HZ BOOKS
华章科技

C++模板编程领域经典著作，深入讲解模板编程的基本原理、标准库中算法与容器等模板的实现原理、模板编程的高级技巧，以及C++11的模板新特性

资深C++开发工程师撰写，以透彻分析原理为前提，以实践为导向，能有效指导读者动手编写各类模板

實戰

温宇杰 著

C++ Template Programming in Practice

深入实践C++模板编程

机械工业出版社
China Machine Press

實戰

C++ Template Programming in Practice

深入实践C++模板编程

温宇杰 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

深入实践 C++ 模板编程 / 温宇杰著 . —北京：机械工业出版社，2013.6

ISBN 978-7-111-42754-4

I. 深… II. 温… III. C 语言 – 程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2013）第 116175 号

C++ 模板编程领域的经典著作，由资深 C++ 开发工程师撰写。本书以透彻分析原理为前提，深入讲解了模板编程的基本原理、标准库中算法与容器等模板的实现原理；以实践为导向，通过大量的模板向读者展示了如何使用模板进行编程以及如何编写自定义模板。除此之外，本书还总结了各种常用的模板编程技巧、C++11 标准中的模板新特性和新语法，以及 C++11 中新增的其他语言特性。

全书共 16 章，分为四部分：第一部分（第 1 ～ 4 章）首先介绍了模板编程的基本概念与用法，然后重点讨论了编译器对模板的具体实现方法及其局限，读者可以通过本部分内容理解模板的基本原理并自行实现简单的类模板与函数模板。第二部分（第 5 ～ 9 章）对标准库中的算法与容器的实现原理和用法进行了深入地剖析，读者通过本部分内容对标准库中的算法、迭代器与容器之间的关系有深入的理解，从而可以精确调节标准容器的行为，自行开发适用于标准算法的容器类模板。第三部分（第 10 ～ 13 章）讨论了模板编程的高级技巧，如模板编程中“概念”的设计、控制代码量的技术、编译期逻辑的控制以及元编程的基本方法等，读者可以通过本部分内容开发更具规模、更加智能的模板库，并利用元编程技术实现编译期的逻辑演绎与类型推导。第四部分（第 14 ～ 16 章）介绍了 C++11 标准中的新增语言特性，以及对模板编程的影响。

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）

责任编辑：白 宇

藁城市京瑞印刷有限公司印刷

2013 年 6 月第 1 版第 1 次印刷

186mm×240mm • 19.75 印张

标准书号：ISBN 978-7-111-42754-4

定 价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991 88361066 投稿热线：（010）88379604

购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

前　言

为什么要写这本书

笔者在工作中最常用的编程语言就是 C++。在最初接触 C++ 语言的时候，只当它是 C 语言与面向对象编程的组合，在写代码时也是积极实践“万物皆对象”的法则，不管什么操作，总要将其在某个对象的某个成员函数中实现才满意。但在长年的编程实践过程中，却常常对面向对象的设计方法产生怀疑。

面向对象的设计思想，是将与某个数据类型相关的操作与该数据类型捆绑在一起，构成一个整体。然而，有时候操作并不完全依赖于数据类型。同一种算法，可能适用于不同的数据类型。如果硬要遵循“万物皆对象”的法则，那么只能将其在各个具体数据类型中重复实现，或是设计一个抽象的类型专门容纳算法。无论怎样，总显得有些“削足适履”。

而在实践中最苦恼的事情是，虽然算法固定，但是只因数据结构不同，每当引入新数据时，只得将算法重写一遍，徒费人工。高手看到此，自然就会想到将算法写成模板。也正因有此困扰，笔者才开始正视 C++ 语言中的“模板编程”，一旦走入这套精密强力却又不广为人知的机制，方知此处别有洞天。

几番研究与实践下来，笔者尚不敢称对 C++ 的模板编程掌握几分，却早已叹于其对笔者编程思路革新起到巨大的帮助。从“万物皆对象”，到如今渐渐变为关注设计容器及抽象算法，这是拜模板所赐。赞叹之余，不禁想尝试写一本书介绍 C++ 中的模板编程，于人分享心得，于己则巩固琢磨。不论章节条理，只求娓娓道来。如能对读者略有裨益，便是幸甚。

读者对象

本书假定读者对 C++ 的语法有基本了解，要求仅此而已。笔者尽力将 C++ 语言中这一部分非常有趣的内容，用详尽的示例以及平实的语言展现在读者面前。即使是刚刚踏入 C++ 队伍的新兵，相信也能跟随本书的进度，渐渐领略这一片 C++ 秘境。而对于久经沙场的老兵，笔者限于自身阅历有限，不敢夸口定能有所提高，唯有以诚意将所思所得写下，请诸位老兵“择其善者从之，其不善者改之”。

本书面向了解 C++ 基本语法并有 C++ 编程经历的读者。对 C++ 模板编程技术、标准模板库用法和原理感兴趣的读者，可以将本书作为一本入门与提高的书使用。对于经常开发中小型 C++ 代码库的读者，本书中有关模板高级编程技巧的内容可为您提供有益的补充。另外，本书中有关 C++11 新标准与模板技术的内容，也可作为对新标准的概要介绍供您参考。

