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

陈皓

00:00 / 17:30

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种轮子让你只需关注业务，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科学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，还是要在这里提一下，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。这里需要推荐一下 Google Guava 库，这个库不但是JDK的升级库，其中有如：集合（collections）、缓存（caching）、原生类型支持（primitives support）、并发库（concurrency libraries）、通用注解（common annotations）、字符串处理（string processing）、I/O 等库，其还是Effective Java这本书中的那些经验的实践代表。

- 《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事的。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。

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
  - 软件设计
- 高手成长篇
  - Linux系统、内存和网络（系统底层知识）
  - 异步I/O模型和Lock-Free编程（系统底层知识）
  - Java底层知识
  - 数据库
  - 分布式架构入门（分布式架构）
  - 分布式架构经典图书和论文（分布式架构）
  - .....

emoji

2018-06-12

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？

作者回复

2018-06-15

1）我把Java放在第一位，就是说Java很重要。C语言要学。C++可以跳过，学了C语言，Go语言很自然就学会了。编程语言不复杂的，多花点时间没坏处。

2）不要取舍，排优先级。这些基础知识都是计算机科学专业大学本科的知识，4年你能拿得出来吗？

3）Java语言让你不用关注底层，而关注业务和架构，C语言让你关注底层原理，Go语言介于C和Java之间，掌握多门语言会让你对他们有比较。他们各有各的适用场景。

如果你想成为一个高手，多学几门语言是必须的！

D瓜哥

2018-06-12

读《Effective Java》时，建议学一学Google Guava库，这两个是出之一人之手。书中的很多思想直接就在Guava库中提现出来了。那种感觉，非常棒。

胖胖的奥利奥

2018-06-12

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

作者回复

2018-06-15

是的。学得多就会越学越快

给我二两面

2018-06-12

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。

D瓜哥

2018-06-12

设计模式方面，我更推荐《大话设计模式》程杰著，清华大学出版社出版。
这本书以对话的方式授业解惑；每个模式也是以故事的方式循序渐进地推进，直至设计模式。
另外，难能可贵地是，它还把过去的关于设计模式的几本经典书籍的重点知识摘抄融入正文中。并以黑体标注。强烈推荐！

LI

2018-06-12

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

云学

2018-06-12

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

zliweijk

2018-06-12

为什么大学老师不是有经验的编程工程师，而是毕业直接任教的，读大学时几乎没有听过老师说现在流行什么技术，什么样的企业用什么样的技术比较多，应届生如何才能更好找到适合的工作

akaQin

2018-06-12

谭浩强是真的坑。。竟然还被用作了大学教材误人子弟

myco 前

2018-06-29

想请教皓哥一个问题：我是计算机专业的同学，工作后写了几年Java；想通过看APUE同时捡起来C和类Unix系统；但是开始敲起书上代码的时候发现系统的头文件远不像jdk代码那样文档清晰，感觉难以入门，不知道如何找系统函数文档，如何了解系统调用底层的实现，有点理不清头绪。为了避免陷入X-Y问题，我再说下我的目的：我目标是想学习C，了解类Unix系统底层的东西。想问下皓哥和同学们又啥好的建议？

作者回复

2018-06-29

挺好的，先学C，再学Uinx。文档谷歌一下就可以找到：C语言的：https://en.cppreference.com/w/c 及 https://www.gnu.org/software/libc/manual/
，Linux的：http://man7.org/linux/man-pages/man2/syscalls.2.html

kuna

2018-06-19

您好，我想问问您对 rust 语言怎么看，相对于 C++ 来说是否有某些场景可以替代？Stackoverflow 的调查显示 rust 连续几年排名第一 most loved dreaded and wanted language，但从社区反映来看口碑不佳，您认为是否有深入了解必要，谢谢！

