计算机科学基础班（第5期，现实接轨班）招生

计算机科学基础班，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。

第 4 期基础班于 2021 年 5 月进行之后，就再也没有开过基础班。现在经过了进阶班和 continuation 和并发计算专项班的巨大成功，我决定再次开展从零开始的基础班。欢迎有志人士报名。

这次课程相比之前的几届，我会增加一些“现实接轨”内容。通过几种当今最流行的语言的特性，掌握现实工程实践中能用到的概念。这些“接轨语言”初步计划是包括 Python，Java 和 Rust。通过理解它们，掌握“面向对象”，“静态类型系统”和“内存管理”等重要概念。

所以这一期的基础班，不但会打下“递归”这样的坚实基础，理解“解释器”这样的深入内容，而且会更容易将这些精髓概念与现实工程接轨。

课程方式

- 课程使用 Zoom 视频会议的方式进行教学。
- 每周一次课，大概 2 小时。
- 每周有一次自由讨论会，为同学们统一答疑。
- 每个学生会被加入自己独有的辅导群，会由经验丰富的助教进行辅导，并且由我监督和指点，保证学生不卡在不必要的环节。
- 学生会被加入第 5 期基础班的大群，方便同学之间交流。

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。之前的课程已经成功让一个 13 岁少年深入地掌握了这些内容，并且超过了很多经验丰富的成年学生。
- 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

- 学费统一为 12800/人。
- 报名请发送 email 到 yinwang.advising@icloud.com。标题为《计算机科学基础班第 5 期报名》。来信请说明自己的基本信息，附件发送一份简历，提供微信联系方式。请写一段 200 字左右的“个人说明”，说明你的学习动机。对于符合要求的求学者，我会进行简短的微信语音面试。

开始时间

课程计划于 2024 年 1 月开始，具体的时间会根据报名情况而定。课程中间如果遇到春节等节假日，会短暂休息。

教学语言

课程开头会使用 JavaScript 作为教学语言，因为 JavaScript 易用而且应用非常普遍。但课程并不是教 JavaScript 语言本身，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。

为了使学生能够和现实的编程工作接轨，这一期的课程会在最后加入其它三种“现实语言”。目前初步的计划是 Python，Java 和 Rust。我会讲授每种语言特有的功能，比如“面向对象”，“静态类型系统”，“内存管理”等等。当然，这些特性都会建立在之前学会的基础上。

对于每种语言我会增加一节课，一次讨论会，和一周的辅导时间。根据进阶班的成功经验，我觉得这些语言每种增加一节课，应该足够。

课程大纲：

第 1 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 1 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 2 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 3 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 4 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 5 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 6 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 7 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。

第 8 课：Python。利用 Python 语言，讲述所谓“面向对象”编程方式。之前的课其实已经用到“面向对象”的精华思想，所以现在用 Python 只是为了和“现实编程”对接一下，让同学们理解之前学到的思想能如何对应到现实的工作代码中。这节课也会覆盖其它命令式语言的基本用法，比如数组，赋值，循环等等。

因为“AI 热”，Python 可能是目前最热门的语言。我个人不在乎 AI，但 Python 作为理解“面向对象编程”的工具，我觉得是一个不错的选择。虽然有些人（比如我）可能不在乎“面向对象”，但理解别人心目中的“面向对象”概念，对于实际工作也是有帮助的，否则你会很难理解这样的代码。

第 9 课：Java。利用 Java 语言，引入“静态类型系统”这个重要工具。静态类型系统是创建大型工程，保证基本质量的重要工具。通过 Java 的类型系统，同学们可以掌握基本的静态类型检查规则，泛型等重要概念。这个基础会为其它类似的语言，比如 C++，C#，Rust 等打好基础。

Java 虽然备受很多“函数式程序员”诟病，说它过度复杂，但只要你学会了精髓的编程技巧，一样能用 Java 写出简单而优质的代码。而且 Java 至今仍然是 Web 服务器领域用得最多，相对而言最可靠的语言。

第 10 课：Rust。Rust 是当今最热门的新兴语言，但里面的功能其实大部分都是其它语言早就有的，只不过 Rust 选择了它们最好的一些部分，也当然包含一些自己的设计错误。Rust 独有的特性是“静态内存管理”，也就是不通过“垃圾回收”（GC），而是通过静态类型系统来保证内存的安全性。我们会通过这一节课，掌握内存管理的基本原理，理解 Rust 独有的 ownership，lifetime 等重要概念。如果有时间，我们也会利用 Rust 来理解其它一些重要的类型系统特性。

通过 Rust 理解了内存管理，也会帮助同学们理解 C，C++ 等完全“手动内存管理”的语言，为底层的系统编程打下基础。