C++历史发展概述

1、三个阶段

要谈C++的发展史，就不得不谈一下，C语言。

C语言是**1972**年由美国贝尔实验室的D.M.Ritchie研制成功的。对于初学者来说，C语言并不友好，因为它是为计算机的专业人员而设计的，随着计算机的提升，软件复杂度跟需求功能的增多，而C语言作为面向过程的底层语言，要求在每一个细节都精确设计，因而用C来编写程序越来越困难，因而，面向对象的C升级版C++就诞生了。

C++语言发展大概可以分为三个阶段：

1. 第一阶段从80年代到1995年。这一阶段C++语言基本上是传统类型上的面向对象语言，并且凭借着接近C语言的效率，在工业界使用的开发语言中占据了相当大份额；
2. 第二阶段从1995年到2000年，这一阶段由于标准模板库(STL)和后来的Boost等程序库的出现，泛型程序设计在C++中占据了越来越多的比重性。当然，同时由于Java、C#等语言的出现和硬件价格的大规模下降，C++受到了一定的冲击；
3. 第三阶段从2000年至今，由于以Loki、MPL等程序库为代表的产生式编程和模板元编程的出现，C++出现了发展历史上又一个新的高峰，这些新技术的出现以及和原有技术的融合，使C++已经成为当今主流程序设计语言中最复杂的一员。

最初导致C++诞生的原因是在Bjarne博士等人试图去分析UNIX的内核的时候，这项工作开始于1979年4月，当时由于没有合适的工具能够有效的分析由于内核分布而造成的网络流量，以及怎样将内核模块化。同年10月，Bjarne博士完成了一个可以运行的预处理程序，称之为Cpre，它为C加上了类似Simula的类机制。在这个过程中，Bjarne博士开始思考是不是要开发一种新的语言，当时贝尔实验室对这个想法很感兴趣，就让Bjarne博士等人组成一个开发小组，专门进行研究。下图即C++之父Bjarne Stroustrup

当时不是叫做C++，而是**C with class**，这是把它当作一种C语言的有效扩充。由于当时C语言在编程界居于老大的地位，要想发展一种新的语言，最强大的竞争对手就是C语言，所以当时有两个问题最受关注：C++要在运行时间、代码紧凑性和数据紧凑性方面能够与C语言相媲美，但是还要尽量避免在语言应用领域的限制。

在这种情况下，一个很自然的想法就是让C++从C语言继承过来，但是我们的Bjarne博士更具有先见之明，他为了避免受到C语言的局限性，参考了很多的语言，例如：从Simula继承了类的概念，从Algol68继承了运算符重载、引用以及在任何地方声明变量的能力，从BCPL获得了//注释，从Ada得到了模板、名字空间，从Ada、Clu和ML取来了异常。

