第 3 讲 程序与递归：组合-抽象-构造

1、快速浏览---本讲视频都讲了什么?

【视频 3.1 计算系统与程序-程序的作用和本质】

“程序”的概念对计算学科学生而言是非常重要的概念。为什么要编程序？是为了告诉计算系统（已经被制造出来的）其应该做什么怎么做？有了程序的概念，制造任意复杂的计算系统都变得简单了：仅制造简单的动作，任意复杂的动作都可通过编程序来实现，因此计算系统是能够执行“程序”的系统。怎么编程序—组合、抽象与构造，那什么是组合？什么是抽象呢？请看视频 3.1 给出的讲解，“抽象”是可学习可训练可操作的，… …。

【视频 3.2 程序构造示例 I-计算对象的定义-构造与计算】

程序是构造出来的。怎样构造呢？本视频从基本的运算式及其组合来看程序的构造。

一个运算式被嵌入到另一个运算式中，是一种组合，是一种构造。一个运算式 A 被嵌入到另一个运算式 B 中，其意义是将 A 的运算结果作为 B 的计算对象，参与 B 的计算。若干个基本的运算式通过嵌入组合的方式被构造成一个复杂的运算式--这即是“程序”，告诉计算系统按照此嵌入组合的运算关系进行计算便能得到结果。

当一个复杂的运算式被重复作为计算对象参与程序构造时，为简化书写，可以用一个名字来表示这个运算式，这就是计算对象的定义、构造和计算。这便是一种抽象：命名计算对象、使用名字参与新的“程序”的构造、“程序”执行时则以被命名的那个运算式替换该名字以便计算。

定义、**构造**和**计算**是学会编写程序的关键之关键，需要很好的理解--**本视频通过示例来展现和讲解如何通过组合、定义、构造和计算来构造程序的基本思维… …。**

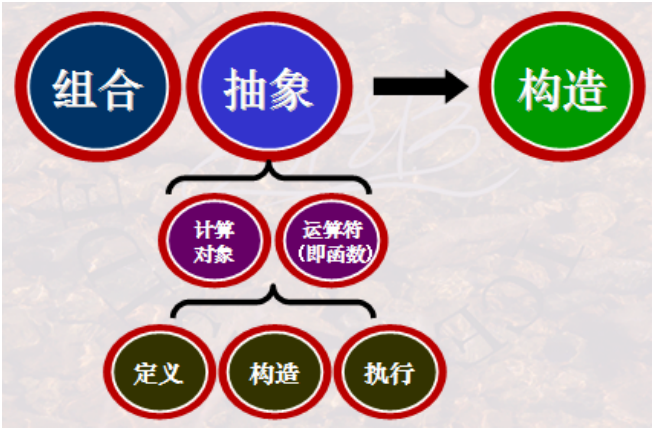
【视频 3.3 程序构造示例 II-运算符的定义-构造与计算】

运算式包括计算对象和运算符。基本的运算符是由计算系统直接提供的。而计算系统还提供了一种**可由用户自主定义运算符**并**使用自定义运算符**的能力--这就是运算符的定义、构造与计算，这种自定义的运算符，被称为函数或过程。

函数或过程可以被认为是将一组程序定义为一个名字，利用该名字可以构造新的更为复杂的程序。而此复杂程序在执行时或说计算时，执行了一个“替换”操作，即将该名字，替换为其所表达的“程序”，这是程序编写与执行的基本机制。

【视频 3.4 程序构造示例 III-条件组合式的构造与总结】

一些复杂的运算式，如当满足这个条件时按这个运算式计算，当满足那个条件时按那个运算式计算，如何表达这种遵循不同条件选择不同运算式运算的情况呢，此视频有简要介绍… …。此外，该视频还对前述视频内容做了简单总结，如下图… …。



【视频 3.5 递归的概念】

让机器进行计算，目的是利用简单的计算规则来进行大量重复的计算。如何告诉机器执行大量重复的计算，这就需要“递归”。递归是计算学科很重要的概念和手段，将来理解复杂的程序，一定要理解递归。本视频告诉你理解递归要理解其三个层面的内容，哪三个层面呢？

【视频 3.6 原始递归函数-复合与递归】

原始递归函数，是用严格的数学方法表达**组合**与**一层层构造**的问题，其基本思想是试图用一组已定义的函数来构造新的函数。**“复合”**解决了视频 3.2 中的嵌入组合的数学表达问题，**“原始递归”**解决了**一层层构造一系列新函数**的数学表达问题。