作者回复

2018-06-19

Rust我还是喜欢的，不过我也是在观望中。因为我这篇文章内容太多了，所以只能选择主流的，这样才能确保不让大家走弯路，所以我没提。对于你的这个问题，我个人建议，就个人兴趣是可以深入了解的，但是就职业生涯来说，我则是持保守立场的。

dingtingli

2018-06-14

代码中使用的依赖注入是属于开发的哪个板块？设计模式？
有没有大神解释一下，谢谢。

作者回复

2018-06-15

设计模式IoC/DIP

狮子王V

2018-06-14

耗子老师您好，我最近想研究下一些开源的负载均衡产品，能推荐一下资料吗？非常感谢

作者回复

2018-06-14

底层的lvs，上层的nginx和haproxy

mingshun

2018-06-12

大部分都看过，也都忘了，除工作中常用的 Java 和 Go。一步步走来，感觉 Erlang/Elixir 的玩法才最接近现实世界事物的运作模式。

Groot

2018-06-12

耗叔可以分享一个印象最深刻的谭浩强C语言的的坑吗

李沛霖-程序猿

2018-06-12

C#呢？它的标准和发展现在都要好过java，

修炼

2018-06-12

明显golang

Ficapy

2018-06-12

The go program language这本书国内是有发行中文版的

cosmos lee

2018-06-12

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？

作者回复

2018-06-15

可以啊

寻路之人

2018-06-12

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。

麦克雷

2018-07-12

虽然知道离耗子叔说的有很大距离，但是会坚持一步步走下去的。

Dawn

2018-07-09

皓子哥，我毕业3年一直做PHP，现在公司又做前端居多，没有code review，而且永远在写业务。
产品流量稍微大一点，后端就被C++团队接手了，被发配写管理后台。
前段时间特别犹豫要不要转Java。
看了您的文章，坚定了我转型的决心，为了自己的长远发展，重头开始！

myco 前

2018-07-01

再请教皓哥一个实践上的问题和学习方法上的问题：

实践问题是：
突然发现C++写个hello world都有很多未解之谜！本以为cout就是个栈里的对象，结果从gbd看它地址和new出来的char[]非常接近，难道是在堆里？猜测cout是在glibc里某处被初始化的；想请教一下cout这个变量到底是在哪里被初始化的，想找到那段代码研究一下。

学习方法上的问题：
其实主最要的问题是我还没找到一条能完整解决问题的链路：上面的问题就是个例子，猜测cout是在glibc里被初始化的，但是到底在哪个文件中，楞是没有搜到。方法链在这里断掉了。再举一个例子，想研究一下main方法调用之前经历了哪些流程、为什么main可以不传参也可以传两个参数，搜到crt1.o里的_start入口，想去glibc里找，结果发现这一块全是crti.S汇编文件，也没找到_start入口。到这里又不知道该怎么办了。想知道我在解决问题的链路上，缺了哪些重要的信息（文档、网站、教程等等）

再小结下我的问题：
1、cout变量在哪里被初始化的，想看源码～
2、main方法为什么可以无参，也可以有两个参数，调用它的地方在哪里，想看源码～
3、想请皓哥推荐些资料，能让我以后自己能找到想看的源码（具体到某个文件）；现在可能想看glibc的，以后可能想看些更贴近内核的代码～

黑小子在路上

2018-06-30

有java经验。没看过java核心卷，直接看java编程思想有没有什么问题么

行沙漠

2018-06-27

C++11还是需要介绍下的，推荐的书都没有涉及到c++11，感觉是个遗憾，毕竟c++11引入了很多革命性特性，lamda，move，forward语意，asyn/task等，和之前的c++相比可以说脱胎换骨了，这是一点建议

刘波3S

2018-06-26

JavaScript是不是和计算机科学关系不大了

颜路

2018-06-22

这是人读的还是机器读的?

kursk.ye

2018-06-21

耗子，两个问题。
(1)你总是说"继承需要给子类暴露一些父类的设计和实现细节"，可是我想了想，除了重写要知道父类方法的参数类型这种必要信息，没有暴露太多细节啊，能不能举例说明哪些信息是不必要的暴露。不过我完全赞成你说继承更多是为了多态。
(2)另外，我每次看编程类的书，其实不是看书，而是在码书，因为很多情况下我只有把书上例子实现一遍，把玩一下代码我才能理解，所以我才能真正理解，这就导致我书"看"得很慢，你以前blog里推荐的JavaScript definition guide我就是这么码完的(不知道什么原因我发现你这次介绍前端时没有介绍这本书)，我不知道你是怎么解决这个问题，因为看你这份书单，我觉得退休前能看完已经不错了，我今年38岁

