开篇词讲C++这么难，为什么我们还要用C++

你好，我是吴咏炜。

C++ 可算是一种声名在外的编程语言了。这个名声有好有坏，从好的方面讲，C++ 性能非常好，哪个编程语言性能好的话，总忍不住要跟 C++ 来单挑一下；从坏的方面讲，它是臭名昭著的复杂、难学、难用。当然，这样一来，熟练的 C++ 程序员也就自然而然获得了“水平很高”的名声，所以这也不完全是件坏事。

不管说 C++ 是好还是坏，不可否认的是，C++ 仍然是一门非常流行且非常具有活力的语言。继沉寂了十多年，并终于发布语言标准的第二版——C++11——之后，C++ 以每三年一版的频度发布着新的语言标准，每一版都在基本保留向后兼容性的同时，提供着改进和新功能。**本专栏主要就是讲这些新特性以及相关的编程实践。**

在讲所有这些细节之前，我想先讲一讲为什么要用 C++，什么时候该用 C++，及如何学习 C++。

C++ 的意义

C++ 程序员应该都听到过下面这种说法：

> C++ 是一门多范式的通用编程语言。

多范式，是因为 C++ 支持面向过程编程，也支持面向对象编程，也支持泛型编程，新版本还可以说是支持了函数式编程。同时，上面这些不同的范式，都可以在同一项目中组合使用，这就大大增加了开发的灵活性。因此，C++ 适用的领域非常广泛，小到嵌入式，大到分布式服务器，到处可以见到 C++ 的身影。

下面是一些著名的用到 C++ 的场合：

- 大型桌面应用程序（如 Adobe Photoshop、Google Chrome 和 Microsoft Office）
- 大型网站后台（如 Google 的搜索引擎）
- 游戏（如 StarCraft）和游戏引擎（如 Unreal 和 Unity）
- 编译器（如 LLVM/Clang 和 GCC）

- 解释器（如 Java 虚拟机和 V8 JavaScript 引擎）
- 实时控制（如战斗机的飞行控制和火星车的自动驾驶系统）
- 视觉和智能引擎（如 OpenCV、TensorFlow）
- 数据库（如 Microsoft SQL Server、MySQL 和 MongoDB）

有些同学可能会觉得，这些应用场景似乎和平时的开发场景有点远啊！你的感觉是对的。有些传统上使用 C++ 的场合现在已经不一定使用 C++，最典型的是个人电脑上的桌面应用。以前 Windows 下开发桌面应用常常用 MFC，微软的 C++ 框架，而现在我估计听说过 MFC 的程序员都不多吧。目前很流行的 Visual Studio Code 主要是用 TypeScript 写的，不是 C++。而我自己也用 C# 写过桌面应用，不过界面逻辑之外的计算和处理仍然是用一个 C++ 的 DLL 来完成。典型情况是，需要性能的组件用 C++ 来写，整个应用程序融合多种不同的语言。

前面我提到了， C++ 的传统领域有被侵蚀的风险，那是因为和它相竞争的语言远远不止一个，可以说是上下夹攻。

- 如果专注性能和最小内存占用的话，C 仍然是首选——嵌入式领域用 C 非常多，而 Linux 也是用纯 C 写的。
- 如果专注抽象表达和可读性的话，那 Python 之类的脚本语言则要方便得多。
- 图形界面（GUI）编程传统上是 C++ 的地盘，但近年来 C# 和 JavaScript 占领了很大一部分市场。
- 游戏算是 C++ 的经典强项了，但有了 C++ 写的游戏引擎，游戏用 C# 写也没啥问题了——你可能不一定知道，Unity 游戏引擎上的首选开发语言是 C#，而王者荣耀是用什么游戏引擎呢？答案正是 Unity——所以王者荣耀可以认为是用 C# 开发的。
- 还有，Go 和 Rust 也加入了战团，对 C++ 形成了一定的竞争……

