教育部大学计算机课程改革项目成果
工业和信息化部所属高校联盟推荐教材

大学计算机实践教程
——面向计算思维能力培养

王立松　潘梅园　朱敏　编著


示例视频

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

教育部大学计算机课程改革项目成果
工业和信息化部所属高校联盟推荐教材

大学计算机实践教程

——面向计算思维能力培养

王立松 潘梅园 朱敏 编著

電子工業出版社
Publishing House of Electronics Industry

内 容 简 介

本书是教育部大学计算机课程改革项目成果，是一本面向计算思维能力培养的大学计算机实践教程，力图从计算机问题求解的角度，引导学生利用可视化的程序设计工具进行问题描述和求解。书中引用和创作了丰富的实例，通过实例逐步介绍计算机问题求解的一般方法，通过设计对应的实验，使得学生在实践中强化计算思维，提高计算思维能力。本书内容结构上具体分为三部分：第一部分为 Raptor 程序设计基础；第二部分为问题求解实例；第三部分为问题求解实践内容，包括基本实验和综合设计实验。

本书配套有丰富的教学资源，包括：PPT、电子素材、示例演示视频等。除此以外，本书特别添加了二维码技术，读者可以通过移动终端扫描本书封面上的二维码来观看相应示例的演示视频。

本书适用于大学低年级学生，可作为大学计算机实践课程的教材，也可作为理解计算思维、提高问题求解能力的参考用书，或者作为软件开发人员或计算机爱好者的自学用书。

图书在版编目（CIP）数据

大学计算机实践教程：面向计算思维能力培养/王立松，潘梅园，朱敏编著. —北京：电子工业出版社，2014.9

ISBN 978-7-121-24248-9

Ⅰ. ①大… Ⅱ. ①王… ②潘… ③朱… Ⅲ. ①电子计算机—高等学校—教材 Ⅳ. ①TP3

中国版本图书馆 CIP 数据核字（2014）第 204023 号

策划编辑：任欢欢
责任编辑：章海涛　　文字编辑：任欢欢
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1 092　1/16　印张：11.25　字数：270 千字
版　　次：2014 年 9 月第 1 版
印　　次：2014 年 9 月第 1 次印刷
定　　价：28.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

近年来，大学计算机课程的教学改革探索和实践表明，大学计算机课程的教学应该以培养计算思维能力为核心任务。在具体教学实施过程中，如何培养计算思维能力就成为大学计算机教学的中心问题。通过对这个问题的深入思考和多年的教学经历，我们认为有两个方面或途径来培养计算思维能力：一是在课堂教学上，把计算（机）的知识放在思维层面进行讲解，学生通过思考“这些知识是如何形成的”来贯通知识，计算思维能力也在这种知识的贯通过程中得到提高；二是围绕计算思维的“应用”，主要基于计算机的问题求解，这样计算思维就必然涉及如何构建计算环境以及如何进行问题求解。大学低年级的学生，主要通过简单的工具进行问题描述，并能在计算机上执行这一过程“体会和实践”计算思维，从而培养计算思维能力，为未来进一步学习诸如高级语言程序设计等课程打下坚实基础。重要的是，学生能结合自身专业，利用计算思维求解问题，甚至可以验证问题求解方法的有效性与正确性。

本教材的目的在于引导学生进行面向计算思维能力培养的实践，切入点是计算思维指导下的计算机问题求解。鉴于大学低年级学生的计算机相关知识水平可能不够，教材采用浅显的语言，简单介绍了一些必要的知识。为了可以在计算机上进行实践，本书选用了非常简便的可视化程序设计工具 Raptor 作为实践工具，给出了工具的基本要素和常用技巧及其应用。有了这些内容作为基础，教材又给出了基本的问题处理策略和问题求解实例。最后设计了一系列精心挑选的问题求解实例和问题求解实践题目，供读者进行学习、参考和实践。希望通过本教材的引导，读者可以有一个深层次的“入门”，在实践中提高计算思维能力，也为后续课程的学习打下坚实基础。

