计算机科学阅读班招生说明

计算机科学阅读班，是一门零基础，目标在于短期内掌握计算机科学精髓的课程。它是我 20 多年的计算机学术和工程实践，加上两年面向社会的教学实验的结晶。课程吸取了世界上主要的计算机入门教学方式的优点，避免了它们阻碍初学者理解的各种问题，以至于完全零基础的学生也可以在短短两个月之内，掌握大学博士阶段才可能学到的精华内容。这些内容足以建立起坚实的知识基础，使得他们对于理解计算机科学的其他方面从容自如。

课程使用特别的授课方式——阅读加练习辅导的方式。阅读内容为我正在写的书《Ground-Up Computer Science》（GUCS），练习辅导是一对一的微信辅导方式。虽然课程只有 7 节课，但是由于每个练习会及时得到提示和反馈，实际的学习时间大大超过传统课堂。这种教学方式避免了普通课堂的各种不灵活性，给予了学习者最大的方便和灵活性。这使得工作学习繁忙的人士也能抽空完成学习。一般学生能够在两个月之内完成课程。对于比较繁忙的学生，时间可以放宽到 4 个月。

现在阅读班的形式已经基本成型，我觉得适合长期面向社会招生。

适用人群：

- 从中学到博士阶段的各专业学生，不管文科还是理科生都能学会。虽然课程已经成功让一个 13 岁少年深入地掌握了这些内容，但对于中学生，希望学生本人对计算机科学感兴趣。如果只是家长的期望，恐怕学生会比较难以用心去阅读。
- 已经从事工作的人员，包括 IT 从业人员，IT 管理类工作和其它工程类工作人员。
- 其它各类对计算机感兴趣，想把它作为业余爱好的人员。

学费和报名方式：

- 学费对于已经工作人员是 10240/人，对于无收入的在校学生（中学，大学本科，但不包括研究生和博士生）是 8192/人。对于经济特别困难，但有求知欲望的学习者，可以来信询问特殊价格。
- 报名请发送 email 到 yinwang.advising@icloud.com。标题为《计算机科学阅读班报名》。来信请说明自己的基本信息，附件发送一个简历，提供微信联系方式。请写一段 200 字左右的“个人说明”，说明你的求学动机。对于符合要求的求学者，我会进行简短的微信语音面试。
- 报名者可以要求一节课的试课。试课费用是 300 人民币。

教学语言。课程目前使用 JavaScript 作为教学语言，但并不是教 JavaScript 语言本身，不会使用 JavaScript 特有的任何功能。课程教的思想不依赖于 JavaScript 的任何特性，它可以应用于任何语言，课程可以在任何时候换成任何语言。学生从零开始，学会的是计算机科学最核心的思想，从无到有创造出各种重要的概念，直到最后实现出自己的编程语言和类型系统。

课程强度。课程的设计是一个逐渐加大难度，比较辛苦，却很安全的山路，它通往很高的山峰。要参加课程，请做好付出努力的准备。在两个月的时间里，你每天需要至少一个小时来做练习，有的练习需要好几个小时才能做对。跟其他的计算机教学不同，学生不会因为缺少基础而放弃，不会误入歧途，也不会掉进陷阱出不来。学生需要付出很多的时间和努力，但没有努力是白费的。

课程大纲：

第一课：函数。跟一般课程不同，课程不从所谓“Hello World”程序开始，也不会叫学生做一些好像有趣而其实无聊的小游戏。一开头我就讲最核心的内容：函数。关于函数只有很少几个知识点，但它们却是一切的核心。只知道很少的知识点的时候，对它们进行反复的练习，让头脑能够自如地对它们进行思考和变换，这是教学的要点。我为每个知识点设计了恰当的练习。

第一课的练习每个都很小，只需要一两行代码，却蕴含了深刻的原理。练习逐渐加大难度，直至超过博士课程的水平。我把术语都改头换面，要求学生不上网搜索相关内容，为的是他们的思维不受任何已有信息的干扰，独立做出这些练习。练习自成系统，一环扣一环。后面的练习需要从前面的练习获得的灵感，却不需要其它基础。有趣的是，经过正确的引导，好些学生把最难的练习都做出来了，完全零基础的学生也能做出绝大部分，这是我在世界名校的学生里都没有看到过的。具体的内容因为不剧透的原因，我就不继续说了。

第二课：递归。递归可以说是计算机科学（或数学）最重要的概念。我从最简单的递归函数开始，引导理解递归的本质，掌握对递归进行系统化思考的思路。递归是一个很多人自以为理解了的概念，而其实很多人都被错误的教学方式误导了。很多人提到递归，只能想起“汉诺塔”或者“八皇后”问题，却不能拿来解决实际问题。很多编程书籍片面强调递归的“缺点”，教学生如何“消除递归”，却看不到问题的真正所在——某些语言（比如 C 语言）早期的函数调用实现是错误而效率低下的，以至于学生被教导要避免递归。由于对于递归从来没有掌握清晰的思路，在将来的工作中一旦遇到复杂点的递归函数就觉得深不可测。

第三课：链表。从零开始，学生不依赖于任何语言的特性，实现最基本的数据结构。第一个数据结构就是链表，学生会在练习中实现许多操作链表的函数。这些函数经过了精心挑选安排，很多是函数式编程语言的基本函数，但通过独立把它们写出来，学生掌握的是递归的系统化思路。这使得他们能自如地对这类数据结构进行思考，解决新的递归问题。

与一般的数据结构课程不同，这个课程实现的大部分都是「函数式数据结构」，它们具有一些特别的，有用的性质。因为它们逻辑结构清晰，比起普通数据结构书籍会更容易理解。与 Haskell 社区的教学方式不同，我不会宗教式的强调纯函数的优点，而是客观地让学生领会到其中的优点，并且发现它们的弱点。学会了这些结构，在将来也容易推广到非函数式的结构，把两种看似不同的风格有机地结合在一起。

第四课：树结构。从链表逐渐推广出更复杂的数据结构——树。在后来的内容中，会常常用到这种结构。树可能是计算机科学中最常用，最重要的数据结构了，所以理解树的各种操作是很重要的。我们的树也都是纯函数式的。

第五课：计算器。在熟悉了树的基本操作之后，实现一个比较高级的计算器，它可以计算任意嵌套的算术表达式。算术表达式是一种“语法树”，从这个练习学生会理解“表达式是一棵树”这样的原理。

第六课：查找结构。理解如何实现 key-value 查找结构，并且亲手实现两种重要的查找数据结构。我们的查找结构也都是函数式数据结构。这些结构会在后来的解释器里派上大的用场，对它们的理解会巩固加深。

第七课：解释器。利用之前打好的基础，亲手实现计算机科学中最重要，也是通常认为最难理解的概念——解释器。解释器是理解各种计算机科学概念的关键，比如编程语言，操作系统，数据库，网络协议，Web 框架。计算机最核心的部件 CPU 其实就是一个解释器，所以解释器的认识能帮助你理解「计算机体系构架」，也就是计算机的“硬件”。你会发现这种硬件其实和软件差别不是很大。你可以认为解释器就是「计算」本身，所以它非常值得研究。对解释器的深入理解，也能帮助理解很多其它学科，比如自然语言，逻辑学。