C++ 语言超详细系统学习路线（2024年最新）

送给学习 C++ 的同学一段翁凯老师讲课说的话：

学 C++ (计算机)一定要有一个非常强大的心理状态

什么呢

C++ （计算机）的所有东西都是人做出来的，别人能想出来的我也一定想得出来，在 C++（计算机）里头没有任何黑魔法，所有的东西只不过是我现在不知道而已

总有一天我会把它里面的细节搞明白的!

之前在这篇文章末尾C++ 牛逼!

(opens new window)挖了一个坑，就是写一个系统学习 C++ 路线。

C++ 真的是一门备受争议的语言，关于 C++ 的书籍也是浩瀚如海。

那几本 C++ 的四书五经想必大家也早已在各路学习路线中见过了。

不过没关系，今天你依然会看见它们的身影哈哈哈。

C++ 最大的缺点就是语法复杂，而且难学难精。

学习 C 语言也许一本 K&R 的《C程序设计语言》就够了，但是在 C++ 世界里，找不到这样一本书。

这是一个爱速成的年代，大多数人很难耐得住寂寞花很长时间去学习一门语言，所以《21天学通C++》的销量还蛮不错的。

但是很遗憾，别说 21 天，就是一两年也很难学通 C++ 。

学好过程，一定离不开读大量的书籍甚至标准库文档，我给大家整理了一下 C++ 必看经典书籍

微信搜索「编程指北」关注后在后台「cpp」即可获取本文提到的书籍电子版PDF以及C++书单大全：

C++书单

👆也可以截图微信扫描上方二维码， 回复「cpp 」即可获取

有需要自取: C++必读书籍(含下载方式)

在这里，我也把 C++ 学习初略的分为几个层次：

#

一、入门

#

1.1 简单入门教程

如果你之前没有 C 或者其它语言的基础，我建议看看网上的 C++ 入门教程，比如菜鸟教程、C 语言中文网上面。

#

1.1.1 需要掌握的基本概念

这个过程你需要了解并且掌握所有编程语言中共同的、最基本的那些概念：

变量、常量、类型、循环、判断、数组、字符串、对象、输入输出等

然后还有 C++ 中一些特有的东西，比如命名空间 namespace、引用、模板等。

以及如何使用一些 STL 中的类库，比如 string、vector、find 之类的东西。

然后就是辅以大量的练习。

#

1.1.2 练习网站

就那种 C++ 教程的课后练习题就足够了，类似水仙花数、四则运算计算器之类的东西。

如果对这些没什么兴趣，可以试试这个网站，

https://www.codewars.com/kata/search/cpp

(opens new window)

有点像 LeetCode 刷算法题一样，这里针对每一个语法知识点都有一系列的练习题可供选择。

而且不仅包含 C++，其它几乎所有语言都可以在这里进行练习：

应该两三周就能过完菜鸟教程这种级别的指引吧。

到这里，你对 C++ 包含了哪些知识点大概有了一个整体的印象。

并且能够写上百行左右的程序，但是还是很难写出一个优秀的类。

#

二、系统全面学习

接下来需要进行全面和深入的学习，算是真正的入门：

#

2.1 系统入门书籍

《Accelerated C++》（美国斯坦福大学的经典教材）

《C++ Primer》（经常被推荐的一本书，大而全，字典书）

《The C++ Programming Language》（C++之父 Bjarne Stroustrup 所著）

这三本，其实各有优缺点，第一本优点是简短，仅仅两三百页，只有最为核心和主干的知识点。

而后两本则都是大而全，尤其是《The C++ Programming Language》。

这两本区别在于，一个是 C++ 大师所著，一本是 C++ 之父所著。

网上有人说 《C++Primer》是目前市面上唯一一本真正的从入门到精通的书，适合初学者，当你对C++比较熟悉之后，就可以把这本书当做字典，遇到不太清晰的点再回过头去看 《C++ Programming language》 是C++专家自学指南，顾名思义，适合有较深厚 C++ 功底的读者。