2、历史事件

下面让我们来一起看一下C++历史上的主要事件：

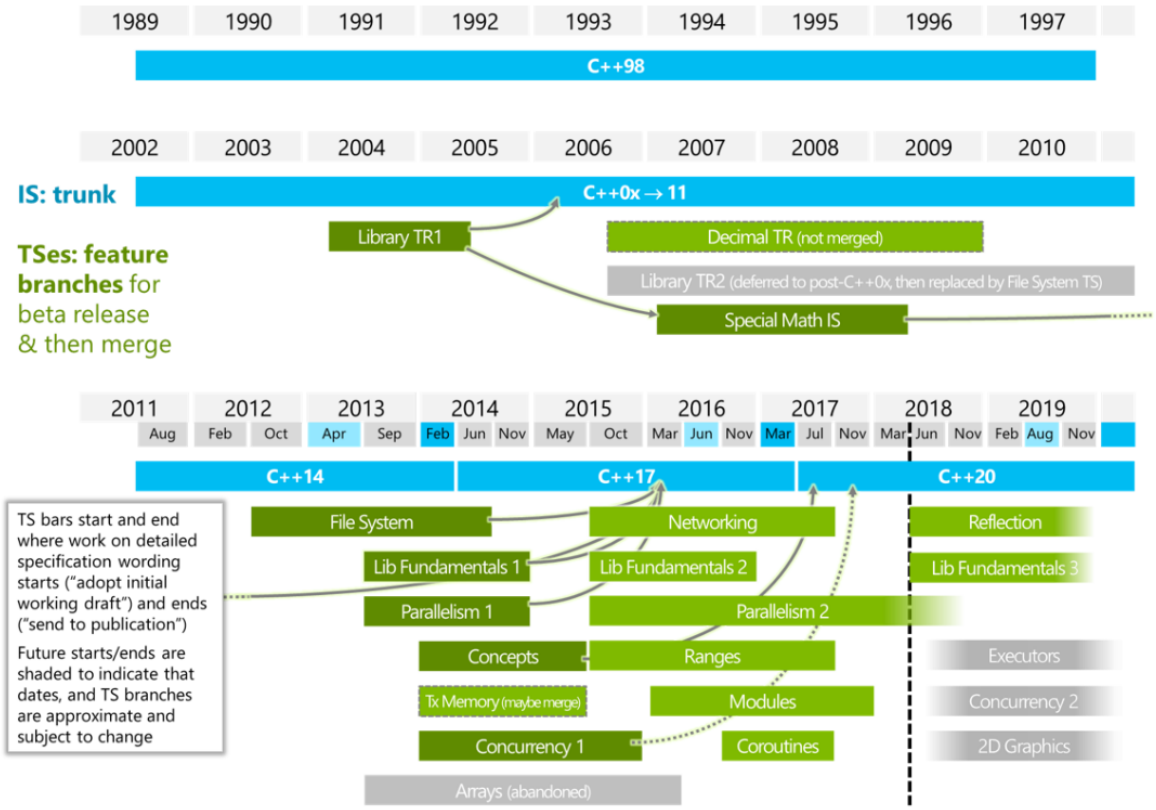
- 1983年8月，第一个C++实现投入使用（所以我喜欢说1983年C++开了天界）
- 1983年12月，Rick Mascitti建议命名为CPlusPlus，即C++（`++`是C语言中的自增操作符）。
- 1985年2月， 第一个C++ Release E发布。
    - 10月，CFront的第一个商业发布，CFront Release 1.0。
    - 10月，Bjarne博士完成了经典巨著The C++ Programming Language第一版
- 1986年11月，C++第一个商业移植CFront 1.1,Glockenspiel。
- 1987年2月， CFront Release 1.2发布。
    - 11月，第一个USENIX C++会议在新墨西哥州举行。
- 1988年10月，第一次USENIX C++实现者工作会议在科罗拉多州举行。
- 1989年12月，ANSI X3J16在华盛顿组织会议。
- 1990年3月， 第一次ANSI X3J16技术会议在新泽西州召开，**ANSI C++**委员会成立.
    - 5月， C++的又一个传世经典ARM诞生。
    - 7月， **模板**被加入。
    - 11月，异常被加入。
- 1991年6月， The C++ Programming Language第二版完成。
    - 6月， 第一次ISO WG21会议在瑞典召开，**ISO C++委员会**成立。
    - 10月，CFront Release 3.0发布。
- 1992年，STL的第一个实现版本**HP-STL**以C++实现
- 1993年3月，运行时类型识别在俄勒冈州被加入。
    - 7月， **名字空间**在德国慕尼黑被加入。
- 1994年2月，**标准模板库**正式成为ANSI/ISO C++的一部分
    - 8月， ANSI/ISO委员会草案登记。
- 1997年7月， The C++ Programming Language第三版完成。
    - 10月，ISO标准通过表决被接受
- 1998年11月，ISO标准被批准。 (第一个C++标准)

3、C++标准发展

由ISO/IEC JTC1/SC22/WG21进行。已经出版的标准文档如下：

发布时间	文档	通称	备注
2017	**ISO/IEC 14882:2017**	**[C++17]**	**第五个C++标准**
2017	ISO/IEC TS 22277:2017	coroutines TS	协程库扩展
2017	ISO/IEC TS 21425:2017	ranges TS	提供范围机制
2017	ISO/IEC TS 19568:2017	library fundamentals TS	标准库扩展
2016	ISO/IEC TS 19571:2016	concurrency TS	用于并发计算的扩展
2015	ISO/IEC TS 19217:2015	concepts TS	概念库，用于优化编译期信息
2015	ISO/IEC TS 19841:2015	TM TS	事务性内存操作
2015	ISO/IEC TS 19570:2015	parallelism TS	用于并行计算的扩展
2015	ISO/IEC TS 18822:2015	filesystem TS	文件系统
2014	**ISO/IEC 14882:2014**	**C++14**	**第四个C++标准**
2011	ISO/IEC TR 24733:2011	-	十进制浮点数扩展
2011	**ISO/IEC 14882:2011**	**C++11**	**第三个C++标准**
2010	ISO/IEC TR 29124:2010	-	数学函数扩展
2007	ISO/IEC TR 19768:2007	C++TR1	C++技术报告：库扩展
2006	ISO/IEC TR 18015:2006	-	C++性能技术报告
2003	**ISO/IEC 14882:2003	**C++03**	**第二个C++标准**
1998	**ISO/IEC 14882:1998	**C++98**	**第一个C++标准**