本教材适合于各类专业的大学生，建议在大学一年级第一学期开设。考虑到教学进度和学生接受程度，总学时安排 30 学时为宜。由于是实践教程，建议在实验室讲授，边学边练边思考。

本教材由王立松、潘梅园和朱敏共同创作和编写，王立松负责统稿。其中，王立松编写第 1 章和第 8 章，潘梅园编写第 4、5、6、9、10 章，朱敏编写第 2、3、7 章。南京航空航天大学长期从事大学计算机教学的一线教师对本教材的书稿进行了讨论，并提出了很好的修改建议。陈龙等研究生参与了部分实例的制作工作。

本书在成稿过程中得到很多专家教授的指点和帮助，哈尔滨工业大学的战德臣教授给予了非常多的建议，南京航空航天大学的陈兵教授细致审阅了稿件。在此对他们表示衷心的感谢。

感谢南京航空航天大学教务处、计算机科学与技术学院及电子工业出版社对本书出版所给予的大力支持。在此对为本书出版做出贡献的所有人员一并表示衷心的感谢。

面向计算思维能力培养的大学计算机实践是一门发展中的课程，由于时间仓促和作者的水平限制等因素，教材中的内容难免有不完善之处，敬请广大读者谅解，并诚挚地欢迎读者提出宝贵建议。

作 者

2014 年 8 月

目　录

第1章

概　述

1.1　培养计算思维能力的重要性

计算思维是指计算机、软件等与计算相关的学科中科学家和工程技术人员的思维模式。美国卡内基·梅隆大学周以真（Jeannette M. Wing）教授认为，计算思维（Computational Thinking）是运用计算机科学的基础概念去求解问题、设计系统和理解人类的行为，计算思维的本质是抽象（Abstraction）和自动化（Automation）。如同所有人都具备是非判断、文字读写和进行算术运算一样，计算思维也是一种本质的、所有人都必须具备的思维能力。有学者认为，计算思维被归纳、提出，可能是近十年来计算科学和计算机学科中最具有基础性的、长期性的重要的学术思想。国内很多学者也对计算思维进行了深入的研究，如陈国良院士、李廉教授等，将计算思维看成是除理论思维、实验思维外的第三大思维。理论思维是以推理和演绎为特征的“逻辑思维”，用假设/预言——推理和证明等理论手段研究社会/自然现象及规律；实验思维是以观察和总结为特征的“实证思维”，用实验—观察—归纳等实验手段研究社会/自然现象及规律；而计算思维则是以设计和构造为特征的“构造思维”，是以计算手段研究社会/自然现象及规律。随着社会/自然探索内容的深度化和广度化，传统的理论手段和实验手段已经受到很大的限制，实验产生了大量数据其结果是很难通过观察获得的，因此不可避免地需要利用计算手段来实现理论与实验的协同创新。战德臣教授等，为了概括计算学科中所体现的重要的计算思维，提出了一种多维度观察计算思维的框架——计算之树。

事实上，更深层地说，计算思维的核心是基于计算模型（环境）和约束的问题求解。计算机学科是研究计算模型、计算系统的设计以及如何有效地利用计算系统进行信息处理、实现工程应用的学科，涉及基本模型的研究、软件/硬件系统的设计以及面向应用的技术研究与工程方法研究。虽然计算机学科研究涉及面广，但其共同特征还是基于特定计算环境的问题求解。比如，计算机科学基础理论研究实际上是基于抽象级环境（如图灵机）的问题求解，计算机硬件体系的设计与研究则是一种指令级的问题求解，程序设计是基于语言级的问题求解活动，系统软件设计与应用软件设计则是一种系统级的问题求解。因此，可以认为，计算思维的本质特征是基于不同层次计算环境的问题求解。而不同层次计算环境的问题求解行为也反映了计算机学科的三种不同形态：科学、技术与工程。