#

2.2 推荐阅读顺序

所以小北推荐的顺序是： 《Accelerated C++》->《C++ Primer》->《The C++ Programming Language》

对于这种上前页大部头我推荐的阅读方式是，以主题为划分，比如 C++ Primer 就明确的分为了：

C++ 基础

C++ 标准库

类设计者的工具

高级主题 并且越到后面，你越可以直接翻书的目录，跳跃着找你感兴趣或者说还不太清楚的知识点看，不用再像看第一本书一样从头翻到尾。

#

2.3 入门必须掌握的知识

如果有一定的基础的话，可以看着目录，看一些自己感兴趣的章节，完全没有必要从第一页开始挨着挨着读。

入门结束你应该掌握以下内容：

基础语言

类与面向对象

输入输出

字符串处理（类库和正则表达式）

指针&引用

容器类库

泛型算法

看着只有几个关键字，实际上每个展开都有很多内容需要学习。

学习过程中把后面的每一个练习题都自己敲一遍，自己多思考对比一下。

多用代码去验证自己的想法，尤其是指针、引用、构造、析构这些地方。

#

2.4 C++ 学习必备网站

学习 C++ 以及日常开发一定要记得这几个网站，可以随时查阅一些语法的用法和标准库：

https://en.cppreference.com/w/

(opens new window)

https://cplusplus.com

(opens new window)

https://isocpp.org

(opens new window)

#

三、提高

#

3.1 提高 C++ 代码的效率 & 规范

看完 C++ Primer 这类书，写出来的代码在效率、规范上还是没那么 “C++”。 甚至会有内存方面的 Bug。 比如返回函数内局部变量的引用，经常 pass-by-value 而非 pass-by-reference-to-const。

#

3.2 系统编程基础

由于 C++ 是一门系统级编程的语言，你需要掌握一些 OS 方面的知识才能更好的使用。

尤其是内存编译链接这一块，比如堆、栈、静态链接、动态链接等等。

推荐这两本书：

#

3.2.1 《深入理解计算机系统》

对于C/C++程序员，强烈推荐阅读《CSAPP》和《程序员自我修养》 当然时间不够可以不全看，对于 CSAPP 来说即便你的时间不够，这几个章节也是要坚持看一部分的：

第一章：计算机系统漫游：对于整个计算机系统概述，非常的精华

第三章：程序的机器级表示：就是C语言和汇编语言，当然对于加强对C++的理解也是非常有帮助

第七章：链接，你会明白什么是静态链接、动态链接、加载等等，这对于编码有实际意义的，尤其是你在大型项目中会遇到各种链接阶段的错误、符号未定义、符号冲突之类的，看完这章基本都能掌握。

第九章：虚拟内存，这块是将内存的，C/C++程序员必须精确的掌握自己程序的内存分配和释放。

#

3.2.2 《程序员自我修养：装载、链接》

时间多的话可以进一步看《程序员的自我修养-链接、装载、库》，这本书整体来说约等于把 CSAPP 的第七章彻底、完全的展开讲解了，所以充斥着大量的细节。

还有一些 C++ 里比较特殊的概念，比如 RAII 的资源管理方式、值语义与生命期、引用、右值引用等等。

这些东西似乎没有一本很系统的书籍可以帮助你掌握，只能在不同的书里吸收完善。

这一块，我也在路上。

书籍推荐：：

这里给出一些在这些方面有帮助的书籍：

#

3.3 《Effective C++》

一言概括，就是帮助你快速习得一些 C++ 里正确的姿势。

否则你可能需要用几万行代码的经验才能慢慢悟出来。

#

3.4 《Effective Modern C++》

上面 Effective C++ 不太涉及 C++11 的一些新语法，而这本 Modern C++，就是关于现代C++(++C11之后)的指南，包C++11之后的一些最佳实践和技巧，

《Effective Modern C++》是一本关于现代 C++ 的指南，它包含了一些最佳实践和技巧，帮我们利用现代 C++ 的特性来编写高效、安全、可维护的代码。

#

3.5《Linux多线程服务端编程》

这本书严格来说不是纯讲 C++ 的。

但是里面会有一些章节，关于 C++ 多线程编程、锁、线程安全、C++ 一些工程实践经验还不错。

#

3.6《Google C++ Style Guide》

谷歌的 C++ 代码规范，写代码是一件专业的事，把代码写得漂亮、专业也是一项能力。

谷歌的代码规范算是 C++ 领域一个标杆了。

鹅厂很多代码规范也是基于谷歌的代码规范做了一些适应性的改进。

在学习编程之初就掌握良好的编码规范大有裨益。

#

3.7《程序员自我修养》&《CSAPP》

上面已经介绍过了。

帮助你掌握一些系统级编程的知识。

如内存管理、堆栈、链接这类在 C、C++ 中极其重要的概念。

#

四、进阶

上面提到的这些书重点在于如何使用、用好 C++ 这一个范畴。

而进阶，我认为则是研究语言一些 case 的具体实现，深刻的思考语言设计和演进本身。 这里分为三个部分：

#

4.1 STL 源码

侯捷老师说：源码之下，了无秘密。

为了更好的使用 STL，一个方法就是阅读 STL 源码 或者掌握部分实现：

#

4.1.1《STL源码剖析》

这本书讲解了 STL 五大组件的底层实现。

包括内存管理、各类容器的数据结构实现、算法的实现等。

建议列为必读，可以帮助深入理解 STL 底层。

其中也有很多面试常考的知识点，如内存池、traits 技法、vector 动态扩容、set、map 等实现原理等。

#

4.1.2 《Effective STL》

作者Scott Meyers，这本书主要关注STL的最佳实践，包括如何选择合适的容器和算法、如何使用迭代器、如何避免常见的STL陷阱等等。这本书适合有一定STL基础的程序员。

