

## 程序员练级攻略（2018）：编程语言

2018-06-12 陈皓



程序员练级攻略（2018）：编程语言

朗读人：柴巍 17'09" | 7.86M

为了进入专业的编程领域，我们需要认真学习以下三方面的知识。

编程语言。你需要学习 C、C++ 和 Java 这三个工业级的编程语言。为什么说它们是工业级的呢？主要是，C 和 C++ 语言规范都由 ISO 标准化过，而且都有工业界厂商组成的标准化委员会来制定工业标准。次要原因是，它们已经在业界应用于许多重要的生产环境中。

- C 语言不用多说，现今这个世界上几乎所有重要的软件都跟 C 有直接和间接的关系，操作系统、网络、硬件驱动等等。说得霸气一点儿，这个世界就是在 C 语言之上运行的。
- 而对于 C++ 来说，现在主流的浏览器、数据库、Microsoft Office、主流的图形界面、著名的游戏引擎等都是用 C++ 编写的。而且，很多公司都用 C++ 开发核心架构，如 Google、腾讯、百度、阿里云等。
- 而金融电商公司则广泛地使用 Java 语言，因为 Java 的好处太多了，代码稳定性超过 C 和 C++，生产力远超 C 和 C++。有 JVM 在，可以轻松地跨平台，做代码优化，做 AOP 和

IoC 这样的高级技术。以 Spring 为首的由庞大的社区开发的高质量的各種輪子讓你只需關注業務，是能够快速搭建企业级应用的不二之选。

此外，我推荐学习 Go 语言。一方面，Go 语言现在很受关注，它是取代 C 和 C++ 的另一门有潜力的语言。C 语言太原始了，C++ 太复杂了，Java 太高级了，所以 Go 语言就在这个夹缝中出现了。这门语言已经 10 多年了，其已成为云计算领域事实上的标准语言，尤其是在 Docker/Kubernetes 等项目中。Go 语言社区正在不断地从 Java 社区移植各种 Java 的轮子过来，Go 社区现在也很不错。

如果你要写一些 PaaS 层的应用，Go 语言会比 C 和 C++ 更好，目前和 Java 有一拼。而且，Go 语言在国内外一些知名公司中有了一定的应用和实践，所以，是可以学习的（参看：《[Go 语言、Docker 和新技术](#)》一文）。此外，Go 语言语法特别简单，你有了 C 和 C++ 的基础，学习 Go 的学习成本基本为零。

理论学科。你需要学习像算法、数据结构、网络模型、计算机原理等计算机专业需要学习的知识。为什么要学好这些理论上的知识呢？

- 其一，这些理论知识可以说是计算机科学这门学科最精华的知识了。说得大一点，这些是人类智慧的精华。你只要想成为高手，这些东西是你必需要掌握和学习的。
- 其二，当你在解决一些很复杂或是很难的问题时，这些基础理论知识可以帮到你很多。我过去这 20 年从这些基础理论知识中受益匪浅。
- 其三，这些理论知识的思维方式可以让你有触类旁通，一通百通的感觉。虽然知识比较难啃，但啃过以后，你将获益终生。

另外，你千万不要觉得在你的日常工作或是生活当中根本用不上，学了也白学，这样的思维方式千万不要有，因为这是平庸的思维方式。如果你想等我用到了再学也不晚，那么你有必要看一下这篇文章《[程序员的荒谬之言还是至理名言？](#)》。

系统知识。系统知识是理论知识的工程实践，这里面有很多很多的细节。比如像 Unix/Linux、TCP/IP、C10K 挑战等这样专业的系统知识。这些知识是你能不能把理论应用到实际项目当中，能不能搞定实际问题的重要知识。

当你在编程的时候，如何和系统进行交互或是获取操作系统的资源，如何进行通讯，当系统出了性能问题，当系统出了故障等，你有大量需要落地的事需要处理和解决。这个时候，这些系统知识就会变得尤为关键和重要了。