看起来，C++ 有点危险啊……

不过，真是这样吗？我们需要回到 C++ 的核心竞争力上来看一下。

- 抽象能力：意味着较高的开发效率，同时，更重要的是，不会因抽象而降低性能。
- 性能：这不用多说了，就是快并且占用资源少。
- 功耗：这是近年来我们越来越关注的问题，跟性能直接相关，性能好了功耗自然就低。

计算机在发明的初期，价格奇高，而性能拿今天的标准来看却是极低的，自然不能不关注性能。慢慢地，计算机的性能“足够”了，性能似乎也就不那么重要了，脚本语言于是也有了用武之地。而随着移动设备的普遍使用，大量设备用电池供电而不接电源了，功耗就逐渐成了我们大家关注的大问题。因此，即使主流移动平台的开发语言不是 C++——而是 Java 和 Objective-C 或 Swift——但任何性能要求高的应用，都几乎必然会用到 C++ 开发的组件。

同时，移动设备要联网，也大大刺激了服务器的增加。在服务器端，虽然没有电池电量的问题，但有着服务器集群的供电问题、空调问题、需要的服务器数量问题等，因而 C++ 的使用也是非常广泛的。

前面说到了王者荣耀的客户端是用 Unity + C# 开发的，但我没有说王者荣耀的服务器端——那可还是用 C++ 开发的。另外，有一点我前面还藏着呢！虽然王者荣耀初期是纯用 Unity 开发的，没有用到 C++；但后来，腾讯又用 C++ 把游戏的逻辑部分独立成了一个 GameCore，进一步提高了性能 [1]。

目前，跟 C++ 定位差不多、能有直接竞争关系的，也就是既支持高度抽象、又追求高性能的通用编程语言，其实只有 Rust 一种。而 Rust 远没有达到跟 C++ 一样的成熟和普及程度。这也可以从 TIOBE 的排名看出来：C++ 是第 4 位，而 Rust 是第 25 位 [2]。

另外，和 C 的兼容性，也是 C++ 的一大优势。虽然现在很多大型程序都混杂了多种语言，但在小项目里，减少语言的数量可以简化开发和部署。前不久，我在 Python 里做了一些加解密运算，发现使用的第三方库性能仍不够高，虽然它已经用了 C 开发的加解密引擎。所以，我找了用 C 写的高性能加解密代码，然后使用 pybind11 库 [3]，只手写了一百来行的 C++11 代码，

就把性能又提高了几倍。

什么时候该用 C++?

如此说来，C++ 既然性能又好，又支持抽象，为什么没有更流行呢?

因为代价更高。C++ 是一种复杂的语言，难以上手和熟练掌握，因此也是一种比较容易出错、被误用的语言。C++ 一直与 C 基本保持了向后兼容性，这种兼容性，也一直是 C++ 的安全性和易用性方面的负担。C++ 比起 C 来，要更安全，更不容易出现缓冲区溢出这类漏洞，但跟没有指针概念的语言比起来，它仍然是一种“不安全”的语言。我的个人经验，完成同样的功能，C++ 需要的代码行数一般是 Python 的三倍左右，而性能则可以达到 Python 的十倍以上。

问题来了：你在开发上额外付出的时间，能从性能上省回来吗?

显然，这取决于你开发软件的用途和开发时间。举个例子，如果你用 Python 开发需要一天，运行需要十秒，并且不需要反复运行；那么，转用 C++ 开发就意味着开发费用也许要增加两倍，开发加运行的总时间增加两天，大亏。

反之，如果用Python开发还是需要一天，单次运行需要十秒，但是软件会作为服务长时间运行、每天被调用十万次。在这种情况下，明显你就需要多台服务器来支撑其使用了。这时，如果用 C++ 开发会需要额外的两天，但跟 Python 相比，部署上有望节约十分之九的硬件和电费——那就很值了。

简言之，当你的软件属于运算密集或者内存密集型，你需要性能、且愿意为性能付出额外代价的时候，应该考虑用 C++，特别在你的代码需要部署在多台服务器或者移动设备的场合。反之，如果性能不会成为你开发的软件的瓶颈，那 C++ 可能就不是一个最合适的工具。

