链滴

十年学会编程

作者：Roger

原文链接：https://ld246.com/article/1461146478177

来源网站：链滴

作者 Peter Norvig 是计算机科学家，Google 的研究总监。在本文中，Peter Norvig 会告诉你为什么急功近利地学习软件开发技术是没效果滴？
==========华丽的分割线==========
为啥都想速成？
随便逛一下书店，你会看到《7 天学会 Java》等诸如此类的 N 天甚至 N 小时学习 Visual Basic Windows、Internet 的书。我用亚马逊网站的搜索功能，出版年份选 1992 年以后，书名关键词是“天”、“自学”、“教你”，查到 248 个结果，前 78 个是计算机类图书，第 79 个是《30 天学孟拉语》。我用“天”换成“小时”，结果更惊人，有多达 253 本书，前 77 本是计算机图书，第 78《24 小时自学语法句式》。在前 200 名中，96% 是计算机的书。
结论就是：要么人们急于学习电脑，要么计算机比其他东西学起来要异常简单。没有任何书是关几天学习贝多芬或量子物理的，甚至连犬类装扮都没有。费雷森（Felleisen）等人在其著作《如何设程序》中同意这个趋势，其中提到：“坏设计很简单，笨蛋才用 21 天学，尽管他们还是真傻。”</p

让我们看看《三日学会 C++》这个书名意味着什么：

学习：三天内你可能没有时间写出有意义的程序，或者从中积累经验。你不可能有时间去跟职业程者一起去理解在 C++ 环境下的状况。简而言之，你没有充足的时间学很多。所以这本书只能说肤的知识。正如亚历山大·波普（Alexander Pope）所言：一知半解是很危险的。
C++：三天内你可能学会 C++ 的句法（如果你已经了解其他的语言），但你还不会使用它。打比方，假如你是个 Basic 程序员，你可能写出 Basic 风格的 C++ 程序，而无法理解 C++ 的真实好。那要点是什么？艾伦·佩里斯(Alan Perlis)曾经说过：“一门不能影响你编程观点的语言不足学的。有可能你学了一点点 C++（或者诸如 Javascript、Flex 之类），因为你需要和现成的工具接口以完手头的任务。这种情况下，你不是在学习如何编程，只是在学习如何完成任务。
三日：不幸地是，这远远不够，下一部分会详细讲。

如何用十年掌握编程
研究人员（Bloom (1985), Bryan & Harter (1899), Hayes (1989), Simmon & Chase (1973)）得出结论：想要在诸多领域达到职业水平需要十年，比如国际象棋，作曲，电报操作，绘画弹钢琴，游泳，网球以及神经心理学和拓扑学的研究。关键是精心练习，只是一遍一遍地重复是不够，必须挑战恰好超越你能限的事情，尝试并思考你的表现，并自我矫正。周而复始。这并无捷径！</>
4 岁的音乐奇才莫扎特用了 13 年才能创作世界级的音乐。另外，披头士乐队似乎在 1964 年的德·苏利文（Ed Sullivan show）演出中一炮而红，但是他们自从 1957 年就在利物浦和汉堡的酒吧出，在取得广泛关注后，第一部重量级作品《佩珀军士》（Sgt. Peppers）是在 1967 年发行。马尔姆·格拉德威尔（Malcolm Gladwell）撰文描述了一项针对柏林音乐学院学生的研究，他们被分为尖，中等和不足三类，并被问到他们练琴的情况：
所有三组中的人，开始学琴的年龄大概相差无几，五岁左右。在刚开始的几年，所有人练习量也不多，一周两三个小时。自八岁开始，实质性变化就有了。那些精英学生开始比其他人练习更多：九的时候一周六个小时，十二岁的时候一周八个小时，十四岁的时候一周十六个小时，一直到二十岁的候一周要超过三十小时。截止到二十岁，在他们的生涯里已经有总计一万小时练琴。仅仅表现可以的部分学生加起来是八千小时，那些未来的音乐老师有四千小时。
所以，更确切地说，一万小时，而非十年，是个神奇之数。
萨缪尔·约翰逊（Samuel Johnson, 1709-1784）认为还需更长时间：“卓越乃一生之追求，而其它”。
乔叟（Chaucer, 1340-1400）抱怨道"the lyf so short, the craft so long to lerne."（生之有，学也无涯）。
希波克拉底（Hippocrates, c. 400BC）因这句话被世人所知："ars longa, vita brevis"（译注：丁语，意为“艺无尽，生有涯”），更长的版本是 "Ars longa, vita brevis, occasio praeceps, experimentum periculosum, iudicium difficile"，翻译成英文就是 "Life is short, (the) craft long, opportunity fleeting, experiment treacherous, judgment difficult."（生有涯，艺无尽，机遇瞬逝，践行导，决断不易）。
我的编程成功秘笈是：