这些东西，你可以认为是计算机世界的物理世界，上层无论怎么玩，无论是 Java NIO，还是 Nginx，还是 Node.js，它们都逃脱不掉最下层的限制。所以，你要好好学习这方面的知识。

## 编程语言

## Java 语言

学习 Java 语言有以下入门级的书（注意：下面一些书在入门篇中有所提及，但为了完整性，还是要在这一提一下，因为可能有朋友是跳着看的）。

- 《[Java 核心技术：卷 1 基础知识](#)》，这本书本来是 Sun 公司的官方用书，是一本 Java 的入门参考书。对于 Java 初学者来说，是一本非常不错的值得时常翻阅的技术手册。书中有较多地方进行 Java 与 C++ 的比较，因为当时 Java 面世的时候，又被叫作 "C++ Killer"。而我在看这本书的时候，发现书中有很多 C++ 的东西，于是又去学习了 C++。学习 C++ 的时候，发现有很多 C 的东西不懂，又顺着去学习了 C。然后，C -> C++ -> Java 整条线融汇贯通，这对我未来的技术成长有非常大的帮助。
- 有了上述的入门后，Java 的 Spring 框架是你玩 Java 所无法回避的东西，所以接下来是两本 Spring 相关的书，《[Spring 实战](#)》和《[Spring Boot 实战](#)》。前者是传统的 Spring，后者是新式的微服务的 Spring。如果你只想看一本的话，那么就看看后者吧。

认真学习前面的书可以让你成功入门 Java，但想要进一步成长，就要看下面我推荐的几本提升级的书。

- 接下来，你需要了解了一下如何编写高效的代码，于是必需看一下《[Effective Java](#)》（注意，这里我给的引用是第三版的，也是 2017 年末出版的书），这本书是模仿 Scott Meyers 的经典图书《Effective C++》的。Effective 这种书基本上都是各种经验之谈，所以，这是一本非常不错的书，你一定要读。
- 《[Java 并发编程实战](#)》，是一本完美的 Java 并发参考手册。书中从并发性和线程安全性的基本概念出发，介绍了如何使用类库提供的基本并发构建块，用于避免并发危险、构造线程安全的类及验证线程安全的规则，如何将小的线程安全类组合成更大的线程安全类，如何利用线程来提高并发应用程序的吞吐量，如何识别可并行执行的任务，如何提高单线程子系统的响应性，如何确保并发程序执行预期任务，如何提高并发代码的性能和可伸缩性等内容。最后介绍了一些高级主题，如显式锁、原子变量、非阻塞算法以及如何开发自定义的同步工具类。
- 了解如何编写出并发的程序，你还需要了解一下如何优化 Java 的性能。我推荐《[Java 性能权威指南](#)》。通过学习这本书，你可以比较大程度地提升性能测试的效果。其中包括：使用 JDK 中自带的工具收集 Java 应用的性能数据，理解 JIT 编译器的优缺点，调优 JVM 垃圾收集器以减少对程序的影响，学习管理堆内存和 JVM 原生内存的方法，了解如何最大程度地优化 Java 线程及同步的性能，等等。看完这本书后，如果你还有余力，想了解更多的底层细节，那么，你有必要去读一下《[深入理解 Java 虚拟机](#)》。
- 《[Java 编程思想](#)》，真是一本透着编程思想的书。上面的书让你从微观角度了解 Java，而这本书则可以让你从一个宏观角度了解 Java。这本书和 Java 核心技术的厚度差不多，但这本

书的信息密度比较大。所以，读起来是非常耗大脑的，因为它会让你不断地思考。对于想学好 Java 的程序员来说，这是一本必读的书。

- 《[精通 Spring 4.x](#)》，也是一本很不错的书，就是有点厚，一共有 800 多页，都是干货。我认为其中最不错的是在分析原理，尤其是针对前面提到的 Spring 技术，应用与原理都讲得很透彻，IOC 和 AOP 也分析得很棒，娓娓道来。其对任何一个技术都分析得很细致和全面，不足之处就是内容太多了，所以导致很厚，但这并不影响它是一本不错的工具书。