4、C++标准未来规划



5、C++11标准的影响

自C++诞生后，在面向对象语言迅速发展的时代背景下，C++以其面向对象的语言特性、对C语言的良好兼容性，以及极其接近C语言的性能效率，在工业界占据了相当大的份额，成为了程序设计语言中的无冕之王。在其后的发展中，C++又不断引入新的内容，标准模板库和Boost程序库的出现、泛型程序设计的流行，使得C++牢牢占据了TIOBE编程语言排行榜前三名的位置，成为业界最流行的程序设计语言之一，成为一个众人传颂的传奇。

然而，随着硬件技术的不断发展，特别是多核技术的出现以及Java、C#等新语言的不断涌现，C++的发展受到了很大的冲击，在业界的应用范围不断萎缩。C++曾经是Visual Studio6.0中的首选语言，但在后续版本中，特别是在微软推出了.NET Framework之后，C++的地位不断下滑，被后来居上的C#抢了风头。很多钟情于C++的程序员不禁发出这样的感叹："C++老矣，尚能编否？"

虽然C++在发展历程中经历了上述小小的波折，但是应该看到，世界上还是有无数的C++代码在稳定地运行着，这些代码还需要维护和升级。另外，C++在某些领域（比如操作系统编程、游戏开发、服务器端开发等）仍然有不可替代的优势。为了应对现代程序设计语言的发展及业界的需求，C++有积极汲取现代程序设计语言的精华，C++的新标准C++11正是在这样的背景下应运而生的。

C++11是自1998年C++首次被ISO标准化以来变化最大的一个新标准，它主要在以下两个方面对C++进行了革命性的改进和增强：

一方面，C++11让C++更加易于使用。我们都知道，C++以其语法简洁而著称于世，受到编程高手们的喜爱。同时，C++也非常灵活而自由，我们可以几乎在C++中完成任何我们想要完成的事情。简洁、自由和灵活是一把双刃剑，它让C++拥有无限的能力，但同时也让C++在程序员的心中成为一门难以掌握的编程语言，特别是让一些初学者望而却步，阻碍了C++的进一步发展。为了改变这一现状，C++11从其他主流的编程语言（Java）中借鉴吸收了很多旨在改善其易用性的语法特性。例如，

- C++11提供了auto这种特殊的数据类型，使用它作为变量的数据类型，编译器可以根据变量的初始值自动推断其合理的数据类型，省去了程序员确定复杂变量的数据类型的繁琐；
- C++11开始支持lambda表达式，让C++中匿名函数的定义和使用成为可能；
- C++11从Java中借鉴了序列for循环语句，让针对某个容器的循环遍历更加简单；

- C++11从Java中借鉴了函数属性，从而可以对函数进行更加灵活的修饰。例如，可以使用noreturn指明一个函数没有返回值，也可以使用final限制某个虚函数被派生类重载，函数属性的引入满足了对函数的不同需求。

一方面，C++11让C++性能更高。相对于其他主流的高级编程语言而言，接近于低级语言的高性能表现，应该是C++最大的优势了。例如，

- C++11提供了对右值引用、移动语义的完全支持，解决了从函数返回一个大对象的问题；
- C++11利用新的语法特性对标准库进行了大规模的改写，极大地提高了标准库的性能表现；
- 为了适应当今越来越普及的并行计算，充分利用主流的多核CPU的计算资源，C++11在标准库中对并行计算提供了全面的支持，我们可以通过线程thread对象轻松完成线程的创建，也可以通过条件变量对线程的执行情况进行控制。

正是C++11在这两个方面的大力改进，不仅进一步增强了C++在性能方面的优势，做到了扬长；同时改善了C++的易用性，做到了避短，使得C++成为了一门“又快又好”的程序设计语言。这些新特性给C++注入了新的活力，使得C++重新焕发青春，带来C++的复兴。

Why C++王者归来 -> https://coolshell.cn/articles/6548.html

6、总结

C++发明至今已经三十多年了，从最早的简单面向对象逐渐发展成为包含泛型、函数式、模板元等许多范式的复杂混合体，其中的每一个编程范式都可以自成体系，在开发过程中打出一片天地。

