

# 介绍

2017 年 9 月 1 日

## 1 目标

- 回顾计算机科学的思想, 提高编程和解决问题的能力。
- 理解抽象化以及它在解决问题过程中发挥的作用
- 理解和实现抽象数据类型的概念
- 回顾 Python 编程语言

## 2 快速开始

从第一台通过接入网线和交换机来传递人的指令的计算机开始, 我们编程思考的方式发生了许多变化。与社会的许多方面一样, 计算技术的变化为计算机科学家提供了越来越多的工具和平台来实践他们的工艺。计算机的快速发展诸如更快的处理器, 高速网络和大的存储器容量已经让计算机科学家陷入高度复杂螺旋中。在所有这些快速演变中, 一些基本原则保持不变。计算机科学关注用计算机来解决问题。毫无疑问你花了相当多的时间学习解决问题的基础知识, 以此希望有足够的能把问题弄清楚并想出解决方案。你还发现编写代码通常很困难。问题的复杂性和解决方案的相应复杂性往往会掩盖与解决问题过程相关的基本思想。本章着重介绍了其他两个重要的部分。首先回顾了计算机科学与算法和研究数据结构所必须适应的框架, 特别是我们需要研究这些主题的原因, 以及如何理解这些主题有助于我们更好的解决问题。第二, 我们回顾 Python 编程语言。虽然我们不提供详尽的参考, 我们将在其余章节中给出基本数据结构的示例和解释。

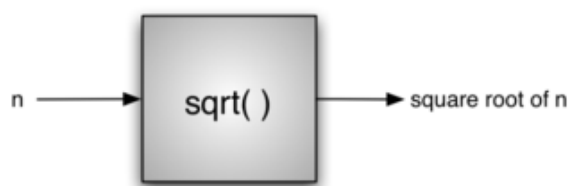
## 3 什么是计算机科学

计算机科学往往难以定义。这可能是由于在名称中不幸使用了“计算机”一词。正如你可能知道的, 计算机科学不仅仅是计算机的研究。虽然计算机作为一个工具在学科中发挥重要的支持作用, 但它们只是工具。计算机科学是对问题, 解决问题以及解决问题过程中产生的解决方案的研究。给定一个问题, 计算机科学家的目标是开发一个算法, 一系列的指令列表, 用于解决可能出现的问题的任何实例。算法遵循它有限的过程就可以解决问题。计算机科学可以被认为是研究算法。但是, 我们必须谨慎地包括一些事实, 即一些问题可能没有解决方案。虽然证明这种说法正确性超出了本文的范围, 但一些问题不能解决的事实对于那些研究计算机科学的人是很重要的。所以我们可以这么定义计算机科学, 是研究能被解决的问题的方案和不能被解决问题的科学。通常我们会说这个问题是可计算的, 当在描述问题和解决方案时。如果存在一个算法解决这个问题, 那么问题是可计算的。计算机科学的另一个定义是说, 计算机科学是研究那些可计算和不可计算的问题, 研究是不是存在一种算法来解决它。你会注意到, “电脑”一词根本没有出现。解决方案是独立于机器而言的。计算机科学, 因为它涉及问题解决过程本身, 也是抽象的研究。抽象使我们能够以分离所谓的逻辑和物理角度的方式来观察问题和解决方案。基本思想跟我们常见

的例子一样。假设你可能已经开车上学或上班。作为司机，汽车的用户。你为了让汽车载你到目的地，你会和汽车有些互动。进入汽车，插入钥匙，点火，换挡，制动，加速和转向。从抽象的角度，我们可以说你看到是汽车的逻辑视角。你正在使用汽车设计师提供的功能，将你从一个地方运输到另一个位置。这些功能有时也被称为接口。另一方面，修理汽车的技工有一个截然不同的视角。他不仅知道如何开车，还必须知道所有必要的细节，使我们认为理所当然的功能运行起来。他需要了解发动机是如何工作的，变速箱如何变速，温度是如何控制的等等。这被称为物理视角，细节发生在“引擎盖下”。当我们使用电脑时也会发生同样的情况。大多数人使用计算机写文档，发送和接收电子邮件，上网冲浪，播放音乐，存储图像和玩游戏，而不知道让这些应用程序工作的细节。他们从逻辑或用户角度看计算机。计算机科学家，程序员，技术支持人员和系统管理员看计算机的角度截然不同。他们必须知道操作系统如何工作的细节，如何配置网络协议，以及如何编写控制功能的各种脚本。他们必须能够控制底层的细节。这两个示例的共同点是用户态的抽象，有时也称为客户端，不需要知道细节，只要用户知道接口的工作方式。这个接口是用户与底层沟通的方式。作为抽象的另一个例子，Python 数学模块。一旦我们导入模块，我们可以执行计算

```
>>> import math
>>> math.sqrt(16)
4.0
>>>
```

这是一个程序抽象的例子。我们不一定知道如何计算平方根，但我们知道函数是什么以及如何使用它。如果我们正确地执行导入，我们可以假设该函数将为我们提供正确的结果。我们知道有人实现了平方根问题的解决方案，但我们只需要知道如何使用它。这有时被称为“黑盒子”视图。我们简单地描述下接口：函数的名称，需要什么（参数），以及将返回什么。细节隐藏在里面