此外，在嵌入式应用的场景，那就根本不是值不值、而是行不行的问题。如果程序完成一个功能不能在指定的若干毫秒、甚至微秒内完成，那产品根本是失败、不可用的。在这种场合，能和 C++ 竞争的只有 C，但 C 是一种开发效率更低、更需要堆人力的语言了。在嵌入式开发使用 C++ 的最大障碍可能不是技术，而是人力资源——搞嵌入式开发的程序员可能大多都习惯使用纯 C 了。

由于 C++ 是解决性能问题的利器，短时间里在市场上没有真正的竞争对手，对 C++ 的需求会在相当长的时间里一直存在，尤其在大公司和像金融机构一样对性能渴求的地方。顺便提一句，C++ 之父 Bjarne Stroustrup 目前就职的地方便是摩根斯坦利。

如何学习 C++?

作为很多聪明人使用过的语言，C++ 在某些场合也可能被用来炫技，写出除了本人之外谁都看不懂的高抽象代码。这恰恰是 Bjarne 想努力抵制的方向。他想让 C++ 对初学者变得更为友好，也明确提出过，他不希望 C++ 是一种让人们耍机灵的语言，而是一种让人们更易于使用的语言 [4]。

这同样也是本专栏的一个目标：我希望你能把 C++ 当作一种实用的语言，能用它写出抽象但自然的代码，而非佶屈聱牙、难以卒读的那种。希望我 30 年的 C++ 经验能够给你一点帮助。

学习 C++ 语言就像学一门活跃使用中的外语，你不要期望能够掌握所有的单词和语法规则——那对于世界上 99.999999% 的人来说是不可能的。但语言是服务于人的，语法规则也是服务于人的，是为了让人们能够更好地沟通和表达。虽然 C++ 的每一个新标准都是让语言从定义和规则的角度变得更复杂，但从用法上来说，新标准允许人们能够更简单地表达自己的计算意图。跟学外语一样，我们需要的是多看多写，掌握合适的“语感”，而不是记住所有的规则。