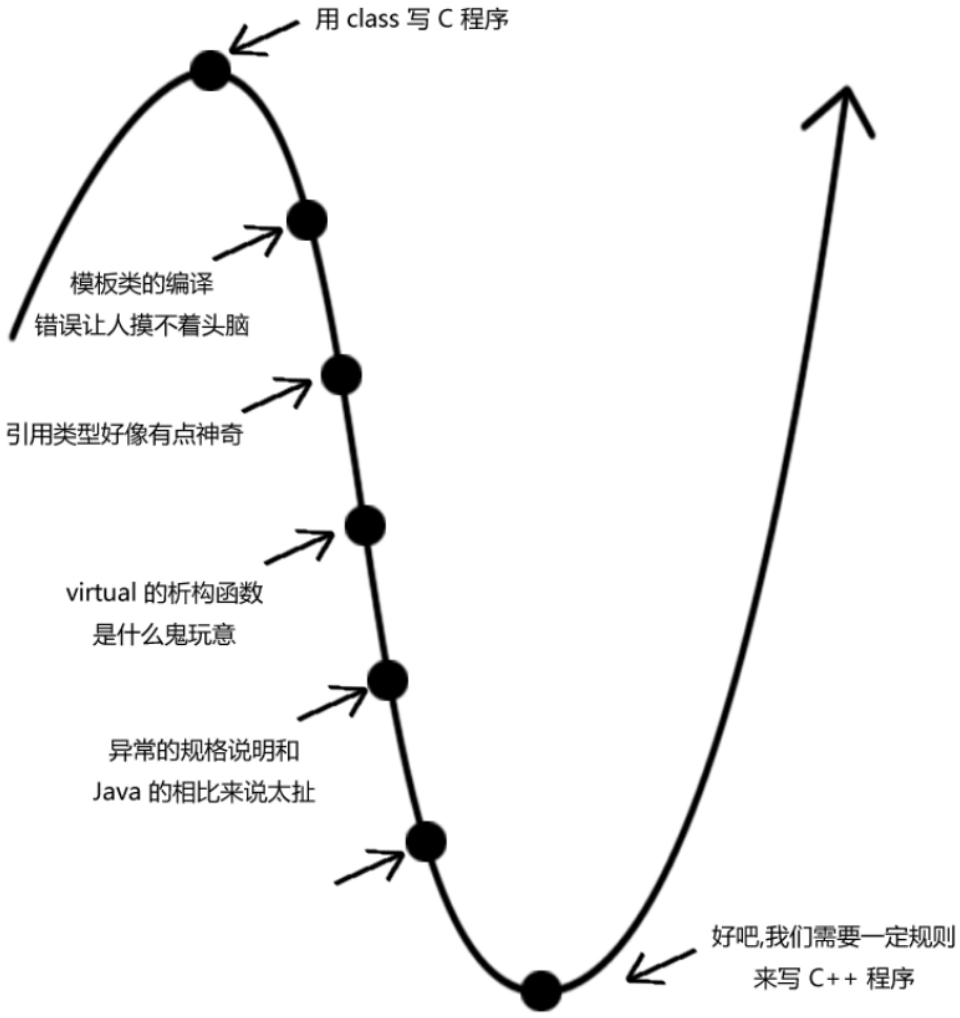
二十年前，面向过程、基于对象是C++编程的主流范式；十年前，主流范式变成了面向对象+设计模式；而现在C++编程的主流范式则有“返璞归真”的趋势，过度使用虚函数的庞大类继承体系逐渐被摒弃，而使用泛型、函数式等新型范式开发精致的小类并给予良好的组合成为了大方向。

STL和Boost充分实践了现代C++编程方法，不使用复杂的继承体系，特别是Boost，它使用泛型编程、模板元编程和编译期的静态多态构建了功能完善的组件，代码规范精炼，是值得学习的极好范例。

7、21天成为C++大神?

