.J. Plauger 亲自实现了一个商业化的 STL 版本，那么作为讲述 STL 实现的本书来说，则更显得意义非凡了。虽说，只要轮子好用就可以不用去管轮子是怎么做出来的，但适当地了解一些轮子制作过程中的知识，肯定可以帮助我们更好地使用这个轮子。在本书中，Plauger 把 STL 分成不同的部分，一个一个头文件地给我们讲述每一部分是如何做出来的，我们该如何去使用它们，然后给出了测试这些组件的完整代码，最后为了让读者加深对 STL 的理解，更是给出了一些颇具挑战性的习题[由于这些习题具有一定的参考价值，所以我们可能会在 C++view（http://www.c-view.org/）杂志上逐步对这些习题的解法给出一些讨论以及自己的方案，以方便大家对于 STL 的理解。] 从某种意义上来说，本书甚至可以改名叫做《STL 从入门到精通》。不过，金无足赤，本书也存在着一些小小的问题，因为是为了向读者展示一个简单的 STL 实现，本书给出的代码在不同程度上存在着异常安全性的问题[异常安全性，始终是 C++中的一个难点所在，目前对于这方面的最好探讨可以参见 Herb Sutter 所著的《Exceptional C++》（Sutter，1999，本书中文版将在近期内由中国电力出版社引进出版）或 GotW 网站（http://www.gotw.ca）]；而且在不少地方的做法与现有的 C++标准稍有出入（这也可能是由于 VC++对于 C++标准的支持不足的缘故吧）；另外可能也是最重要的一点是，本书对于读者的要求比较高，它假定读者对 C++中的一些比较高级的用法都有所了解，虽说我在翻译过程中尽量对这些地方给出一些注释，但这依然会使读者产生一些困惑（幸好据小道消息透露，最近国内应该会出版很多有关 C++的经典著作，因此这个问题在一段时期后也就自然会不成为问题了☺）。

最后，对于本书中文版的出现，我想感谢以下几个人，首先当然是本书的作者 P.J. Plauger 等人，感谢他们给我们带来了如此好的一本书，在翻译过程中，我曾经就书中的一些问题和 Plauger 互通过几次 email，更正了原书中的一些错误；另外就是来自于 brainbench 的 Andrei Iltchenko，对于本书代码的异常安全性问题，就是他在 comp.lang.c++.moderated 上提出来的；然后就是中国电力出版社负责该书编辑的刘江先

生、关敏女士、宋宏女士以及参与此书中文版出版的全体员工，感谢他们给予我机会翻译此书并最终出版；再接下来就是 jjhou，没有他的帮助，我也不会那么早就开始学习 STL；最后我要感谢的就是经常在网上和我讨论 ISO/ANSI C++以及 STL 的朋友们，包括 babysloth、bugn、myan 等人，和他们的讨论也使得我对 STL 有了更深层次的理解。另外，如果对本书有任何疑问的话，可以与我联系，我的 email 是：cber@email.com.cn。

王 昕

2002 年 5 月

序　言

标准模板库（Standard Template Library，简称 STL）是 ANSI/ISO C++ 语言的库的一个主要组成部分。它最初是惠普实验室（Hewlett-Packard Labs）的产物，开发者为 Alexander Stepanov 及 Meng Lee（参见 **S&L95**），主要基于早期的、由 Stepanov 和 David R Musser 两人完成的工作。（参见 **M&S87**、**M&S89** 及 **M&S94**。这里所有参考文献都列于本书后的“参考文献”中。）你将会发现在这个库中包含着对于 C++中的模板（template）的各种使用，其中大部分的使用都值得炫耀，并且具有良好的连贯性。确实，STL 已经开始有意义地改变着许多程序员编写 C++代码的方式。

本书展示了怎样使用 STL 中提供的模板类（template class）及模板函数（template function），一如 C++标准中所要求的一样（参见条款 20 及 23～26）。我们在此将 STL 视为标准 C++中所定义的庞大的库中一个相对独立的子集。C++中库的设计与 ANSI/ISO 标准 C（**ANS89** 与 **ISO90**）中的库设计一脉相承。因此，也可以将本书视为 P.J. Plauger 先前的两本书，《The Standard C Library》（**Pla92**）与《The Draft Standard C++ Library》（**Pla95**）的一个后续版本。几乎所有 C++程序员感兴趣的有关库的内容都可以在这三本书中找到。