当然，学 Java 你一定要学面向对象的设计模式，这里就只有一本经典的书《[设计模式](#)》。如果你觉得有点儿难度了，那么可以看一下《[Head First 设计模式](#)》。学习面向对象的设计模式时，你不要迷失在那 23 个设计模式中，你一定要明白这两个原则：

- Program to an 'interface', not an 'implementation'
  - 使用者不需要知道数据类型、结构、算法的细节。
  - 使用者不需要知道实现细节，只需要知道提供的接口。
  - 利于抽象、封装，动态绑定，多态。符合面向对象的特质和理念。
- Favor 'object composition' over 'class inheritance'
  - 继承需要给子类暴露一些父类的设计和实现细节。
  - 父类实现的改变会造成子类也需要改变。
  - 我们以为继承主要是为了代码重用，但实际上在子类中需要重新实现很多父类的方法。
  - 继承更多的应该是为了多态。

至此，如果你把上面的这些知识都融汇贯通的话，那么，你已是一个高级的 Java 程序员了，我保证你已经超过了绝大多数程序员了。基本上来说，你在技术方面是可以进入到一线公司的，而且还不是一般的岗位，至少是高级程序员或是初级架构师的级别了。

## C/C++ 语言

不像我出道那个时候，几乎所有的软件都要用 C 语言来写。现在，可能不会有多少人学习 C 语言了，因为一方面有 Java、Python 这样的高级语言为你屏蔽了很多的底层细节，另一方面也有像 Go 语言这样的新兴语言可以让你更容易地写出来也是高性能的软件。但是，我还是想说，C 语言是你必须学习的语言，因为这个世界上绝大多数编程语言都是 C-like 的语言，也是在不同的方面来解决 C 语言的各种问题。这里，我想放个比较武断话——如果你不学 C 语言，你根本没有资格说你是一个合格的程序员！

- 这里尤其推荐，已故的 C 语言之父 Dennis M. Ritchie 和著名科学家 Brian W. Kernighan 合作的圣经级的教科书《[C 程序设计语言](#)》。注意，这本书是 C 语言原作者写的，其 C 语言的标准不是我们平时常说的 ANSI 标准，而是原作者的标准，又被叫作 K&R C。但是这本书很轻薄，也简洁，不枯燥，是一本你可以拿着躺在床上看还不会看着看着睡着的书。



- 然后，还有一本非常经典的 C 语言的书《[C 语言程序设计现代方法](#)》。有人说，这本书配合之前的 [The C Programming Language](#) 那本书简真是无敌。我想说，这本书更实用，也够厚，完整覆盖了 C99 标准，习题的质量和水准也比较高。更好的是，探讨了现代编译器的实现，以及和 C++ 的兼容，还揭穿了各种古老的 C 语言的神话和信条.....是相当相当干的一本学习 C 语言的书。

对了，千万不要看谭浩强的 C 语言的书。各种误导，我大学时就是用这本书学的 C，后来工作时被坑得不行。

在学习 C 语言的过程中，你一定会感到，C 语言这么底层，而且代码经常性地崩溃，经过一段时间的挣扎，你才开始觉得你从这个烂泥坑里快要爬出来了。但你还需要看看《[C 陷阱与缺陷](#)》这本书，你会发现，这里面的坑不是一般大。

此时，如果你看过我的《编程范式游记》那个系列文章，你可能会发现 C 语言在泛型编程上的各种问题，这个时候我推荐你学习一下 C++ 语言。可能会有很多人觉得我说的 C++ 是个大坑。是的，这是世界目前来说最复杂也是最难的编程语言了。但是，C++ 是目前世界上范式最多的语言了，其做得最好的范式就是“泛型编程”，这在静态语言中，是绝对地划时代的一个事。