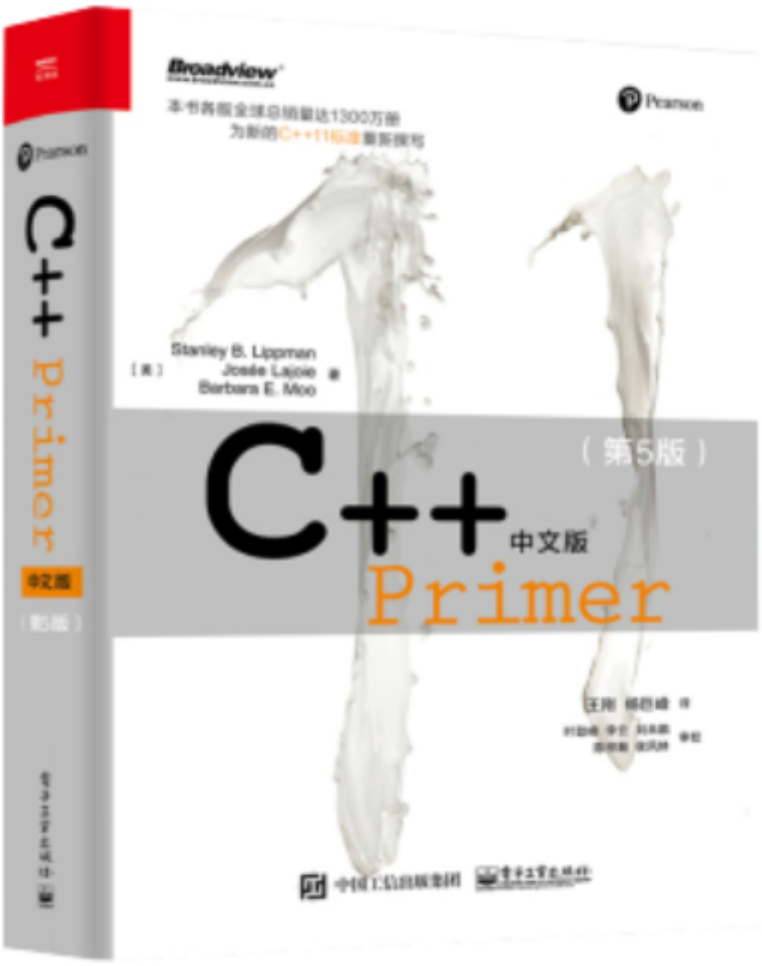

第1-10天
学习变量，常量，数组，字符串，表达式，语句，函数……
第11-21天
学习程序流程，指针，引用，类，对象，接口，多态……
第22-697天（两年）
进行大量的“休闲娱乐”方式的编程，并在Hack代码中找到乐趣，从错误中学习总结。
第698-3648天（八年）
与其它程序员互相影响，并一同开发项目，并向他们学习。
第3649-7781天（十年）
学习高等理论物理，使用公式证明量子物理理论。
第7782-14611天（二十年）
学习生物化学，分子生物学，以及遗传学。
第14611天
（自学习编程来的四十年）
使用在生物学方面的知识造一个“返老还童”药剂。
第14611天
使用在物理学上的知识造一个通量电容器，把自己传送回学编程的第21天。
第21天
把那时的自己给替换了。

