如果你已经理解了消息是如何广播并接收处理的，下面我们就开始学习能够管理复杂大型程序的结构化程序设计。

## 将大型程序分而治之

我们之前看到的脚本都比较短小，而且功能简单。随着学习的深入，总有一天你会写出更长更复杂的甚至上百块积木的脚本，这时想要理解和维护程序就非常困难。

20 世纪 60 年代中期出现了一种叫作结构化程序设计的方法，它能简化计算机程序的编写、理解和维护。采用这种方式编写的程序不是用一段很长的代码实现所有的功能，而是将所有的功能划分为许多实现部分功能的单元。

我们用这种方式来思考一下怎么制作蛋糕。制作蛋糕通常是按照食谱上的配方执行多个必需的步骤，而不是用一个步骤直接完成。食谱可能包含如下步骤：第一步，混合 4 个鸡蛋、60 克面粉和 1 杯水；第二步，将混合物放入锅内；第三步，把锅放入烤箱内；第四步，用 177℃烘烤 1 小时。从结构化程序设计的角度讲，面包的制作流程被食谱分解为不同的逻辑步骤。

与之类似，当你解决某个计算机问题时，这种思维方式能把问题分解为许多易于管理的部分，同时还能帮助你梳理整个程序的脉络和逻辑，利于维护各个部分之间的关系。

图 4-8 展示了一段很长的脚本，其整体功能是在舞台上绘制一个图形。但是你能一眼就直接看出最终绘制的图形吗？因此，我们需要把脚本分解到多个更小的逻辑块中。例如，图 4-8 中左侧的前六块积木初始化角色的相关信息，第一个**重复执行**绘制正方形，第二个绘制三角形，以此类推。使用这种结构化编程的方式可以聚集功能类似的积木，从而形成过程。

当创建了许多过程后，我们便能以特定的顺序使用它们解决编程问题。图 4-8 右侧的脚本展示了多个过程卡合在一起完成了与原脚本相同的绘图结果。你是否也觉得使用过程的脚本（右侧）比最初的脚本（左侧）更加模块化且易于理解呢？

过程还可以避免多次出现相同的脚本。如果脚本中多次使用了一系列相同的积木，建议你将其替换为过程。这种方式可以避免脚本的复制和粘贴，专业地讲叫作代码复用，就像图 4-8 中的过程