如果说计算思维的本质特征是基于计算模型（环境）和约束的问题求解，那么计算思维就必然涉及怎么构建计算环境以及如何进行问题求解，更进一步地包括怎么验证问题求解方法的有效性与正确性。因此，计算思维的核心方法就是“构造”，不但构造计算环境，而且构造基于计算环境的问题求解过程，以及构造对问题求解过程的验证方法。可以称这三类构造为对象构造、过程改造、验证构造。

鉴于计算思维的重要性，人们认识到，大学计算机的教学和学习核心任务就是培养计算思维能力。

1.2 为什么培养计算思维需要有实践

计算思维的重要意义在于指导如何利用计算（机）的理论、方法、技术、系统及其工具进行问题求解。在问题求解实践过程中，计算思维得到进一步的强化，计算思维能力得到不断增强；反过来，这种增强了的计算思维能力又可以提升实践者问题求解的能力。所以，我们学习和理解计算思维，培养计算思维能力，需要配合必要的具体应用和实践，使得“抽象”的思维活动可以具体“显现”出来。

计算思维本身就是在计算（机）科学和技术的研究和实践中不断被总结和发现的，如果脱离了实践，则有可能使得计算思维成为一种裹足不前的空洞“思维”，而培养计算思维能力则有可能成为镜花水月，所以从计算思维自身的发展来说，计算思维也是离不开实践的。

培养一个人的计算思维能力最终还是要促进人的实践能力，而实践能力却是需要在实践活动中不断思考和演练才能逐步提高，这也是创新能力培养的一般过程。总的来说，计算思维的培养和实践能力的培养是一体两面、相辅相成的，计算思维在计算实践活动中被总结和反复应用，而各种借助于计算工具进行的实践又在反复地应用着计算思维，在计算思维指导下进行创新实践，使得人类的计算思维能力越来越强，也使得利用计算（机）科学和技术进行问题求解的能力越来越强。

1.3 如何进行面向计算思维培养的实践

本书中，面向计算思维培养的实践试图体现抽象、构造和自动化这些计算思维的本质特征，以培养问题求解，系统设计和人类行为理解等方面的能力为目标，从问题

的描述、计算（机）原理和过程的展示、算法和程序的设计、系统的设计和实现、模型的分析和验证等方面进行具体的实验项目设计，这些实验项目可以让读者循序渐进地在实践环节中得到训练，配合面向计算思维能力培养的教学，培养学生的实践能力甚至创新思维能力。

虽然计算思维可以脱离具体的计算机器、系统和工具，就如使用不插电的计算机那样，但是，本书所说的面向计算思维培养的实践是要借助计算机为工具的，所说的“问题求解”是指利用计算机系统进行问题求解，也就是说，我们在对问题进行分析和理解时，使用了计算机问题求解的基本步骤和方法，通过计算机问题求解的方法找出（或者说是构造出）解决问题的具体有效的办法，再利用计算机最终把问题具体“解”出来，并能分析求解这个问题付出了怎样的代价，进一步验证某些期望的性质是否得到满足。

要达到上述目的，就需要了解到底利用计算机进行问题求解需要我们做什么。正如一个人与另外一个人合作交流并希望对方帮忙解决问题一样，首先需要把自己的问题分析清楚，然后用对方能听得懂的语言无二义性地表达出来（还可以告诉对方如何完成这个任务），所以我们碰到的第一个问题就是如何用一个语言来描述问题，在计算机科学和技术中，这是计算机语言和程序设计需要解决的问题。其次，为了表达问题中涉及的处理对象（亦即数据对象）及其对象之间的联系等信息，计算机科学家为此建立了描述数据对象和对象之间关系的一种结构，即所谓的数据结构。最后，在解决问题的方法上，人们通过不断地研究、实践和总结，形成了很多解决问题的一般性的方法和步骤，这些方法和步骤往往基于某种数据结构，即所谓的算法。当然，要完成一个问题的最终求解，还需要有一定能力的计算机，所以还要了解计算机硬件系统和软件系统的基本构成、工作原理，以及其中蕴含的思维方式。