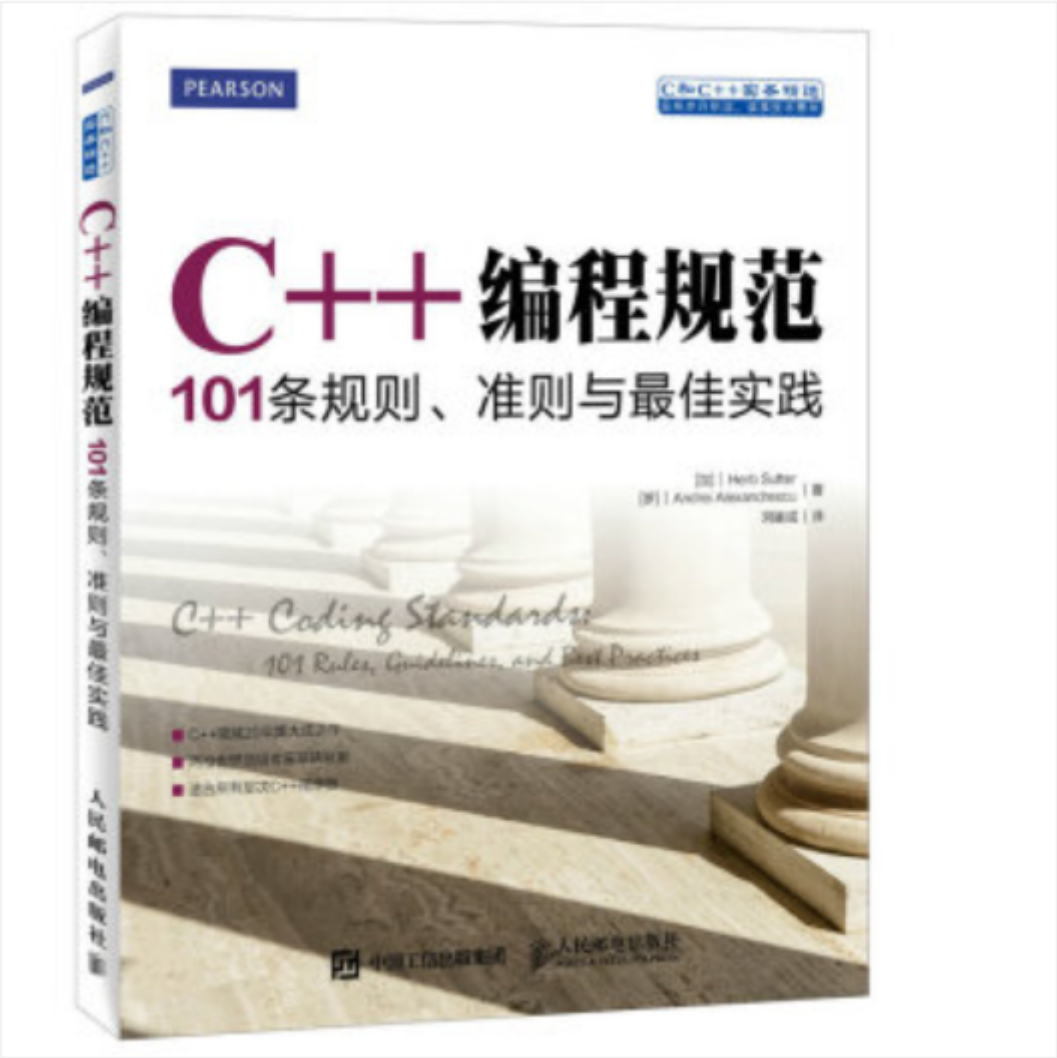
8、C++程序员自信心曲线



9、C++学习推荐书籍

初学入门:

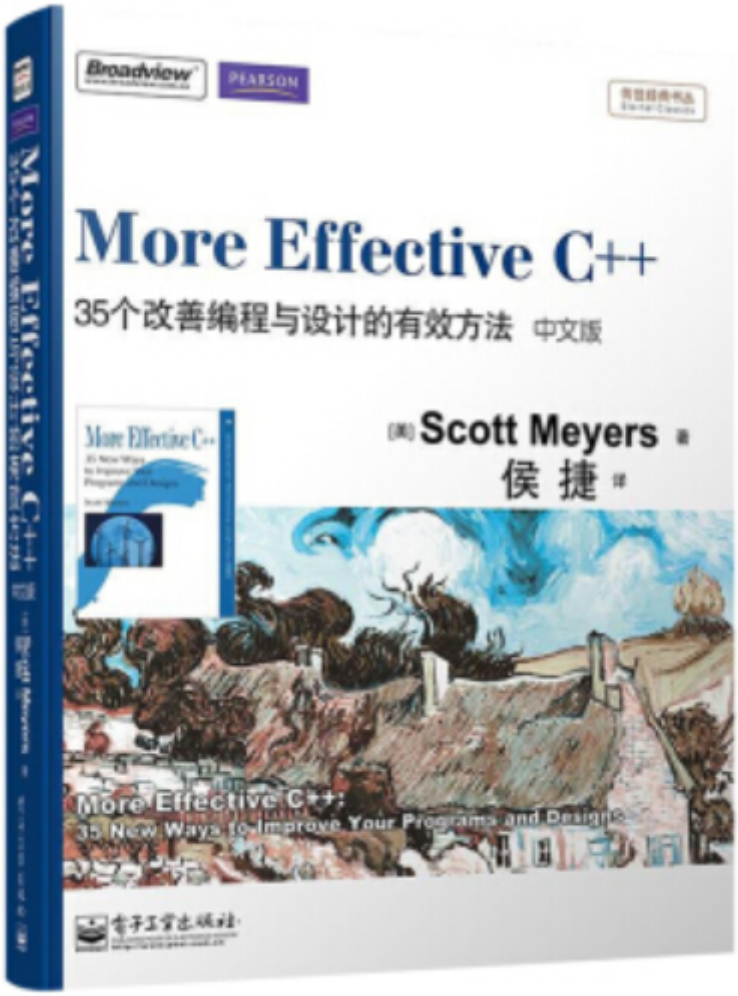

Broadview
本书再版全球总销量达1300万册
为新的C++11标准重新撰写
Pearson
Stanley B. Lippman
[美] Josée Lajoie
Barbara E. Moo 著
C++ 中文版 Primer
（第5版）
王刚 杨巨峰 译
Pearson
C++ Primer
中文版
（第5版）