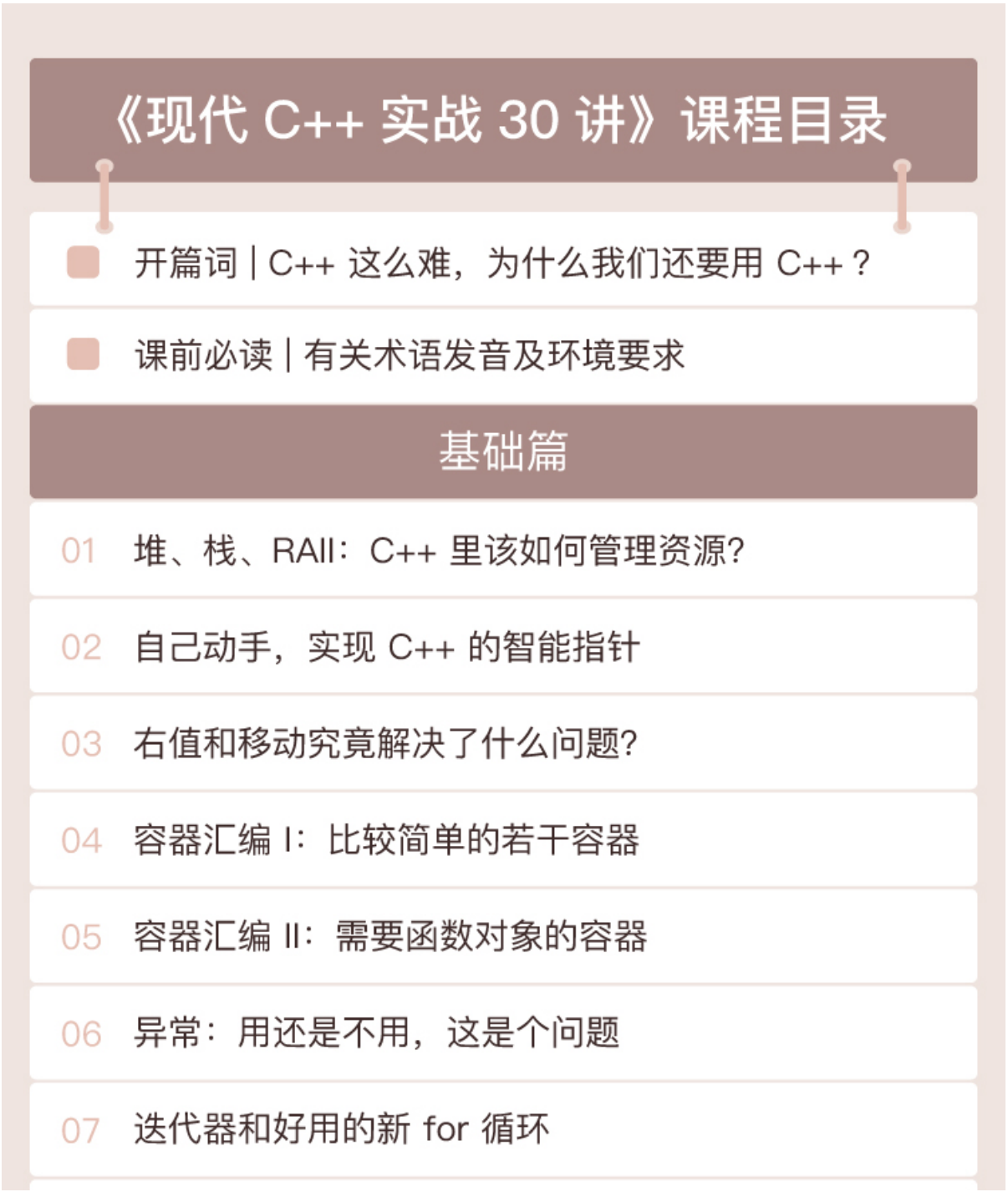
Bjarne 有一个洋葱理论：抽象层次就像一个洋葱，是层层嵌套的。如果想用较低的抽象层次表达较高的概念，就好比一次切过了很多层洋葱，你会把自己的眼泪熏出来的。与这个思路相反，教 C++ 往往有一种不好的倾向，从那些琐碎易错的底层教起，自底向上，使得学生常常在尚未领悟到抽象的真谛之前就已经被 C++ 的复杂性吓翻，从入门到放弃；或者，在学了基本

的 C 语法和 class 之后就满足了，错过了高级抽象带来的全新境界。他主张学习应当自顶向下，先学习高层的抽象，再层层剥茧、丝丝入扣地一步步进入下层。如果一次走太深的话，挫折可能就难免了。

作为专栏而非具体的工具参考书，我会重点讲是什么和为什么，而不是语法细节。同样，我也不讲或少讲技巧，但在需要的地方，我会给出合适的参考资料。希望你在学习了本专栏之后，能够知道某个 C++ 的功能为什么存在和应该在什么情况下使用。那样的话，本专栏的目的就达到了。

具体内容上，本专栏共分为四个部分，详情如下：

- 第一部分——基础篇，讲解现代 C++ 中的最重要特性，帮助你理解基础概念。
- 第二部分——提高篇，讲述几个独立的专题，帮助你掌握 C++ 中的一些高级技巧。
- 第三部分——实战篇，讨论实际的工具和第三方库，帮助你打磨手头的兵器库。
- 第四部分——未来篇，讨论 C++20 中即将引入的一些新特性，帮助你培养前瞻性。



提高篇

实战篇

希望你在完整地学完这四个部分之后，能对现代 C++，这一熟悉而又陌生的语言，有着新的理解，并用它去更好地解决项目中的实际问题。

最后，感谢你的关注。关于 C++ 呢，不知道你有什么样的看法？你学习它的主要原因是什么？你平时使用 C++ 的场景有哪些？欢迎你在留言区和我交流。

参考资料

[1] 游戏葡萄，“《王者荣耀》技术总监复盘回炉历程：没跨过这三座大山，就是另一款MOBA霸占市场了”。https://gameinstitute.qq.com/community/detail/115782