从上面的讨论可以看出，要让计算机为我们解决问题，其实首先是人要有解决问题的思路和方法，然后以计算机能接受的方式“告诉”计算机，才能利用计算机得到最终想要的“答案”。为此，为了后续章节的学习，围绕计算机问题求解，本章将简要介绍一些必要的概念和术语，它们分别是：程序、程序设计、程序设计语言、算法、数据结构，以及本书使用的主要工具——Raptor，并适当地用 Raptor 来说明这些概念和术语的含义。

1.4 程序、程序设计和程序设计语言

要培养计算思维，必然需要表达计算思维，表达计算思维就需要用到语言，因为最终要把一个想法或者问题的解决方案在计算机上实现，就需要了解和掌握计算机中的语言，即程序设计语言，这里需要了解什么是程序、程序设计是做什么的、程序设计语言是什么这三个问题。

1.4.1 程序及其基本要素

我们日常生活中打算完成一项任务的时候，通常会事先拟定一系列有序的具体步骤，具体执行的时候就可以按照这些具体步骤循序渐进地完成此项任务。这些完成目

标任务的一系列有序步骤就是程序，只是本书中的“程序”指的是计算机程序。你会发现，生活中在描述一项任务的具体步骤时，一定会需要用某种语言（不一定是自然语言，可能是其他的文字或者符号）来描述。同样，让计算机完成任务的步骤序列即计算机程序也是需要用语言来描述的，只不过是计算机能“认识”的语言，计算机能“认识”的语言最基本的形式是机器能执行的指令。因此，我们可以把程序简单地定义为：由基本动作指令构造的若干指令的一个组合或一个执行序列，用以实现千变万化的复杂动作，或者说，程序是为解决一个信息处理任务而预先编制的工作执行方案，是由一串 CPU 能够执行的基本指令组成的序列，每一条指令规定了计算机应进行什么操作（如加、减、乘、判断等）及操作需要的有关数据。例如，从存储器读一个数送到运算器就是一条指令，从存储器读出一个数并与运算器中原有的数相加也是一条指令。

这里需要特别强调的是，人们在从事计算机程序相关的工作中使用的是一种称为“程序”思维的思维模式，即一个复杂系统是怎样实现的？系统可被认为是由基本动作（注：基本动作是容易实现的）以及基本动作的各种组合所构成（注：多变的、复杂的动作可由基本动作的各种组合来实现）。因此，实现一个系统仅需实现这些基本动作以及实现一个控制基本动作组合与执行次序的机构。对基本动作的控制就是指令，指令的各种组合及其次序就是程序。系统可以按照“程序”控制“基本动作”的执行以实现复杂的功能。计算机或计算系统就是能够执行各种程序的机器或系统，指令与程序的思维也是最重要的一种计算思维。

对于计算机的程序而言，它有自己的基本构成要素，通常由两种基本要素组成：一是对数据对象的运算和操作，二是程序的控制结构。

1. 程序中对数据的运算和操作

每个程序实际上是按解题要求从环境能进行的所有操作中选择合适的操作所组成的一组指令序列。因此，计算机程序就是计算机能处理的操作所组成的指令序列。

通常，计算机可以执行的基本操作是以指令的形式描述的。一个计算机系统能执行的所有指令集合称为该计算机系统的指令系统。计算机程序就是按解题要求从计算机指令系统中选择合适的指令所组成的指令序列。在一般的计算机系统中，基本的运算和操作有以下 4 种。

① 算术运算：主要包括加、减、乘、除等运算。

② 逻辑运算：主要包括“与”、“或”、“非”等运算。

③ 关系运算：主要包括“大于”、“小于”、“等于”、“不等于”等运算。

④ 数据传输：主要包括赋值、输入、输出等操作。

前面提到，高级程序设计语言也可以作为程序的一种描述，但由于在高级语言描述程序时通常要考虑很多与方法和分析无关的细节问题（如语法规则），因此，在设计程序之初，通常并不直接用高级语言来描述程序，而是用其他描述工具（如流程图、伪代码，甚至用自然语言）来描述程序。但不管用哪种工具来描述程序，程序设计一般都应考虑按解题要求从上述 4 种基本操作中选择合适的操作，组成解题的操作序列。程序的主要特征着重于程序的动态执行，这使得它有别于传统的着重于静态描述或按演绎方式求解问题的过程。传统的演绎数学是以公理系统为基础的，问题的求解过程

