关于写书

关于写书

写书计划已经进入策划阶段。这篇文章是因为之前那篇的最后部分的一些想法越改越长，开始具有独立的价值，所以截取下来放在这里。

动机

说了挺久要写书，一直没有动力。一方面是由于我觉得国内出版社都不大靠谱，出的很多书水平太低，处于培训班级别，教流水线工人死知识那种，或者就是把别人软件的使用说明书翻译了拿来出版，要不就是翻译国外教材。这有损我的品牌形象。从大学二年级开始，我就不看中文教科书和技术类书籍，也不看翻译过来的国外教材，就是因为质量低劣。有些国外教材本来还行，翻译过来就走样了。

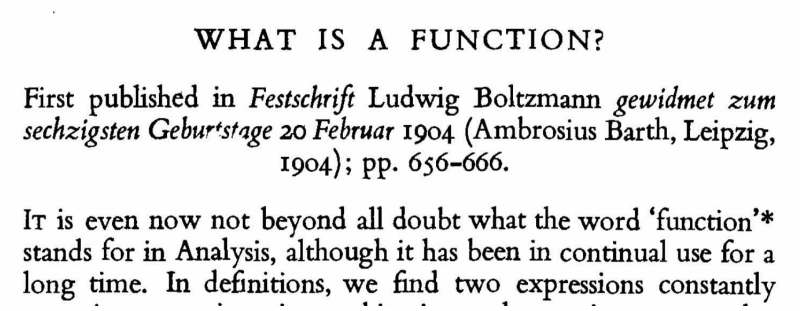
另外很大一部分原因，是因为我写东西都是一针见血，容易懂，而且不会留一手。我很怕一不小心说太明白，就让关键的知识点落到道德水平低的人手里，拿出去装模作样吹牛逼，不告诉别人那想法是哪里来的，还反过来说我是吹牛逼。我也许顾虑太多了。其实我只需要传授我愿意提供的知识，而保留我不愿意的就行。

我写“入门书”而不是“进阶书”的一个原因，就像爱因斯坦说的：如果你不能向一个六岁小孩解释清楚一个问题，那么你其实并不真的懂。我在大学里见过太多讲不清楚问题的教授，中国的美国的都有，后来我发现那是因为他们自己都没弄明白。没有非常深入的见解，你是不可能把深奥的东西解释清楚的。反过来，试图把一个问题向完全不懂的人讲清楚，也会大大加深你自己的理解。看了我的『怎样写一个解释器』而学会解释器的人都会明白，我的理解程度在全世界处于什么地位。没有成千上万次写各种各样解释器的试验，失败和领悟，你是不可能理解到那种程度的。

深入理解任何一门学问的关键，不是试图去回答越来越“高级”，越来越复杂的问题，而是试图去回答最基础的问题，反复地问自己最基础的问题…… 爱因斯坦之所以能发现相对论，不是因为他去思索看起来高级的难题，而是因为他去思索一个最基本的问题：时间是什么？其他人觉得这问题很傻，时间不就是一秒一秒过去的那个东西吗？现在是半夜两点，那就是时间！然后这些人就永远没机会发现相对论了。同样的，深入理解计算机科学的关键，不是去学习云计算，大数据或者区块链，而是去思索最最基本的问题：“计算是什么？”“程序是什么？”“函数是什么？”“变量是什么？”“抽象是什么？”…… 你觉得自己知道这些问题的答案吗？那请你再想一想！

实际上直到 20 世纪初，全世界没有一个数学家真正的理解“函数是什么？”这个如此基础的问题。这些人却天天都在用“函数”这个词，以至于他们的定理和证明里面出现各种奇怪的错误。直到 1904 年 Frege 写了这篇论文“What Is A Function?”这种情况才得到了改善。数学发展了几千年，居然没有人真的理解如此基础的，天天都在用的概念。他们以为自己明白了，所以根本没有仔细思考过它是什么。就像我们从来没思考过什么是时间，却天天都在谈论“需要多少时间”一样。

为了感受一下这个问题，我请大家来读一读这篇文章的第一句话：



WHAT IS A FUNCTION?

First published in *Festschrift* Ludwig Boltzmann *gewidmet zum sechzigsten Geburtstage* 20 *Februar* 1904 (Ambrosius Barth, Leipzig, 1904); pp. 656-666.

It is even now not beyond all doubt what the word 'function'* stands for in Analysis, although it has been in continual use for a long time. In definitions, we find two expressions constantly

回答最基础，看上去谁都知道的问题，也将会是我这本书的开端。Frege 是一个不幸的人，他的作品在他有生之年都不被人重视。我比他幸运一点，我的博客还有一些读者 :)

所以这本书虽然被我叫做“普及”或者“入门”读物，但它并不只是针对初学者的：它针对所有人。对我来说，很多“资深”的程序员其实根本就不算入了门。当我进入研究生阶段的时候，偶然发现了 SICP，看了这本所谓“入门书”，我惊讶地发现自己以前其实不会编程。在美国工作的时候，我发现很多高级别的程序员也是一样的情况。他们以为自己懂了，资历很深了，而其实还差很远。由于一些初级问题一开头就没理解清楚，到了关键的时候就会犯错误。这就是我