## 4 什么是编程

编程是将算法编码为符号，编程语言的过程，以使得其可以由计算机执行。虽然有许多编程语言和不同类型的计算机存在，第一步是需要有解决方案。没有算法就没有程序。

计算机科学不是研究编程。然而，编程是计算机科学家的重要能力。编程通常是我们为解决方案创建的表现形式。因此，这种语言表现形式和创造它的过程成为该学科的基本部分。

算法描述了依据问题实例数据所产生的解决方案和产生预期结果所需的一套步骤。编程语言必须提供一种表示方法来表示过程和数据。为此，它提供了控制结构和数据类型。

控制结构允许以方便而明确的方式表示算法步骤。至少，算法需要执行顺序处理，决策选择和重复控制迭代。只要语言提供这些基本语句，它就可以用于算法表示。

计算机中的所有数据项都以二进制形式表示。为了赋给这些字符串含义，我们需要有数据类型。数据类型提供了对这个二进制数据的解释，以便我们能够根据解决的问题思考数据。这些底层的内置数据类型（有时称为原始数据类型）为算法开发提供了基础。

例如，大多数编程语言为整数提供数据类型。内存中的二进制数据可以解释为整数，并且能给予一个我们通常与整数（例如 23,654 和 -19）相关联的含义。此外，数据类型还提供数据项参与的操作的描述。对于整数，诸如加法，减法和乘法的操作是常见的。我们期望数值类型的数据可以参与这些算术运算。通

常我们遇到的困难是问题及其解决方案非常复杂。这些简单的，语言提供的结构和数据类型虽然足以表示复杂的解决方案，但通常在我们处理问题的过程中处于不利地位。我们需要一些方法控制这种复杂性，并能给我们提供更好的解决方案。

## 5 为什么要学习数据结构和抽象数据类型

为了管理问题的复杂性和解决问题的过程，计算机科学家使用抽象使他们能够专注于“大局”而不会迷失在细节中。通过创建问题域的模型，我们能够利用更好和更有效的问题解决过程。这些模型允许我们以更加一致的方式描述我们的算法将要处理的数据。之前，我们将过程抽象称为隐藏特定函数的细节的过程，以允许用户或客户端在高层查看它。我们现在将注意力转向类似的思想，即数据抽象的思想。抽象数据类型（有时缩写为 ADT）是对我们如何查看数据和允许的操作的逻辑描述，而不用考虑如何实现它们。这意味着我们只关心数据表示什么，而不关心它最终将如何构造。通过提供这种级别的抽象，我们围绕数据创建一个封装。通过封装实现细节，我们将它们从用户的视图中隐藏。这称为信息隐藏。



Figure 2 展示了抽象数据类型是什么以及如何操作。用户与接口交互，使用抽象数据类型指定的操作。抽象数据类型是用户与之交互的 shell。实现隐藏在更深的底层。用户不关心实现的细节。

抽象数据类型（通常称为数据结构）的实现将要求我们使用一些程序构建和原始数据类型的集合来提供数据的物理视图。正如我们前面讨论的，这两个视角的分离将允许我们将问题定义复杂的数据模型，而不给出关于模型如何实际构建的细节。这提供了独立于实现的数据视图。由于通常有许多不同的方法来实现抽象数据类型，所以这种实现独立性允许程序员在不改变数据的用户与其交互的方式的情况下切换实现的细节。用户可以继续专注于解决问题的过程。

## 6 为什么要学习算法

计算机科学家经常通过经验学习。我们通过看别人解决问题和自己解决问题来学习。接触不同的问题解决技术，看不同的算法设计有助于我们承担下一个具有挑战性的问题。通过思考许多不同的算法，我们可以开始开发模式识别，以便下一次出现类似的问题时，我们能够更好地解决它。算法通常彼此完全不同。考虑前面看到的  $\text{sqrt}$  的例子。完全可能的是，存在许多不同的方式来实现细节以计算平方根函数。一种算法可以使用比另一种更少的资源。一个算法可能需要 10 倍的时间来返回结果。我们想要一些方法来比较这两个解决方案。即使他们都工作，一个可能比另一个“更好”。我们建议使用一个更高效，或者一个只是工作更快或使用更少的内存的算法。当我们研究算法时，我们可以学习分析技术，允许我们仅仅根据自己的特征而不是用于实现它们的程序或计算机的特征来比较和对比解决方案。在最坏的情况下，我们可能有一个难以处理的问题，这意味着没有算法可以在实际的时间量内解决问题。重要的是能够区分具有解决方案的那些问题，不具有解决方案的那些问题，以及存在解决方案但需要太多时间或其他资源来合理工作的那些问题。经常需要权衡，我们需要做决定。作为计算机科学家，除了我们解决问题的能力，我们还需要了解解决方案评估技术。最后，通常有很多方法来解决问题。找到一个解决方案，我们将一遍又一遍比较，然后决定它是否是一个好的方案。