原始递归函数是理解“递归构造”这个计算学科最基本思维模式的基础，应该说是最简单的，但因为其比较“拗”，就像是盘旋型楼梯，攀登起来有些头晕，怎么克服呢？就是按照视频中讲解示例的方法，自己应用“复合”和“递归”一步步地构造一些新函数来体验，即由一些已知的 f 和 g，亲自推导计算一下 h(0), h(1), h(2), … …的函数形式，亲自体验是学习递归的重要手段，… …。

【视频 3.7 两种不同的递归函数-递归与迭代】

两种不同的递归函数，一个是斐波那契数列，一个是阿克曼函数。二者都是用递归的形式进行定义，但斐波那契数列可以用迭代的方法来执行来计算，而阿克曼函数则只能用递归的方法来执行来计算。课程视频通过这两个函数讲解了递归和迭代的差异是… …

本段视频也需要通过体验的方式来学习。

【视频 3.8 运用递归与迭代】

学习递归和迭代，更重要的是运用递归和迭代。

可以用递归和迭代来进行定义，将一些可不断的无限的重复组合的事情和对象表述清楚，例如怎样用递归来表述“表达式”---表达式可一层层构造，以致包含任意多个基本运算符；如何表述“祖先”---祖先也是一层层的，上一代、上两代、…、上 n 代。如何表述“树”---树也是一层层构造的…

可以用递归和迭代来进行构造，用递归的方法构造程序和算法……

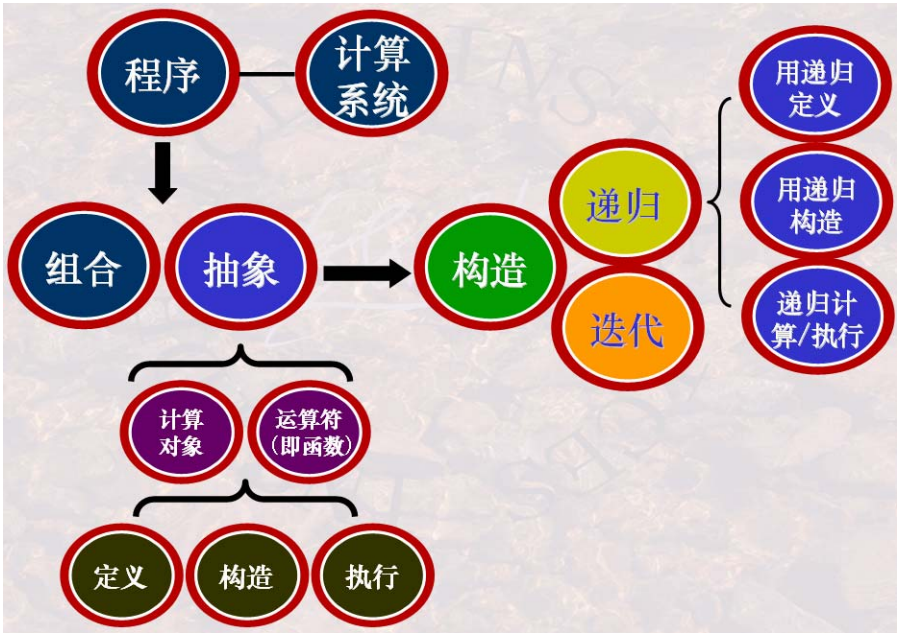
【视频 3.9 递归与迭代程序的执行】

为什么要学习递归和迭代，因为计算系统的构造本身是“递归的”，计算系统执行程序的过程是递归的，因此理解递归和迭代及其执行，是理解计算系统本身的构造、理解计算系统执行程序的关键。只有能够理解递归和迭代执行程序的过程，才能更好地运用递归和迭代构造算法和程序。本视频分别以一个递归程序和一个迭代程序为例，展示了其执行的过程、二者的差异… …

2、学习要点指南

2.1 要点一：深入理解“程序”与“递归”

本讲的目的是使你深入理解：什么是“程序”，理解程序是计算系统的基本要素，是计算系统实现千变万化复杂功能的一种手段；理解程序是构造出来的，而“构造”的重要手段是组合/复合、抽象、递归和迭代，这些概念是理解计算学科后续各种计算思维和计算技术的基础，是最基本的思维模式。下图给出了本讲的主要概念之间的关系。



计算学科的一个重要任务就是构造计算系统(包括软件系统和硬件系统)，**而构造计算系统的关键就是程序，程序是表达计算系统千变万化功能的重要手段，因此说，计算系统就是执行程序的系统。**计算系统，在制造时只需完成基本动作的制造，而在应用时则可通过对基本动作的各种组合实现任意的、复杂的动作；应用阶段的表达手段是“程序”，使用者编写了程序，计算系统便按照程序来实现任意的、复杂的动作；制造阶段，除了基本动作的制造外，还需要有一个可以执行程序机构的制造，即计算系统应是一个能够执行程序的系统。