C++标准

1998 年，C++标准正式通过，并且将在接下来的一段时间中保持一种稳定的状态[①]。它既是 ANSI 标准，也是 ISO 标准，也就是说，它既是美国的国家标准，也是一种全球通用的标准。作为标准化进程的一部分，整个标准 C++语言及库第一次完整地描述在一起。一个相对较晚加入这个标准化进程的事件是在 C++标准的草案中接纳 STL 为 C++标准的一部分。现在，标准 C++的编译器及库的各种实现也不过刚刚出现。也就是说，即使是很有经验的 C++程序员也将发现本书中有许多东西是全新的。

同样，在 1994 年 6 月被标准化委员会接纳为 C++标准草案的一部分之前，对 STL 的早期描述（至少早于 **S&L95**）仅限于一个相对狭窄的范围内。在被接纳的过程中，STL 本身又被重新组织并且在几个重要的环节做了修改。现在，在标准 C++中的 STL 已经不再是由惠普实验室开发出来的那套软件开发包，同时它与那些由不同的代理提供的强化版

① 一般的标准化文档在正式通过后的 5 年内将保持一种稳定的状态，期间即使需要对它进行修改，也要等到 5 年后再提交给标准化委员会进行表决。——译者注

本也不同。这也就是说，即使是那些有着早期的 STL 使用经验的程序员也可以从本书中获得新的知识。你将会在本书中发现对 STL 的完整描述，一如在 C++标准中指定的那样。

本书中还将至少告诉你一种实现 STL 的方法。书中提供了大概 6000 行经过测试、证明可行的代码，同时这些代码也被证实可以移植到许多 C++编译器上。实际上，这些代码在本质上和 Microsoft Visual C++、IBM Visual Age 或其他厂商所提供的 C++编译器中的 STL 代码几乎是相同的。我们仅仅是在排版及符号表达方面做了小小的改动，以使得这些代码在书中更具有可读性，同时将它们用来作为教学示例也比较方便。

作为对 C++库的扩展，书中提供的这些代码可以工作于任何其他 C++库上面（参见附录 A）。然而，当与一个完全符合 C++标准的库一起工作时，它表现得尤为出色。我们尽可能地去除了那些不具有可移植性或不可能被广泛利用的代码。那些过分依赖于 C++语言中最近才加入的特性的代码，如模板的部分特化（template partial specialization）等，有可能在一些编译器上导致问题的产生。你也可以在那些商业化的 STL 版本中发现对于这些特殊情况的各种不同的折衷处理。

无论如何，你都可以通过使用本书中提供的代码，获得一些对于如何使用由模板构成的库的宝贵经验，而使用模板库也已经开始成为 C++世界中的一个重要标准。同样重要的是，我们相信看完一个实际的 STL 实现将有助于你更好地理解和应用它。

这就引入了本书的另一个目的。除了给出 STL 的标准，以及实现它的可行代码之外，本书还可以作为一本如何使用该库的教程。你将发现在本书中存在着一些有用的背景信息，告诉你 STL 是如何演变成现在的这个版本的，使用它意味着什么，以及如何使用它。为此并不需要阅读和理解我们所提供的所有代码。哪怕是对于本书的粗略学习，也能给你带来不少有益的收获。不需要是老手才能从本书中获益，那些偶尔才显得比较老练的程序员将会发现这里所提供的信息简直就是无价之宝。

本书的目的不是教你如何写 C++代码。我们假定你已经了解了足够多的 C++知识，已经可以看懂简单的 C++代码。虽然在本书中出现的代码并不是那么简单，但不要怕，我们会解释其间所使用的各种技巧。

扩展 STL

本书的最终目的就是告诉程序员如何设计及实现对 STL 的扩展。STL 集合了大量的算法、数据结构以及编程技巧。然而，它并没有提供程序员可能需要的全部特性。相比而言，它只是提供了那些核心的、被广为使用的特性，并且描述了这些核心代码的编写规则。

当你掌握了这些规则后，就可以往 STL 中添加自己的算法，并且让它与已有的数据结构一同正常地工作。你也可以添加自己的数据结构，使之与已有的算法协同工作。通过将本书中提供的 STL 代码作为例子来

使用，你将很快学会如何以最少的新增代码来处理新的问题，并且发现这些新增的代码可以在以后的项目中重用。这也就是设计库的最终原因。

本书的结构

本书的结构看起来更像是 STL 代码本身的结构。C++标准中列出了大量的头文件，但只有其中的 13 个头文件中定义了 STL 中所有的模板。这 13 个头文件的每一个在本书中都有对应的一章。其他的章节则全面地介绍了 STL，并讨论了 STL 中三个涉及较为广泛的主题——迭代器（iterator）、算法（algorithm）和容器（container）。这些头文件中的大部分都有着适量的相关内容，因此也就引起了一些相关的讨论。有关它们的相关章节必然会导致讨论范围的扩大。