[2] TIOBE Index for November 2019. https://www.tiobe.com/tiobe-index/

[3] Wenzel Jakob et al., pybind11. https://github.com/pybind/pybind11

[4] Interview, “What’s all the C Plus Fuss? Bjarne Stroustrup warns of dangerous future plans for his C++”. https://www.theregister.co.uk/2018/06/18/bjarne_stroustrup_c_plus_plus/

精选留言

阿阳

回首往事，12年啃《c++ primer》第4版，真的很认真，从头看到尾。记得中间面试一家游戏开发的公司，我就手写并默写了该书的一个示例代码，面试官就眼前一亮，跟我谈待遇。那时我记忆力还好，也很喜欢c++。无奈当时应聘了前端开发，从此很少再学习c++了。

c++就像一位老朋友，多年后再见，很激动很期盼。前端的浏览器和js引擎就是用c++写的。要想做好真正的前端，不了解底层是不可能的。期盼能和老师重新学习！

2019-11-25 18:26

作者回复

挺棒的。欢迎返回C++的世界。

2019-11-25 21:02

昨日火冷啊

个人感觉，c++之所以难，还有一个很重要的问题是细节太多，导致每个公司都可能都停留在不同的阶段深耕从而导致互相之间不好交流。 比如我们公司用的是22年前从UNIX C语言转过来的代码。所以C+class处理一切。 而前两天来面试的有20年工作经验的“大牛”确实连深拷贝和浅拷贝都不懂。我自己还面试过另外几个职位，面试官提问题大多是模板类模板函数之类的。。。。所以个人感觉c++的复杂性很大程度上来自于多样性，我们也许在工作只用一小点就够了，但是面试的时候，你永远不知道面试官会问你哪一小点。

2019-11-26 01:33

作者回复

考语法细节的面试官就略low了。C++里语法细节很多，能知道怎么去差就够了。要掌握的是基本概念：指针，引用，递归，泛型，值语义和引用语义，堆和栈，对象的生命周期，等等。

2019-11-26 09:40

metalmac.kyle

曾经十几年前大二那会儿硬啃了TICPP 《C++编程思想》，那也是年少痴狂下仗剑走天涯的时代想投身于安全或游戏领域，误打误撞投身于嵌入式开发，用C多于Cpp，但饱含着对Cpp又爱又恨的情愫，作为一名“不太合格”的程序员想把cpp啃透一直是心愿，希望此刻重拾知识再次启航　　给自己加油，学无止境，保持持续学习的能力其实是一种幸福呢！

2019-11-26 00:50

作者回复

加油，学无止境。

2019-11-26 09:36

HowYoo

做无人驾驶感知系统的开发，尽管快速原型可以用python或者Matlab，但C++几乎是上车的唯一选择，就是因为它兼顾了强大的抽象能力、丰富的表达方式和高性能，尤其是汽车行业计算资源就是嵌入式设备…

2019-11-26 06:25

作者回复

是的，有些场景C++是不二选择。

2019-11-26 09:44

evan_kim

cpp不愧是终极语言 开发Java这么多年 最大的痛苦是 看不到native 来写的本地代码 还有很多高性能框架基本上都是用CPP 来写的 比如容器编排框架 mesos 通过这个专栏能弥补这方面的遗憾吧

2019-11-25 19:55

作者回复

是的，C++ 还是很独特的语言。

2019-11-25 21:09

雷霹雳的爸爸

对于会cpp的我都先鞠一个躬，学习这东西...如果我真打算学的话...那就好像学散打不一定是为了打架一个道理吧

2019-11-25 17:30

coder

参与过clang/llvm项目的开发，也用C++写过商用的编译器，也贡献过不少cpp的开源项目。但是回过头来去看，依然觉得C++还是挺复杂的，自己从来没有系统地学过cpp，因为没时间。在开源项目中看到一些有意思的cpp 用法和特性，就会去devdocs.io和isocpp.org去查看。

这门语言确实很深奥，有时候也很难懂。尤其是template类型系统的推导那里(cpp template是图灵完备的)，太复杂了，再加上

一些左右值的概念，写代码的时候图方便，再配上auto，写着写着就不知道它原始的类型应该是什么了。更糟糕的是，clang llvm本身对cpp特性的支持就不太完善。
总之，这门语言太复杂了