#

4.2 C++ 对象模型

#

4.2.1 《深度探索C++对象模型》

这本书其实就是围绕一个点：C++ 里的对象是在内存中是如何组织的。

最重要的是作者并非是直接给出结论，而是带着读者一起思考，如果作为语言设计者，该如何去设计与实现，是一步步引导出结论的。

掌握了C++的对象组织，其实其它语言的对象也就是大同小异了，如 Java、Python，你能很快的 Get 到面向对象其实都是差不多的实现方式。

这是每个章节内容：

关于对象

构造函数语意学

Data语意学

Function语意学

构造、解构、拷贝语意学

执行期语意学

我第一次看见这本书就被深深的吸引了，当时在学校，每天看八九个小时，花了大概一周刷完了。

看完这本书感觉自己对于 C++ 的底层实现机制有了更深的理解。

比如多重继承、菱形继承、虚函数底层的实现机制等等。

当然了，也有人会认为对象如何在底层实现属于 C++ 编译器实现细节，而非语言标准，不应该以实现细节倒推标准。

诚然如此，掌握实现细节还是会对你理解 C++ 有促进作用。

#

4.3 语言设计与演化

这里重点是思考 C++ 语言很多语法的设计和演进方向，可以看看 C++ 之父 Bjarne Stroustrup's FAQ、《C++设计与演化》。

设计与演化这本书重点在于通过 C++ 之父去窥见 C++ 的前世今生。

看完这本书的意义，功利一点来说，或许对于提高 C++ 编程能力没有太多帮助的，不像 Effective 是立竿见影的效果。

但是，对于加深对 C++ 的理解很有帮助。

想要更好的使用 C++ 构建好的、高效的系统，对语言本身的深入理解是先决条件。

另一方面，书中提到了很多语言设计的思想、哲学，这些，相信在学习其他编程语言的时候也是有帮助的。

#

五、关于 C++ 书单

这篇文章中对于书籍，我的一个原则就是能省则省，因为 C++ 相关的优秀书籍实在太多了，我只想挑出每个阶段最有用的的书。

书不在多，贵在精。

看书也是一样的，我推荐书不一定都需要去看完，深入细致的阅读完一本技术书比走马观花似的阅读 N 本更有效。

当然了，后面我也会推出一份分门别类整理后的 C++ 书单供大家索引。

#

六、关于视频

对于初学者来说看书也许确实很难 get 到那个点，就像我大一看 C 语言的教材里说文件分为文本流文件和二进制流文件，当时我愣是理解不了。

很多对于会的人是常识的东西，对于初学者就是无法理解、很难的概念。

那也许通过别人讲授、视频的方式会更容易入门一点，这里推荐浙大翁凯的 C++ 视频。

翁凯老师无论是 C、C++、Java 的视频都讲得非常深入浅出，只能说水平越是高的人讲的课越容易明白。

这是网易云课堂地址： https://study.163.com/course/introduction/271005.htm

(opens new window)

另外还要推荐侯捷老师的一些列课程：

我当时看见这些标题：《内存管理》、《C++11》、《STL源码分析》......

简直如获至宝一样，一个寒假就把这些视频全看完了，对于 C++ 和内存管理方面理解提高了很多。

真的非常感谢 B 站、感谢这些讲师和分享者，甚至要感谢互联网时代！ 他们让知识的传播变得更加的便捷，降低了后来者的学习门槛，当然啦也变相加重了内卷化hahah（逃

不过最近B站上侯捷老师的视频好像都不在了，我网盘里有一份，分享给大家学习。

侯捷老师的视频课程可以在这获取学习: 侯捷C++视频全系列下载

(opens new window)

最近还看了一个国外 C++ 游戏引擎开发工程师录制的一套偏向 talk 类型的教程，比较通俗易懂和深入浅出， 推荐一下：

https://www.bilibili.com/video/BV1Ay4y1i7Z6

(opens new window)

#

END

最后有两个网站希望大家学习 C++ 过程中多去逛逛：

C++ Reference. https://en.cppreference.com

(opens new window)

Core Guidelines. https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines

(opens new window)