每章都以相同的模式讲述了一个头文件。一开始是一个简短的背景介绍小节，接着就是有关这个头文件内容的功能性描述，然后是对于如何更好地使用该头文件中提供的特性所给出的一些建议。再接着给出了组成头文件的 C++代码，并伴随有代码是如何工作的有关注释。我们还为每一个头文件给出了一个小小的测试程序，给出至少一个粗略的例子演示每一个定义的模板是如何使用的。

在每一章的最后给出了一些习题。如果本书是作为某个大学课程的课本的话，这些习题可以当作家庭作业来完成。大部分习题都是简单的练习，例如使用库，或是代码重写等。它们能够使人理解实现中的某些重点，或是展示了实现中的一些合理变更。那些较难的习题都标注出来了，它们可作为一个更大范围的项目的基础。那些自学本书的人可以将这些习题作为一种锻炼，以激发更多的思考。

代码

本书中提供的代码以及对这些代码的描述都是基于惠普公司被广泛使用的 STL 版本。这个版本包括如下的声明：

本书中所提供的代码都经过了各种程度的测试，能够正常工作于如下公司提供的C++编译器：Microsoft、IBM、Edison Design Group以及Project GNU。这些代码通过了Dinkum C++ Proofer和Dinkum Abridged Proofer所提供的STL测试，这两种工具是由Dinkumware公司开发的用于评估相应的库质量的工具。它还通过了由Perennial Software公司和Plum Hall股份有限公司所提供的各种版本的商业库验证套件的测试。我们尽力将出错情况最小化，但不能保证没有任何出错情况。

同时请大家注意，在本书中所出现的代码同样处于版权保护之下。它并没有放置在公共域中，也不具有共享性质。它不处于“copyleft”[②]协议的保护下，即与自由软件基金会（FSF）所提供的代码不一样。P.J. Plauger拥有这些代码的所有版权，Dinkumware公司视它们为商业化的基础代码而提供许可。

给定的C++编译器中所带有的STL代码可能与我们在本书中提供的代码在很多方面存在着不同。这是可以理解的，即便是那些基于本书中所提供代码的STL代码也会如此。C++中的各种方言（dialect）依然存在，尤其是在那些复杂的领域中（如模板处理）。随着时间的推移，这些不同肯定会被消除，为此我们应该感谢C++标准化进程的完成。但是在未来的某些时间，你应该将本书中提供的代码视为许多实现的典型。

致谢

惠普公司在引导STL发展到现阶段的形式的过程中起了极大的作用。我们极为感激他们的重要贡献。我们同样感谢惠普公司开发出最初的STL版本并将其作为免费库发放。他们的慷慨使得STL在C++社区中得以迅速传播。

在标准化进程的后期，Bjarne Stroustrup和Andy Koenig极力向ANSI委员会X3J16（现在是J16）和ISO委员会WG21建议将STL加入到C++标准草案中。如果没有他们的热情支持，本书也不可能出现，更不会以现在这种形式出现。

Matt Austern对本书的最终草案提出了很多有用的注解。我们也非常感谢他的建设性批评。

本书中的一些原始素材最初出现在月刊《The C/C++ Users Journal》

② copyleft也是由Project GNU提出来的一种版权方式，对应于传统的copyright（版权）。它允许用户自由地分发软件及其源代码，不必受到传统的版权的限制，但又对用户修改软件源代码的行为做了一定的限制，在这一点上与在公共域中不一样。——译者注

的“Standard C/C++”专栏中，此专栏由 P. J. Plauger 负责。我们同样感谢出版社能够让我们重复利用这些文章。

Geoffrey Plauger 协助了本书排版工作的完成。

P.J. Plauger 于马萨诸塞州，康科德城
Alexander Stepanov 于加利福尼亚州，帕洛阿尔托
Meng Lee 于加利福尼亚州，库珀蒂诺
David R. Musser 于纽约，Loudon ville

目　　录

第0章 简 介

背景知识

库（library）是一系列程序组件的集合，它们可以在不同的程序中重复使用。C++语言依照传统的习惯，提供了由各种各样的函数（function）所组成的库，用于完成诸如输入/输出、数学计算等功能。这其中的某些函数直接来源于标准C的库（如printf），其他的则专属于标准的C++库（如set_new_handler）。然而，不论其来源如何，库函数都遵照以下规则：接受一些符合预先指定类型的参数，返回一个特定类型的值或改变一些已有的值。设计一个能被广泛使用的C或C++库的一个重要组成部分就是，猜测出这些函数中最有可能出现的参数组合情况。