所以，你有必要学习一下 C++，看看 C++ 是如何解决 C 语言中的各种问题的。你可以先看看我的这篇文章“[C++ 的坑真的多吗？](#)”，有个基本认识。下面推荐几本 C++ 的书。

- 《[C++ Primer 中文版](#)》，这本书是久负盛名的 C++ 经典教程。书是有点厚，前面 1/3 讲 C 语言，后面讲 C++。C++ 的知识点实在是太多了，而且又有点晦涩。但是你主要就看几个点，一个是面向对象的多态，一个是模板和重载操作符，以及一些 STL 的东西。看看 C++ 是怎么玩泛型和函数式编程的。
- 如果你想继续研究，你需要看另外两本更为经典的书《[Effective C++](#)》和《[More Effective C++](#)》。这两本书不厚，但是我读了 10 多年，每过一段时间再读一下，就会发现有更多的收获。这两本书的内容会随着你经历的丰富而变得丰富，这也是对我影响最大的两本书，其中影响最大的不是书中的那些 C++ 的东西，而是作者的思维方式和不断求真的精神，这真是太赞了。
- 学习 C/C++ 都是需要好好了解一下编译器到底干了什么的。就像 Java 需要了解 JVM 一样，所以，这里还有一本非常非常难啃的书你可以挑战一下《[深度探索 C++ 对象模型](#)》。这本书是非常之经典的，看完后，C++ 对你来说就再也没有什么秘密可言。我以前写过的《[C++ 虚函数表解析](#)》，还有《[C++ 对象内存布局](#)》属于这个范畴。
- 还有 C++ 的作者 Bjarne Stroustrup 写的 [C++ FAQ](#)（[中文版](#)），也是非常值得一读的。

## 学习 Go 语言

C 语言太原始了，C++ 太复杂了，Go 语言是不二之选。有了 C/C++ 的功底，学习 Go 语言非常简单。

首推 [Go by Example](#) 作为你的入门教程。然后，[Go 101](#) 也是一个很不错的在线电子书。如果你想看纸书的话，[The Go Programming Language](#) 一书在豆瓣上有 9.2 分，但是国内没有卖的。（当然，我以前也写过两篇入门的供你参考 "[GO 语言简介（上）- 语法](#)" 和 "[GO 语言简介（下）- 特性](#)"）。

另外，Go 语言官方的 [Effective Go](#) 是必读的，这篇文章告诉你如何更好地使用 Go 语言，以及 Go 语言中的一些原理。

Go 语言最突出之处是并发编程，Unix 老牌黑客罗勃·派克（Rob Pike）在 Google I/O 上的两个分享，可以让你学习到一些并发编程的模式。

- Go Concurrency Patterns（[幻灯片](#)和[演讲视频](#)）。
- Advanced Go Concurrency Patterns（[幻灯片](#)、[演讲视频](#)）。

然后，Go 在 GitHub 的 wiki 上有好多不错的学习资源，你可以从中学习到多。比如：

- [Go 精华文章列表](#)。
- [Go 相关博客列表](#)。
- [Go Talks])(<https://github.com/golang/go/wiki/GoTalks>)。

此外，还有个内容丰富的 Go 资源列表 [Awesome Go](#)，推荐看看。

## 小结

好了，最后我们来总结一些今天分享的内容。在编程语言方面，我推荐学习 C、C++、Java 和 Go 四门语言，并分别阐释了推荐的原因。

- 我认为，C 语言是必须学习的语言，因为这个世界上绝大多数编程语言都是 C-like 的语言，也是在不同的方面来解决 C 语言的各种问题。
- 而 C++ 虽然复杂难学，但它几乎是目前世界上范式最多的语言了，其做得最好的范式就是 "泛型编程"，这在静态语言中，是绝对地划时代的一个事。尤其要看看 C++ 是如何解决 C 语言中的各种问题的。
- Java 是我认为综合能力最强的语言。其实我是先学了 Java，然后又去学了 C++，之后去学了 C 语言的。C -> C++ -> Java 整条线融汇贯通，这对我未来的技术成长有非常大的帮助。
- 在文章最末，我推荐了 Go 语言，并给出了相关的学习资料。