是通过有限次推演来完成的，每次推演都将对问题做进一步的描述，如此不断推演，直到直接将解描述出来为止。而计算机程序则是用一些最基本的操作，通过对已知条件一步一步地加工和变换，从而实现解题目标。

2. 程序的控制结构

一个程序的功能不仅取决于所选用的操作，还与各操作之间的执行顺序有关。程序中各操作之间的执行顺序称为程序的控制结构。

程序的控制结构给出了程序的基本框架，不但决定了程序中各操作的执行顺序，而且直接反映了程序的设计是否符合结构化原则。描述程序的工具通常有传统流程图、伪代码、程序设计语言等。一个程序一般都可以用顺序、选择、循环这 3 种基本控制结构组合而成。

例 1-1：有黑和蓝两个墨水瓶，但错把黑墨水装在了蓝墨水瓶子里，而蓝墨水错装在黑墨水瓶子里，要求将其互换。

解：这是一个非数值运算问题。因为两个瓶子的墨水不能直接交换，所以，解决这类问题的关键是需要借助第三个墨水瓶。设第三个墨水瓶为黄瓶，其交换步骤如下：

① 将黑瓶中的蓝墨水装入黄瓶中。

② 将蓝瓶中的黑墨水装入黑瓶中。

③ 将黄瓶中的蓝墨水装入蓝瓶中。

④ 交换结束。

例 1-2：计算函数 $f(x)$ 的值。函数 $f(x)$ 为：

$$f(x)=\begin{cases} bx+a & x \leqslant a \\ ax+b & x>a \end{cases}, \text{其中 } a,b \text{ 为常数}$$

解：本题是一个数值运算问题。其中 $f(x)$ 代表要计算的函数值，有两个不同的表达式，根据 x 的取值决定采用哪一个算式。根据计算机具有逻辑判断的基本功能，用普通语言描述如下：

① 将 a，b 和 x 的值输入到计算机。

② 判断 x 是否小于等于 a，如果条件成立，执行第③步，否则执行第④步。

③ 按表达式 $bx+a$，计算出结果并存放到 f 中，然后执行第⑤步。

④ 按表达式 $ax+b$，计算出结果并存放到 f 中，然后执行第⑤步。

⑤ 输出 f 的值。

⑥ 程序结束。

由上述两个简单的例子可以看出，一个程序由若干操作步骤构成，并且任何简单或复杂的程序都是由基本功能操作和控制结构这两个要素组成的。程序的控制结构决定了程序的执行顺序。

1.4.2 程序设计

前面解释了什么是程序和程序的组成要素，知道了计算机是执行程序来完成指定工作的，那么如何得到让计算机完成我们想要完成的任务的程序呢？显然，这个程序还是要由人来把它“做”出来，也就是人要先构造出计算机能一步一步执行的指令序列，即程序，这个构造过程就是程序设计，所以说，“构造”是计算思维的一个本质特

征就是这个原因。通过一定的手段和技术，把一个计算机能执行的程序构造出来，所以程序设计就是给出解决特定问题的程序的过程，是软件构造活动中的重要组成部分。程序设计往往以某种程序设计语言为工具，给出这种语言描述的程序。程序设计过程应当包括分析、设计、编码、测试、排错等阶段。专业的程序设计人员常被称为程序员。

在某种意义上，程序设计的出现甚至早于电子计算机的出现。英国著名诗人拜伦的女儿爱达·勒芙蕾丝曾设计了巴贝奇分析机上计算伯努利数的一个程序。她甚至创造了循环和子程序的概念。由于她在程序设计上的开创性工作，爱达·勒芙蕾丝被称为世界上第一位程序员。