如何阅读本书

本书的内容分为四个部分。

模板基础（第 1 ～ 4 章）介绍模板编程的基本概念与用法，并着重讨论编译器对模板的具体实现方法及其局限。学习完本部分之后，读者可以理解模板的基本原理并自行实现简单的类模板与函数模板。

标准库中的模板（第 5 ～ 9 章）介绍标准库中的算法与容器，并对其实现细节进行仔细推敲。算法与容器可以说是 C++ 模板编程的最典型用法。C++ 程序员可以不知道模板编程技术，但不可能不用到由模板写成的标准算法与容器。了解其实现原理的重要意义不言而喻。学习完本部分，读者将能洞悉标准库容器的实现原理，对标准库中的算法、迭代器与容器之间的关系有深入的理解，从而可以精确调节标准容器的行为以及自行开发适用于标准算法的容器类模板等。

模板编程高级技巧（第 10 ～ 13 章）讨论模板编程中“概念”的设计、控制代码量的技术、编译期逻辑的控制以及元编程的基本方法等。学习完本部分，读者可以开发规模更大、更加智能的模板库，并利用元编程技术实现编译期的逻辑演绎与类型推导。

模板与 C++11（第 14 ～ 16 章）介绍新标准 C++11 中的重要改进及其对模板编程的影响。在学习完本部分之后，读者将能够准确把握 C++11 的几大重要革新的用意和用法，为利用新一代标准进行软件开发打下坚实基础。

勘误和支持

除封面署名外，本书编写工作的还有：陈凯、石少华、刘宏业、林文雯、王春、张敏、付强。由于笔者学识与经历有限，书中内容难免有偏颇及谬误之处。笔者将秉持谦虚的态

度和开放的胸襟，认真对待各位读者的批评与指教。书中的全部源文件可以从华章网站[⊖]下载，如您对本书内容有任何意见和建议，或者有关 C++ 和程序设计的任何话题，欢迎发送邮件至 yjwen.ty@qq.com 和通过新浪微博 @ 宇杰 W 与笔者交流。

致谢

感谢我的好友张哲君。作为本书草稿的第一位读者，你的意见与鼓励是我写作本书的动力之源。

感谢机械工业出版社华章公司的各位同仁为本书的出版付出的辛勤劳动。特别要感谢杨福川编辑和白宇编辑，本书从草稿到成书的过程，离不开你们的精心指导与耐心审校。

最深深的谢意送给我的母亲和我的妻子。没有你们在我身边一直默默地支持和付出，不会有提笔写作的我，也不会有本书存在。

最美好的祝愿送给我即将出生的孩子们。愿你们快乐成长，并拥有淡定从容的人生。

最后，借此机会，向 C++ 语言的发明者 BjarneStroustrup 教授、标准模板库的主要作者 Alexander Stepanov 和 Meng Lee 致以崇高的敬意。

温宇杰

⊖ 登录华章网站 www.hzbook.com 本书页面下载完整代码。

目　录

第一部分

模板基础

本部分内容：

- ❑ Hello 模板
- ❑ 类亦模板
- ❑ 模板参数类型详解
- ❑ 凡事总有“特例”

第 1 章　Hello 模板

C++ 可称为强类型语言，凡值必有类型，凡变量声明时必声明其类型，且变量类型“一生”不变。除少数预设类型外，不同类型值之间不能自动转换，除非有用户自定义的相关构造函数和类型转换函数。如此则编译时已知各变量所占内存大小，编译器可精确规划变量值在内存与寄存器间之调度，提高运行效率。而其弊端是代码无法独立于类型之外。对于某些算法，针对不同数据其操作过程完全一致，只因所操作数据类型不同，在 C++ 中需实现为不同的函数，难免重复。

1.1　为什么需要模板

以从一列值中找出最大值为例，其算法可概述为“设置一个变量 max_value，从列中选择某一元素值作为 max_value 初值；然后将 max_value 与列中其余元素一一比较，更新 max_value 为其中较大值，待全部元素比较完毕，则 max_value 的值即列中最大值”。以上叙述都不涉及具体元素类型，可见算法描述可脱于类型之外。

如果以某种动态类型语言，比如 Python，来实现此算法则非常简洁。在 Python 中，一个从序列中寻找最大值并返回的函数其代码可写成以下形式[⊖]：

```
def max_element(l):
  max_value = l[0]
  for elem in l[1:]:
    if elem > max_value: max_value = elem
  return max_value

print max_element([2, 0, 1, 1, 0, 8, 1, 8])
print max_element(['2011', 'August', '11', 'Thursday'])
```

Python 中变量类型随所赋值而变，因此函数 max_element 的参数 l 既可接纳整数列表，又可接纳字符串列表。又因为在 Python 中字符串默认按字典顺序比较大小，所以无论是整数列表还是字符串列表都适用上述代码。而函数中变量 max_value 既可接受“整数”值，也可接受“字符串”值。正因变量类型可变，同一函数可用于在各种列表中寻最大值。

但使用 C++ 实现一个通用的 max_element 函数就不那么简单。由于 C++ 中变量的类型不可变，不能像 Python 那样用同一函数、同一变量处理不同类型的列表。如果没有模板，则必须根据所操作列表类型确定函数参数以及内部各变量的类型。假如列表为一个整数数组，则函数代码可写成例 1.1 的形式。

⊖ Python 中已有内建函数 max() 实现此功能。