我认为，一个合格的程序员应该掌握几门语言。一方面，这会让你对不同的语言进行比较，让你有更多的思考。另一方面，这也是一种学习能力的培养，会让你对于未来的新技术学习得更快。

下篇文章中，我们将分享每个程序员都需要掌握的理论知识。敬请期待。

下面是《程序员练级攻略（2018）》系列文章的目录（持续更新中）。

- [开篇词](#)
- 入门篇
  - [零基础启蒙](#)
  - [正式入门](#)
- 修养篇
  - [程序员修养](#)
- 专业基础篇
  - [编程语言](#)
  - 理论学科
  - 系统知识
- 软件设计篇
- 高手成长篇



版权归极客邦科技所有，未经许可不得转载

#### 精选留言



LI

3

耗子叔推荐的书都很不错，但是实际中没有遇到也没有办法深刻理解

2018-06-12



emoji

1

几个问题，

1.您一下子提到了四门语言，我们这些后生究竟应该精通一门呢，还是多元呢？如果多元，深度是个问题？

2.总有一些经常变和亘古不变的东西，数据结构，算法，网络，计算机基本原理，这些都是很少改变的，而且也需要花很多精力和时间去学习，您提到的几个语言都是经久不衰的，也需要花精力和时间去学习，而时间的总量是固定的，那么我们如何取舍呢？

3.语言本身只是工具，能不能用学到的语言解决问题，这个很关键，不是吗？能不能这么理解，学另一个语言，是因为本语言在某些特定的问题上遇到瓶颈，只能用别的语言来解决？

2018-06-12



pomysky

0

千万不要觉得在你的日常工作或是生活当中根本用不上，学了也白学，这样的思维方式千万不要有，因为这是平庸的思维方式

2018-06-12



akaQin

0

谭浩强是真的坑。。竟然还被用作了大学教材误人子弟

2018-06-12



Link

0

推荐，艾伦.克莱门茨的计算机组成原理，与计算机存储与外设

2018-06-12



Ficapy

0

The go program language这本书国内是有发行中文版的

2018-06-12



cosmos lee

0

感谢皓叔的推荐。我现在公司用的是php做开发，自己目前在学习c，之后想继续学习c++再到java。但是皓叔前面说java的竞争力最强，那么是不是尽快开始入门java更好呢？

2018-06-12



云学

0

搞了8年的c c++，正在接触java，不同的语言确实可以开阔思维，写出更好的代码

2018-06-12



胖胖的奥利奥

0

刚开始学的PHP，后面再学习C语言之后就会发现，其实很多语言的实现都有这些底层语言的影子



2018-06-12



寻路之人

0

干货满满，可以为那些工作了几年后还在迷茫的码农形成技术栈的基本框架，每期都第一时间看。

2018-06-12



寻路之人

0

干货满满，可以为那些工作了几年后还在迷茫的码农形成技术栈的基本框架，每期都第一时间看。

2018-06-12



13683815260

0

准备入坑go.

其实一直觉得C#很好，不过在企业级应用占比现在好像小了。

2018-06-12



给我二两面

0

Go语言确实很简单，我花了一周时间读了本《go programming language》就可以上手写了。如果你已经会了一门语言，再学习其他编程语言时，要从语言特性角度去学(比如支不支持闭包，如何实现类继承机制，包管理机制是什么，静态作用域还是动态作用域)，就会发觉学的非常快。语法细则看一遍即可。实际写代码时，IDE会给你充分提示，静态语言尤是。写着写着就熟练了。

2018-06-12



Jason2Young

0

感谢耗子叔

2018-06-12



Bernard

0

机器学习，区块链，这些怎么处理呢？

2018-06-12



chaoqi

0

common lisp也是一个多范式的编程语言，我看c++懵逼的时候看的它，有一种醍醐灌顶的感觉

2018-06-12