任何设计活动都是在各种约束条件和相互矛盾的需求之间寻求一种平衡，程序设计也不例外。在计算机技术发展的早期，由于机器资源比较昂贵，程序的时间和空间代价往往是设计关心的主要因素；随着硬件技术的飞速发展和软件规模的日益庞大，程序的结构、可维护性、复用性、可扩展性等因素日益重要。

另一方面，在计算机技术发展的早期，软件构造活动主要就是程序设计活动。但随着软件技术的发展，软件系统越来越复杂，逐渐分化出许多专用的软件系统，如操作系统、数据库系统、应用服务器，而且这些专用的软件系统越来越普遍地成为计算环境的一部分。在这种情况下，软件构造活动的内容越来越丰富，不再只是纯粹的程序设计，还包括数据库设计、用户界面设计、接口设计、通信协议设计和复杂的系统配置过程。

由于现代人们的工作、学习和生活几乎很难离得开计算机，例如，每天都在使用计算机帮我们完成一定的工作，所以程序设计自然就成为大多数人需要了解并掌握的基本概念和技能了。

1.4.3 程序设计语言

以上在介绍程序和程序设计中都提到，程序是用某种语言来描述的，程序设计也是要用到某种语言来设计程序，我们不妨将用于说明程序和进行程序设计的语言称为程序设计语言（Programming Language）。图 1-1 形象地描述了程序设计语言的作用，即人类要想让计算机解决问题，必须用程序设计语言和计算机进行沟通。

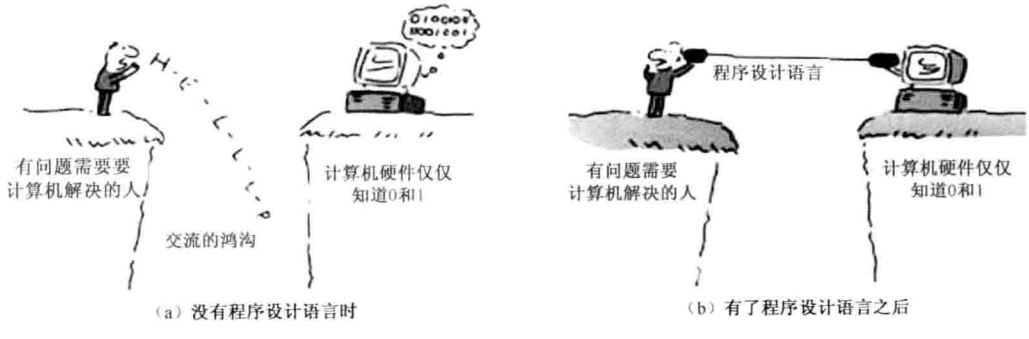

图 1-1 程序设计语言的作用

程序设计语言同其他语言一样，是人与机器之间、机器与机器之间沟通的媒介；类似地，它的基础也是一组记号和一组规则。根据规则，由记号构成的记号串的总体就是语言（读者可以仔细类比你自己说的自然语言来理解这一点）。在程序设计语言中，这些记号串就是程序。

我们不妨来看一个例子，现在想要完成一项任务："求数字1055减去383与545的和的结果。"这里，我们首先获得了第一种表达这个任务的方式，就是自然语言的表达方式。这句话是通过符合自然语言语法规则组成的一个符号串，对于这个符号串，懂中文并且学过小学算术的人都可以"执行"，但没有学过中文的人看不懂，无法"执行"。当然，此任务机器是不能执行的。

为了让没有学过中文的人能理解这个任务，我们可以采用通用的数学语言，即："1055 -(383 +545)"，这样全世界学过整数算术运算的人都能理解和"执行"了。显然，这个符号串是符合数学运算规则的符号串，也可以看成一个程序，但机器还是不能执行。

为了让机器能理解和执行这个任务，就需要用机器能认识的语言来表达这个任务，图1-2中的符号串就是用机器语言表达的"求1055 -(383 +545)的结果"这个任务。

```
B8  7F  01
BB  21  02
03  D8
B8  1F  04
2B  C3
```