Geek_c426a3

2018-06-20

讲讲Python吧

宇小跑

2018-06-20

第二版的TCPL是ANSI C。

HA

2018-06-19

我现在有个问题，我毕业到现在一共用过c，c+1，python，golang，ruby，erlang，java等，其中erlang和java接触的不多，但是看代码什么的没什么问题。我曾经也思考过编程语言这方面的问题，到底一门语言我需不需要研究到非常深的理论层面。我有这几个困惑：1. 深入到语言理论层面会占用我大量的时间，因为每个细节都要搞懂，与此同时，实战中用到还很少，投入产出不成比例。ps：这里不代表我一点理论知识都不懂，本人对语言的理解还是很到位的。2. 我总把语言当做一个实现编程思想的表达工具来看待，所以我更看重编程思想，而不关注自己应该掌握哪门语言，对我来还是看实际工作具体用到什么语言。3. 编写高质量代码真的需要那么深入的理论知识吗，因为我觉得每种语言的特点其实很明显，除去编程思想外，由于语言本身特点而影响到高质量代码的编写的可能性没那么高，除非你对这门语言的特点了解的不够深入，再不然就是一些高效库你没了解清楚。以上是我对编程语言的认知，我也看过老师提到的部分书籍，但是都是关于c c++的，这是我入门的语言。我本人还是很重视理论层面的知识，理论知识是指导思想，很重要。希望老师能帮忙解惑。

kuzan

2018-06-16

推荐的关于java的书我都买了，也都看了，确实都是经典，最喜欢java并发编程实战，作者都在jdk源码里挂着名的

Join

2018-06-16

在学校里我们的语言学习路线是，C-C#-Java,毕业设计用Java做的，然后工作了之后用C++，然后一直深耕C++领域，当你工作里面之后再次去看Effective C++,More Effective C++,Inside The C++ Object Model 这样书的时候会有不一样的理解，不同阶段不同经历后读同一本书给你带来的认知是完全不一样的，当你读完Inside The C++ Object Model后你会发现理解问题的粒度更细了，会从对象的布局上考虑，会从内存的分配上考虑，这些带给的是思想上的升华，之后你去学习其他语言的时候是很快的，表面上无非就是一些语法糖的区别，很多背后的思想是一致的。So，耗子哥的推荐Java-C-C++还是有道理的，让你的学习粒度，思考角度逐渐细化，而Go正是处于这两者之间，学学不是坏事

野马

2018-06-16

都是特别干特别干的干货，太棒了！

兔子ORZ

2018-06-15

推荐那本《seven languages in seven weeks》（7周7语言）读起来很爽：）

鸢俊在美国

2018-06-14

惭愧 我工作快二十年了 还没读全这些书 争取退休前读完吧

石头

2018-06-14

精通一门语言！

fsj

2018-06-13

求加入读者群

木子李

2018-06-13

求拉群

米兰的小铁匠

2018-06-13

冒号课堂 这本书是我读过讲编程范式及面向对象最好的书，只可惜已经没有实体书卖了。

google666s

2018-06-13

订阅专栏极大丰富了知识视野，技术提升是没有捷径的，唯有不断学习与实践，耗子哥的文章就是指路明灯。

王赛

2018-06-13

咋不见Rust，kotlin这样网红也实用的语言呢

汇通科技

2018-06-13

学完需要多久

小屎丸

2018-06-12

ruby 怎么样？

沃德啊

2018-06-12

数据结构与算法的书有推荐吗

创意

2018-06-12

rust语言也很火，不知老师为什么不提

右耳朵猫咪

2018-06-12

语言没有好坏之分 只是编程的人有强弱之别~

骑着蜗牛赛跑

2018-06-12

刚刚辞了工作，在培训，开始程序员这一职业，正需要一个指路人，带我入门。

黄超杰

2018-06-12

应该推荐一门动态语言或者函数式语言，不过java最新的也支持函数式编程了

宋桓公

2018-06-12

不知道为毛，学了很多语言，就是有点排斥Java，这次听耗叔叔的，看看那本可以复习c++的书

pomysky

2018-06-12

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

Link

2018-06-12

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

寻路之人

2018-06-12

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。

13683815260

2018-06-12

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。

Jason2Young

2018-06-12

感谢耗子叔

Bernard

2018-06-12

机器学习，区块链，这些怎么处理呢？

chaoqi

2018-06-12

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