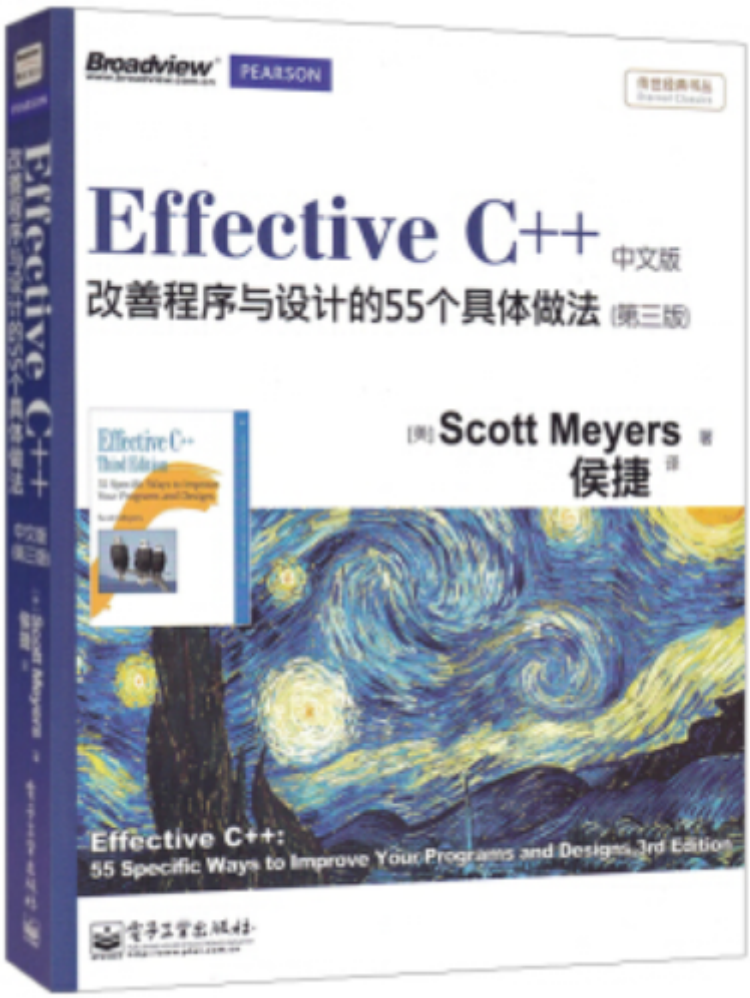


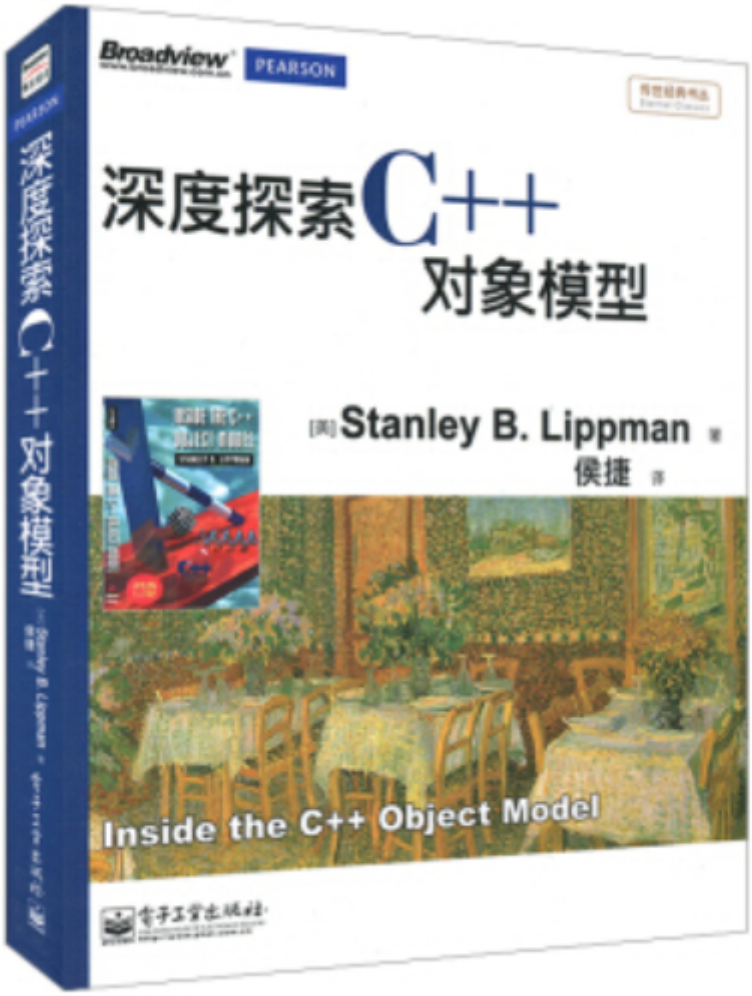
Google的C++编程规范

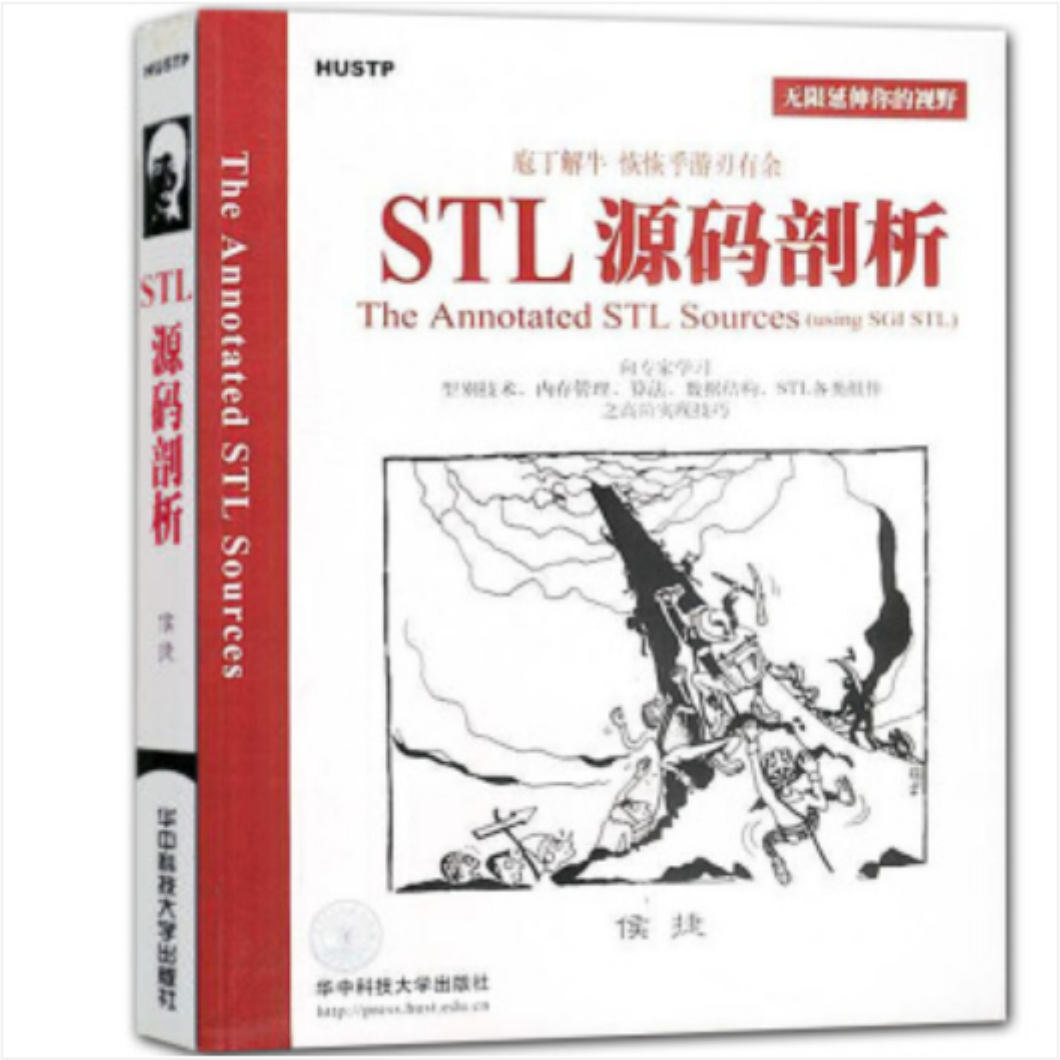
https://zh-google-styleguide.readthedocs.io/en/latest/google-cpp-styleguide/

进阶：


Broadview
PEARSON
More Effective C++
35个改善编程与设计的有效方法 中文版
[美] Scott Meyers 著
侯捷 译
More Effective C++:
35 New Ways to Improve Your Programs and Designs


Broadview
PEARSON
Effective C++ 中文版
改善程序与设计的55个具体做法 (第三版)
[美] Scott Meyers 著
侯捷 译
Effective C++:
55 Specific Ways to Improve Your Programs and Designs, 3rd Edition


Broadview
PEARSON
深度探索C++对象模型
[美]Stanley B. Lippman 著
侯捷 译
Inside the C++ Object Model
电子工业出版社


HUSTP
STL 源码剖析
侯捷
The Annotated STL Sources
HUSTP
无限延伸你的视野
庖丁解牛 恢恢乎游刃有余
STL 源码剖析
The Annotated STL Sources (using SGI STL)
向专家学习
型别技术、内存管理、算法、数据结构、STL各组件
之高阶实现技巧
侯 捷
华中科技大学出版社