首先要对编程感兴趣，能从编程中得到乐趣。一定要让它足够有趣，因为你要保持你的兴趣长达年。
与别的程序员交流；阅读别人的代码——这比看任何书或参加培训课都重要。
实践。最好的学习乃实践。俗话说：“编程的至高境界一定要通过充分的实践才能达到，而个人能力可通过不懈努力获得显著提升。”（p. 366）“最有效率的学习需要明确的目标，适当的难度，知回馈，并容许重复或修正错误。”（p. 20-21）《实践认知：每日的思维、数学及文化》（Cognition i Practice: Mind, Mathematics, and Culture in Everyday Life）在这方面可做参考。
如果你愿意，花四年学习大学课程（或者再加上读研）。这将给你赢得某些工作机会，并给予你该领域的深层见解。但如果你不喜欢学校的学习，你同样可以在工作中获得相似的经验。无论如何，靠书本是远远不够的。“学习计算机科学不会让你成为编程专家，如同学习绘画和色彩理论不会让你为画家一样”。这是埃里克·雷蒙德（Eric Raymond）说的，他是《新黑客字典》（The New Hacker' Dictionary）的作者。我雇用过的最优秀程序员，只有高中文凭。但他开发过许多伟大软件，有自己新闻组，通过公司认股赚的钱就让他买下了自己的夜店。
和其他程序员一起参与工程项目。在某些项目中担当最优秀程序员，在另一些项目中充当最差劲序员。充当领头羊的时候，你要测试你领导一项工程的能力，并用你的视野来激发他人；如果在项目中垫底，就应该学习其它牛人在做些啥，以及他们不喜欢做的（看他们把哪些活让给你做）。
继续别人的工程项目。去理解先前程序员写的程序。学习如何理解并解决先前程序员没有考虑到问题。思考你的程序该如何设计以便让之后的程序员更容易维护。
至少学 6 种程序语言。其中包括一种支持类抽象的（Java 和 C++），一种支持函数抽象的（如 Lsp 或 ML），一种支持语义抽象的（Lisp），一种支援声明规范的（如 Prolog 或 C++ 模板），还一种支援协程的（Icon 或 Scheme），另外一种支持并发的（Sisal）。
记住，在“计算机科学”里有“计算机”一词。理解计算机执行你的代码的时候花费的时间。比：从内存中取一个字（考虑有无缓存未命中情形），连续从磁盘读字，或者在磁盘中定位。
参加语言标准化工作。这可能是有关 ANSI C++ 委员会，也可能是决定你编码风格是两格缩进四格缩进。无论如何，你要知道其他人对语言的喜好程度，有时还要想想他们为什么喜欢这样。
知道自己应该在何时脱身于语言标准化。

所有上述这些，很难通过书本的学习来达到。我头一个孩子出生时，我读了所有的“如何做”（ow To）系列的书籍，却依然对育婴毫无头绪。30 个月后，我第二个孩子出生，我还需要温习一下些书吗？绝对不！相反，我完全可以参照个人经验，而结果相当有效。这更让我确信：我的经验胜过些专家们写的上千页文字。
弗雷德·布鲁克斯（Fred Brooks）在《没有银弹》（No Silver Bullet）一书给出了寻找顶级设师的三条建议：

尽早系统地识别出顶级设计师。
分配一个人作为其职业规划的导师。
给予机遇让成长中的设计师互相磨砺。

此处假定有部分人已经有成为伟大设计师的潜质，你所需的就是要诱导他们。艾伦·佩里斯（Alan Perlis）一针见血地指出："假如人人都可以学雕刻，那就得教米开朗基罗如何不去干雕刻。对于伟大序员，也是如此。”
所以，简单地买一本 Java 书，你或许能找到些有用的东西，但绝不会让你在 24 小时内甚至 24 抑或 24 月内，成为行家里手。
==========华丽的分割线==========
转载：
https://plus.oogle.com/113559088971921339544/posts/UpQodJZHDLx