图1-2 机器语言程序（5条机器指令）

令人尴尬的是，虽然这个机器语言是人发明的，但就算是发明这个机器语言的本人，如果不查阅相关的"说明书"或者"字典"，也是难以理解这段程序的。人们不得不去寻找一种方法，用简单、易理解、有限的符号来表示这个机器语言，使得这个符号语言很容易就能对应到机器语言，从而使得机器能理解和执行的同时，也能方便使用这个语言的人设计程序和阅读程序，这就是后来的汇编语言。"求1055 -(383 +545)的结果"这个任务的汇编语言程序如图1-3所示。

```
MOV AX 383    将383传送到AX寄存器
MOV BX 545    将545传送到BX寄存器
ADD BX AX     将BX内容加AX内容，结果在BX中
MOV AX 1055   将1055传送到AX寄存器
SUB AX BX     将AX内容减BX内容，结果在AX寄存器中
```

图1-3 汇编语言程序的例子

虽然汇编语言给人们进行程序设计带来了很大的方便，但在实际的程序和软件的"构造"中，还是有烦琐和效率低下等问题。随着计算机技术的发展，人们创立了更直观的所谓高级程序设计语言。这里的"高级"可以简单地理解为接近于某种自然语言、给程序设计带来了高效率以及更高级的程序设计方法等。为了展示高级语言的概貌，图1-4给出了一个用高级语言编写的略显复杂的程序，这段程序将完成把一个数字序列进行排序的任务。

初学的读者看不懂这些程序没有关系，本章只是在向读者介绍本书中用到的基本概念及其由来，有兴趣的读者可以通过查阅资料自己看懂这些语法。

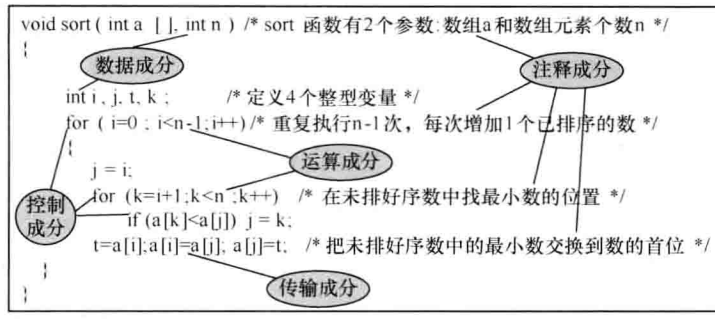

图 1-4 高级语言（C 语言）程序：选择法对 a 数组中的 n 个元素升序排序

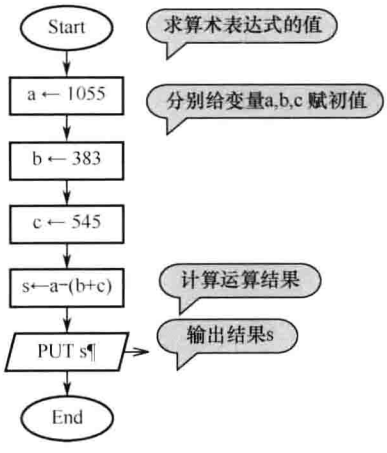

图 1-5 Raptor 程序

由于程序设计语言是构造程序和软件的基础，所以程序设计语言及其工具一直是计算机科学和技术中发展非常迅速的领域，新的语言和工具层出不穷。从培养计算思维能力的角度，过于复杂的工具并不适用于初学者，那么有没有适合初学者进行思维表达而且无须太多学习就能入门的语言和工具呢？答案是肯定的。例如，图 1-5 中的例子就是采用一个称为 Raptor 的工具来表达“求 1055 - （383 +545）的结果”这个任务的程序。这个程序对于大多数的初学者来说，很容易明白，要让计算机给自己解决问题，其实很简单，就是用流程图的方式表达自己的想法即可，这样，我们就可以把主要精力放在如何表达自己的想法上，而无须了解过多的程序设计技巧和工具本身的技巧，非常有利于思维能力的培养。