C++与C相比而言有着重大的改进。它将C中的数据结构的概念进行了扩展，使得它不但能包含成员对象，还能包含成员函数。C++中的类（class）可以更好地将一些相关的数据及在其上进行的操作封装在一起。设计一个能被广泛使用的C++库的另一个重要组成部分就是猜测出这些类中最有可能出现的成员对象类型的组合情况。

在其最新的发展过程中，C++新增了模板（template）这个特性。通过使用模板，C++允许推迟对某些类型的选择，直到真正想使用模板或者说对模板进行特化（specialize）的时候。可以定义模板类及模板函数。在一定的限制下，模板类或模板函数可以允许程序员针对不同的类型提取出特定的、一般化的代码为这些类型服务。正如你所想到的，模板库相对于传统的、由函数及类组成的库来说，可以提供更好的代码重用机会。当然，标准化委员会也不会忽视这些机会，所以它也被标准化委员会添加在C++标准中。

ANSI X3J16
ISO WG21

ANSI（American National Standards Institute，美国国家标准学会）负责制定计算机编程语言的美国标准。X3J16是经ANSI组织授权制定C++语言标准的标准化委员会的名称，它于1989年正式启动，现在我们称它为J16。ISO（International Organisation for Standardization，国际标准化组织）是一个与ANSI相似的组织，不过它负责的是制定全球化的标准。ISO于1991年成立了技术委员会JTC1/SC22/WG21，它协同X3J16一起工作，目的是制定出一个既符合ANSI要求，又符合ISO要求的C++标准。在本书中，“标准化委员会”同时指代X3J16/J16与WG21。在过去

的8年间，标准化委员会每年都举行三次会议，一次持续一周，每次会议都会对C++标准的制定过程产生重大的影响。

我们现在所见的C++标准的最后一次技术上的修改完成于1997年11月，其官方名称为ISO 14882。这意味着从现在起，这份文档将进入一个相对稳定的阶段——一般来说每一份经由ISO制定出来的标准都将在以后的5年间（至少是5年）被冻结[①]。然而，在这份草案被制定为标准前，标准化委员会还是对它做了一些较大的改动，扩充了其中有关库的部分条款。其中一个比较大的改动就是将模板引入标准库，使用模板类来代替传统的C++中定义的类。但是对标准库的最大的一次改动出现在1994年7月。在1994年7月，标准化委员会通过投票决定，将STL加入到C++的标准中。

STL

STL（Standard Template Library，标准模板库）是惠普实验室开发的一系列软件的统称。它是由Alexander Stepanov（现在AT&T实验室工作）、Meng Lee（仍在惠普实验室工作）和David R Musser（现在Rensselaer Polytechnic Institute工作）在惠普实验室工作时所开发出来的（S&L95和M&S96）。现在虽说它主要出现在C++中，但在被引入C++之前该技术就已经存在了很长的一段时间。实际上，Musser和Stepanov在十多年前就已经向世人展示了它的一个早期实现，那时他们用以完成STL的工具是Ada中的generics（这也是模板的一种形式，M&S87和M&S89）。这些工作也是基于他们两人在更早的时期所完成的一些工作。

如同我们在这里所展示的一样，惠普实验室最初的工作产生了超过6000行的、非常精致的代码。几乎所有的代码都采用了模板类及模板函数的方式。可以为这些模板指定各种类型来进行特化，这些类型可以是语言本身提供的内置（built-in）类型，也可以是自己所定义的类。通过指定类型对模板进行特化所产生的代码与自己所写的代码非常接近。而且对于STL中所提供的数据结构和算法，你也很难自己编写代码来正确地完成它们。因此我们可以尽情地享受STL带来的好处，使用它所提供的精巧的方案来解决常见问题，从而获得在代码空间及执行时间上的效率。

STL的代码从广义上大致分为三类：算法（algorithm）、容器（container）和迭代器（iterator）。在这些分类中，迭代器可能是其中最有趣、最有创造性的一类，不过也是最难解释清楚的一类。因此，我们在此先介绍算法和容器。

算法

严格地说，算法是用以决定结果的一个数学过程。它与数学中的函数在某些方面是有区别的。可能对于我们来说最重要的还是，算法

① 即不会被修改。——译者注