什么是程序，程序就是 对控制系统基本动作的指令 进行组合、抽象与构造 以便使系统能够完成各种各样的功能。那什么是“复合/组合”呢？复合/组合就是将一系列动作代入到另一个动作中，进而构造出复杂的动作，是对简单元素的各种组合；最直观的例子就是：一个复杂的表达式是由一系列简单的表达式组合起来构成的。再比如，如果大家学过程序设计语言的话，一个复杂的函数是由一系列简单的函数组合起来构成的，函数之间的调用关系等体现的就是组合。那什么是抽象呢？“抽象”是对各种元素的已经构造好的组合进行命名，并将其用于更为复杂的组合构造中。比如将一系列语句命名为一个函数名，用该函数名参与复杂程序的构造，抽象是简化构造的一种手段。所以我们说程序是构造出来的，而不是编写出来的。

这里要强调一点，计算学科的“抽象”与我们平常所表达的“抽象”既有相通的一面又有些微的差别，计算学科的“抽象”是一种可掌握可操作的方法，即用名字表达一种组合，而该名字可以参与新的更为复杂的组合构造，构造中用名字进行构造，执行/计算中用被命名的组合来替代名字进行计算，这是计算学科最本质的方法。

而在构造过程中，最重要的手段就是递归与迭代，需要构造让机器不断重复执行的程序。递归是表达具有近乎无限的自相似性对象和动作的构造手段，是计算系统的典型特征。学习递归，要理解三个层面：(1)用递归进行各种对象的定义；(2)用递归进行算法和程序的构造；(3)递归定义的对象或程序的执行过程。

程序的构造手段，包括组合、抽象与递归。一个构造被代入到另一个构造中---这是组合。一个复杂的组合型构造被表达成一个名字以便参与新的构造---这是抽象。一个构造代入到另一个构造中，再被代入到另一个构造中，…，不断重复地代入到新构造中，表达这种形式的构造就是递归。

本讲内容是对第一讲“计算之树”中的“程序”“递归”思维的深化讲授。

2.2 要点二：理解并区分递归与迭代的异同

这里要注意区分两个术语。(1)递推与迭代，数学学科一般称为递推，计算学科一般称为**迭代**，而在本门课程中它们是指具有相同含义的术语。(2)递归，递归虽然包含了递推的含义，但也有不同于递推的含义。

我们先看递推和递归主要解决什么问题。二者其实都是解决如何由 h(0), h(1), …, h(n)来构造 h(n+1)的问题，这是他们的共同点。它们的差别是，递推是能够由 h(0)计算 h(1), 由 h(1)计算 h(2)，这样依次计算下去，就能计算任何一个函数 h(n+1)，即从递归基础计算起，沿相同的计算路径，总能计算到 h(n+1)；而完全的递归，却不一定，即给定任何一个 h(n+1)，由递归基础计算起，沿相同的路径却不一定能计算出 h(n+1)，因此自前向后的、沿相同路径计算是不行的。那么怎么办，它需要由 h(n+1)开始，依递归公式代入回去，直至代入到 h(0)这个递归基础，找到一条计算 h(n+1)的路径，然后再沿这条路经由前向后依次计算，最终计算出 h(n+1)。所以简单而言，**递推是自前向后计算；而完全的递归则需要由后向前代入，找到一条计算路径后，再按这条计算路径由前向后计算。**最典型的两个例子就是斐波那契数列和阿克曼函数，前者是递推的典型代表，而后者是递归的典型代表。应该说，递归包含了递推，而递推未必包含递归。递归和迭代的其他差别，视频课件中已经讲得很清楚了，请大家再仔细看看！

2.3 要点三：关于原始递归函数的理解

原始递归函数看上去很难，实际一点也不难，只是理解的时候有些“拗”，就像攀登盘旋的楼梯一样，有些晕。只要我们耐心地看一看它的定义，就可理解。

它本身也是用递归的思想来定义的：**首先**它给出了三个本原函数，即这三个函数是个起点，---一个是常数，一个是后继函数，一个是投影函数，这三个函数可以直接使用，是递归基础，仔细看看视频课件，理解并不难。**其次**，它给出了两种构造新函数的方法，一个是复合，一个是原始递归。前者解决了一个函数代入另一个函数中的问题，这实际上就是组合，只是用数学函数的形式表达出来。而后者解决了自相似性地重复构造一系列函数的问题，即依次定义 h(0), h(1),…,h(n+1)的问题，再仔细看看视频也能理解。