2019-11-27 21:32

作者回复

即使用了auto，还是有编译时的明确类型的，肯定比脚本语言更容易找出类型，也能用一些方法找出auto对象的类型。

为什么会说Clang对C++特性的支持不完善呢？Clang对标准的支持应该是不下于GCC的，我个人认为领先于微软家。

2019-11-27 22:49

WindZQ

从事音视频编解码开发，主要使用CUDA以及Intel IPP。工作中主要以cpp/c为主。

2019-11-26 13:54

作者回复

嗯，适合C++的场景。加油。

2019-11-26 18:30

技术乞丐

最近在学习海康威视的摄像头的SDK，不过里面的代码组织比较看不明白，比如接口的合理设计，接口分层等，这方面老师后面可以提供一个大概学习思路吗，比如SDK设计之类的?

2019-11-26 13:25

作者回复

这个问题比较复杂，我也不知道海康的SDK写得好不好。以常理论，这不是他们的主业，恐怕重视程度不一定很高。建议拿好的开源软件来参考。

2019-11-26 18:30

沧潇雨浪

学校只教了些基础的 想提升一下 特来关注了专栏

2019-11-26 11:31

作者回复

学校教得肯定很浅，不过有了点基础，看这个也许正好。

2019-11-26 18:14

！null

目前从事android camera hal开发，用cpp

2019-11-26 00:06

作者回复

很适合的场景。

2019-11-26 09:37

旧草

[1] 游戏葡萄，“《王者荣耀》技术总监复盘回炉历程：没跨过这三座大山，就是另一款 MOBA 霸占市场了”。这个链接失效了，找了鹅厂的新链接。
https://gameinstitute.qq.com/community/detail/115782

2019-11-25 23:25

作者回复

啊，多谢了。我回头修复一下。

2019-11-26 09:41

廖熊猫

我记得微软C++教程的标题好像有一个就是欢迎回到现代C++，前一段重新入门了一次rust，发现如果有C++底子的话，学起来更容易一些，很多东西都是相通的

2019-11-25 22:48

作者回复

是的，理念上Rust是和C++最接近的语言。

2019-11-26 00:05

小林coding

从事音视频开发、流媒体开发，工作中C/C++语言，前来报到学习^^

2019-11-25 22:30

作者回复

欢迎欢迎，那可正是该用C++的地方了。

2019-11-26 00:13

黄振宇

沙发，支持。好好跟着吴老师学习

2019-11-25 17:04

作者回复

哈哈，你是，够快的。欢迎。

2019-11-25 20:49

贾陆华

我觉得rust成熟之后，有替代c++的可能性，不过还很早

2019-11-27 21:55

作者回复

应该会分掉一部分市场。代替我不觉得会。

2019-11-27 22:47

Encoded Star

庆幸的是，我会很多语言，但我最喜欢的语言是c++

2019-11-27 20:06

作者回复

诚实地讲，我Python和C++都很喜欢。

2019-11-27 20:43

lostsky

引用1里其实说了农药到后来也用c++写了个core

2019-11-27 19:57

家香

本科学完C++基础后也用来写小习作，用着发现其中有很多内容让人即迷惑又吸引人去探寻，但苦于内容太多学起来杂乱。希望能在专栏引导下好好梳理一遍。

2019-11-27 18:49

作者回复

理解为什么和惯用法，在实际项目里进行操练。不用求全的。我对C++都抱着每天学习一点点的态度。

2019-11-27 20:39

seven boo

c++难的不仅仅是语言本身，还有需要c++的业务领域，大型界面，大型后台，编译器，游戏引擎...哪个不是要付出多年的心血才能在这些领域才摸到一点点门槛，更何况再加上c++语言本身的使用难度。所以要找工作的新手而言真的不推荐c++为入口

2019-11-27 18:34

作者回复

对的。我不认为 C++ 该是学习编程的第一门语言。

2019-11-27 20:34