所谓“入门”的含义。所以这本书也可以作为资深程序员们的进修读物。当然我会降低门槛，努力让初学者都能看懂。

与经典书籍的区别

因为我好像很推崇 Lisp/Scheme 语言。有些人看我想写这种入门读物，可能以为我会写一本“王垠的 Little Schemer”或者“王垠的 SICP”。这是一种比较常见的误解。如果我只是模仿 The Little Schemer （TLS） 或者 SICP，那是完全没有意义的。你去读那些书的中文版就行了。

很多年前我就是从 SICP 入门的，但是经过多年的研究，直接跟这些书的作者们学习交流，我发现这些书虽然贡献卓著，是不可磨灭的经典，我尊敬它们的作者，可是它们也有很多不足的地方甚至误导（这句话不要传到某些人耳朵里哈）。这就是为什么有好几个出版社请我翻译 TLS，我最后都拒绝了，因为我想写很不一样的东西。

很多人曾经问我：“我该看这本书还是那本书？”我都不想回答，也是类似的原因。因为我的脑子里有一本更好的书，我觉得回答这样的问题有点降低自己的身份。我不再是此类书籍的消费者，而是创造者的级别。出于尊重的原因，你不可以问一个创造者这样的“消费级问题”。这就像你不可以问法拉利的设计者：“我是买奥迪好还是奔驰好？”出于礼貌他也许会给你一个回答，但他的内心会很受伤 ;p 同理的，请我翻译别人的此类书籍，也让我感觉很悲哀。

我在之前的好几篇文章已经指出了 Scheme 语言的一些设计上的弱点，完全以 Scheme 的方式写书，显然会把很多这样的弱点当成优点，对新手造成误导。从 SICP 或者 TLS 入门的学生，很多片面地认为 Lisp （或者 Haskell，Scala）是世界上最好的语言，以为 Lisp 的 list 是世界上最好的数据结构，以至于写 Java 代码还要在里面自己造出 Lisp 的 list 结构，搞得又复杂效率还很低。我不希望给我的读者们造成这样的效果，因为很显然我知道 list 的缺点。

我希望我的书是一本有机融合多种思维方式的精华，它应该本着科学的态度，而不是宗教的。它不会包含函数式语言的宗教，也不会包含面向对象方法的宗教，但它却会包含它们的精髓，把它们无缝的融合和统一在一起。这本书要教会读者的，不是某一种语言或者某一种思维方式，而是所有的语言和所有的思维方式的精华结合在一起，并且提醒你小心它们的缺点。

另外，我发现 SICP 这样的书籍还有很多写作上的弱点，很多地方有没必要的细节和冗长，很多例子需要比较困难的数学才能理解，导致初学者读起来头痛。书中代码的实现有些时候并不简洁清晰，过度抽象，到了第四章就很难看懂了。实际上 SICP 根本不适合初学者，他们需要先从其它地方学会编程，然后再来看这本书。虽然 SICP 曾经是 MIT 本科生的入门读物，但大部分 MIT 学生进学校之前就已经会一些编程了。所以 SICP 只适合作为进阶读物。

TLS 的“孔夫子式”写作方式很精悍，比起 SICP 之类的教材有很多优点，但它还是可以很伤脑子看不懂。显然 TLS 不是一本入门教材，它一开头就直接冒出各种术语（atom，list，s-expression……），好像假设你已经知道了一样。除非你已经学过 Scheme，否则将寸步难行。过度的强调递归，会导致学生倾向于在工作中过度使用递归代码，而忽视了循环语句的重要性。另外 TLS 缺少跟实际工作接轨的内容，这会让读者看了书却不知道怎么改善工作要用的代码，以至于失去动力，迷茫，半途而废。

我曾经很推崇费曼的物理学讲义，可是实话说吧我真想再学点物理，所以看了一些费曼的讲义。感觉开头好玩，到后来还是很累很痛苦想睡觉…… 所以我需要探索更好的方式来表达这些内容。这本书不会再号称“计算机科学的费曼讲义”，它应该更好！如果它不是更好，我就继续改进它 :)

为了知识的民主和社会的文明，提高普通大众的技术教育水平迫在眉睫。这些事情我不放心其它人来做，更害怕发言权落到吹牛扯淡的野心家手里。仔细看过我的技术文章的人，都应该知道它们的见识深度是很难超越的。所以很希望大家能够支持我开张写书。祝愿大家走出迷茫，获得真知！

启动经费和投票支持

我希望在书的第一章发布之前，也就是现在，收集一些“启动经费”，来开始写作的过程。这些经费用于建立工作环境，也用于“估算”有多少人会想买我的书。

如果你喜欢这篇文章，而且有意要买我将要写的书，可以点击这个付费页面，对本文进行少量的付费（30元），留言就写“期待CS入门书”。之前已经付过类似费用的就不用了。我会根据付费的人数来估计图书将来的销量，所以你如果感兴趣的话，请一定向我发出你的支持。但是请注意，这个付费不代表你付了买书的钱。我的书显然不会这么便宜的。由于这篇文章本身的价值，你是在给这篇文章付费 :)