递归函数很重要吗，有什么作用吗？它确实很重要，在视频课件中可以看出它是表达重复性“组合”的一种手段。比如，我要将若干个表达式，按照一定次序装配起来，构造一个复杂表达式，怎样自动装配呢？将来大家在研究体系结构问题、算法构造问题等都将用到递归这种思想。

不知道大家注意到没有，视频课件的两个例子，在重复性构造新函数时，都是相同的形式，即： h(S(n), x) = g (h(n), n, x) ，这个式子说明将 h(n)和 n 以及相关的一些参量代入到 g 中便可构造出 h(n+1)函数的形式，为什么会相同呢？它是否是一种程序执行机构的表达呢？ f 和 g 是两个输入，输入是不同的，我们可以给出任何的 f 和 g，也就是任何一个程序。但构造新函数的函数 h(S(n), x)却是相同的，即程序执行机构是相同的，是不是这样呢？

理解递归函数，不能仅仅是从概念上理解，我们要动手体验一下，按照视频课件给定的例子，体验一下 h(0), h(1), h(2),…, h(n)的计算过程，你在计算这些函数时一定会体验到给定两个基本函数，为什么就能定义出这么多新的函数呢？而这些新函数又具有了一种什么规律呢？计算思维是体验并悟出来的！

所以，有人说计算思维是一种递归思维，确实是这样的。我们后面要讲的图灵机也体现了这样的递归的思想。递归真的很重要，计算学科的灵魂就是算法，而算法理解的基础就是递归。如果递归理解得不好，那很多的算法也是难于理解的。

3、常见问题

3.1 为什么要理解递归，递归应理解到什么程度？

递归是计算学科最基本的表达手段---将一种具有自相似性、重复的无限个对象或动作表达出来的手段；递归是计算学科最基本的构造手段---构造算法、构造程序，一种自身调用自身、高阶调用低阶的构造手段。计算机器/计算系统一般都是“递归”地执行重复的动作，以完成大量的计算任务。因此，能否理解计算机器/计算系统的关键是能否理解“递归”，能否理解复杂算法和程序的关键也是能否理解“递归”。若大家要进一步学习“计算理论”“编译”等更深入的课程，递归的理解是最最基本的内容。若大家要成为一个高水平的程序员，能够编写类似操作系统一样的

系统程序，则不理解递归也是不能够达到的。

关于“递归”要理解到什么程度的问题？我们说要理解递归的三个层面的内容，即：用递归进行定义(表达具有自相似性的无限的对象或动作)，用递归进行构造(构造算法和程序)，理解(计算系统/计算机器)递归地执行(算法和程序的)过程。

3.2 这一讲有些学员感觉很难，感觉很抽象？感觉难以把握，怎么解决？

本讲内容涉及到对程序和递归本质的认识，通常在这个层面的内容大家都会感觉很难，感觉很抽象，这很正常。但这又是学习计算思维必须跨过的一道门槛。比如，从递归的作用角度，我们后面将要介绍的图灵机—为什么一个五元组集合能表达任何一个程序呢？为什么简单的计算机语言，可以构造出千变万化的程序呢？为什么写了一个程序，却可以处理无穷无尽的数据呢？这些都有递归思想的作用在里面。所以程序和递归的理解很重要！

关于怎么解决，给出一些建议如下：一是你可关注本讲的模拟练习题，通过多做几遍这些模拟练习题，来体验递归的思想。二是你可以网上搜索一些递归函数，自我进行训练；三是有很多算法里面都有递归的思想--比如汉诺塔的求解算法(在《大学计算机-计算与信息素养》教材的第 6 章中有此内容的介绍)、再比如用递归思想进行排序（待排序元素分成两个子集合，比某一元素小的在左子集合中，比其大的在右子集合中；再进一步对左子集合和右子集合的元素再应用前述思想进行处理。直到所有的都排好序）。四是借鉴视频《运用递归和迭代》，训练自己去定义某种语言，比如什么是 XML 语言，什么是 Python 语言，能否用递归的方法对它们进行严格的定义呢？---不一定要定义所有内容，仅定义你所关注的内容即可。五是多看几遍视频，为什么要多看几遍，要强化对术语的接受度，要反复听、多思考，自然就可理解。计算思维是一种思维，如果看一遍你就完全明白了，那还是计算思维吗？计算思维一定是需要你仔细揣摩、仔细思考、仔细回味的东西，因此学完一门课后，是否有余味，是否引起你更多的思考，这就是这门课是否有深度的一个标志。