更一般地说，程序设计语言有 3 方面的因素，即语法、语义和语用。语法表示程序的结构或形式，即表示构成语言的各记号之间的组合规律，但不涉及这些记号的特定含义，也不涉及使用者。语义表示程序的含义，即表示按照各种方法所表示的各个记号的特定含义，但不涉及使用者。语用是语言成分相对语言情景的含义，即语言使用的语境不同含义可能不同（语用的讨论超出本书范围，有兴趣读者可以查阅相关文献资料）。

对于一个具体的计算机程序设计语言来说，它的目标就是利于进行程序设计，那么涉及计算机程序语言必然和计算机能做什么相关，计算机能存储指令和数据、接受输入、执行指令和处理数据、输出结果，所以，从这个层面我们很容易得到程序设计语言的基本成分（见图 1-4）：

① 数据成分，用以描述程序所涉及的数据。
② 运算成分，用以描述程序中所包含的运算。
③ 控制成分，用以描述程序中所包含的控制。
④ 传输成分，用以表达程序中数据的传输。

程序设计语言按照语言级别可以分为低级语言和高级语言。低级语言有机器语言和汇编语言。低级语言与特定的机器有关、功效高，但使用复杂、烦琐、费时、易出差错。机器语言是表示成数码形式的机器基本指令集，或者是操作码经过符号化的基本指令集。汇编语言是机器语言中地址部分符号化的结果，或进一步包括宏构造。高级语言的表示方法要比低级语言更接近于待解问题的表示方法，其特点是在一定程度上与具体机器无关，易学、易用、易维护。

在计算机科学和技术发展过程中，很多程序设计语言被研究和使用。由此产生了很多的程序设计语言以及程序设计方法和技术，形成了现代软件的技术基础。程序设计语言正朝着模块化、简明化、形式化、并行化和可视化发向发展，例如，支持基于模型驱动的开发方法的开发工具就是用可视化的方法建立模型，由模型直接生成程序代码。

1.5 算法和数据结构

瑞士计算机科学家尼古拉斯·沃斯（Niklaus Wirth）提出了一个著名的公式："算法 + 数据结构 = 程序"，从某种意义上说，正是这个公式使得他获得了计算机界的最高奖——图灵奖（1984 年）。这个公式告诉我们，程序的核心是算法和数据结构，而算法往往是基于某种数据结构来构造的。读者在以后的实践中将会慢慢体会到这种思维方式，我们在利用计算机进行实现求解问题时，必须设计相应的算法和数据结构，通过具体的语言和工具（如 Raptor）就得到了程序。本节将简单介绍算法和数据结构。

1.5.1 算法

所谓算法，就是一个有穷规则的集合。其中，规则规定了解决某一特定类型问题的一个运算序列。通俗地说，算法规定了任务执行/问题求解的一系列步骤。

应该看到，这里对算法的定义和前面对程序的定义似乎非常接近，容易混淆。首先，从公式"算法 + 数据结构 = 程序"可以看到，算法和程序显然是密切相关的，但算法不等于程序，程序却一定是算法，因为二者都涉及解决问题的方法步骤，都隶属于典型的计算思维范畴。由于我们在谈程序的时候都隐含着是在谈计算机的程序，从这个角度出发，程序是用一种计算机能理解并执行的计算机语言描述解决问题的方法步骤；算法也是解决问题的方法步骤，但不一定要让计算机能理解并执行。所以，我们解决问题首先是要考虑好算法，然后据此依托一个具体的程序设计语言实现这个算法即得到程序，也就是说，算法表达了解决问题的步骤，程序是算法的计算机程序设计语言代码实现。具体实现时，算法是靠程序来具体完成功能的，程序则需要算法作为其灵魂。所以，算法和程序其实是一体两面，算法更多地 体现的是思维层面，程序则是算法的实现层面。

我们可以从下面的例子中体会算法的含义，这个算法是经典的辗转相除法求最大公约数，也称为欧几里得算法。

算法：欧几里得算法（辗转相除法）。

STEP 0：输入两个正整数 m，n。