重点大学计算机教材

HZ BOOKS
华章教育

第2版

从问题到程序

程序设计与C语言引论

裘宗燕 编著
北京大学

机械工业出版社
China Machine Press

重点大学计算机教材

第2版

从问题到程序
程序设计与C语言引论

裘宗燕 编著
北京大学

机械工业出版社
China Machine Press

本书以 C 作为工具语言，讨论了基本程序设计的各方面内容，详细解释了与 C 语言和程序设计有关的问题。在新版中，特别加强了针对近年日益受到业界和学术界广泛重视的问题的讨论，并通过详细地分析和讨论大量符合 C99 标准的实例，给出了分析和分解问题、找出解决问题的主要步骤、确定函数抽象、找出循环、选择语言结构直至最后做出所需程序的完整过程。

本书适合作为高等院校计算机及相关专业第一门程序设计课程的教材，也可供其他学习 C 程序设计的读者自学使用。

图书在版编目（CIP）数据

从问题到程序：程序设计与 C 语言引论 / 裘宗燕编著. —2 版. —北京：机械工业出版社，2011.5
（重点大学计算机教材）

ISBN 978-7-111-33715-7

Ⅰ. 从…　Ⅱ. 裘…　Ⅲ. C 语言-程序设计-高等学校-教材　Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2011）第037223号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）
责任编辑：刘立卿
北京诚信伟业印刷有限公司印刷
2011 年 5 月第 2 版第 1 次印刷
185mm×260mm • 21.75 印张
标准书号：ISBN 978-7-111-33715-7
定价：39.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

前　言

作为计算机科学教育的第一门专业课程，程序设计课程的重要性毋庸置疑。本书的目标是作为第一门程序设计课程的教材或入门自学读物，内容集中在如何理解程序语言，如何做程序，如何为在计算机领域中的进一步学习和工作做好准备上。

本书以C作为工具语言，讨论了基本程序设计的各方面问题。书中实例没采用常见的“提出问题，给出程序，加些解释”的简单三步形式，而是强调问题的分析和讨论，意在帮助读者认识程序设计的实质，理解从问题到程序的思考过程。书中尽可能详细地解释了与C语言和程序设计有关的各种问题，通过大量实例讨论了分析和分解问题，找出主要步骤，确定函数抽象，找出循环，选择语言结构，最后做出所需程序的过程。

程序设计应该注重系统性和科学性，有关研究的发展形成了程序的理论。入门书不可能深入介绍理论成果，但也要反映其精神，使初学者从开始就接触到程序的一些本质问题。本书特别强调问题的分析，将函数的概念尽可能前移，而后反复讨论，实例中也特别注重程序的功能分解。书中还介绍了一些深入的理论问题，如通过统计程序运行时间介绍计算的基本性质（复杂性），通过分析循环过程是否完成所需工作介绍“循环不变关系”的概念等，这是希望帮助读者了解更多情况，也作为思考程序的线索。

程序设计也是一种工程性工作，需要寻找可能的解决方案，评价不同方案并做出选择，还要对所做选择有清醒的认识（优点、缺点、倾向性或不足）。工程中通常没有完美的结果，更多是权衡和折中。程序设计也是如此。本书的许多实例包含较详细的分析讨论，不少实例给出了基于不同考虑形成的多种解法和性质比较，有时还指出其他可能性，提出了还可能如何看问题、如何做等，以帮助读者理解程序设计的真谛。书中还常给读者提出一些问题，希望读者发挥思维能力和主观能动性。书中练习也力图反映这些想法。

本书强调：编程问题没有需要记住的标准答案。由于对问题的不同考虑，设计中的不同选择，对同一问题可以得到许多合理而正确的不同程序。它们常常各有长短或各有侧重，或反映了对问题的不同认识。在这里应特别学习如何分析问题，如何把复杂问题分解为相对简单的部分，如何在可用功能中选择等。进而，还应该认清各种选择的后果，包括收获和损失。各种书籍（包括本书）里的程序不是“金科玉律”，只是作者对问题的一种理解，必然存在很多其他可能性。要学好程序设计，应该养成一种习惯：读程序时要特别注意思考作者的考虑和选择，分析其中哪些是合理而有价值的（或不合理无价值的），还可能有哪种选择，采纳其他选择可能得到什么（或失去什么）等。这样思考将使人受益无穷。当然，这并不是说书中示例不重要。恰恰相反，因为程序设计中有许多可能选择，好的教科书应当给出好的分析和选择，供读者参考。入门书还应解释所做选择的理由，指出带来的问题（缺点和限制等）。正如本书书名所示，程序设计是“从问题到程序”的思考和工作过程。学好程序设计，既要发挥聪明才智，也需要细致认真、一丝不苟的工作态度。

考虑入门课程的需要，并没有一种语言具有无可比拟的优势，无论用什么语言教基本程序设计，都需要考虑其有利方面，处理其不利因素。本书选择C语言的主要理由是：C是使用最广的语言之一，包含了基本程序设计需要的主要机制，能满足课程的需要。学生可以用它完成练习，得到相关的知识积累和能力锻炼，还能掌握一种实用工具，作为后续课程学习的基础。C语言适合（也正在被）作为许多计算机专业课程的教学语言。

C 是很灵活的语言，用它写程序常需要了解一些细节，这是人们对用 C 作为基础课语言的主要疑虑。但另一方面，用 C 写程序可能得到对程序设计的更多认识。C 程序可以在较低层次上做，也可以在较高层次上做，学生需要理解程序设计中各种层次的问题。此外，许多语言从 C 借鉴了想法和表达形式，有些就是 C 的扩充和发展，C 语言知识对于进一步学习其他语言，包括未来的新语言都很有价值。

本书以程序设计为基本线索，也详细介绍 C 语言的各方面情况。这里强调的是如何认识程序、写程序、用 C 语言写程序，特别强调好的程序设计风格，强调“好的”C 程序设计，强调通过函数抽象建立清晰结构的重要性，强调程序的良好结构、可读、易修改，也指出了多种不良编程习惯及其危害。历史原因使 C 成为一个不太严格的语言，如不注意，用 C 写的程序有可能隐藏一些不易发现的错误⊖。ANSI C 和 C99 倡导“好的”C 程序。本书坚持这一正确方向，讨论如何写出更可靠且不易包藏隐含错误的清晰、简洁、高效的 C 程序，通过实例说明应该如何写和不应该如何写。书中介绍了许多实用的 C 程序设计技术，详细解释了 C 语言的结构和机制，也尽量提供了一些背景说明。

本书包括如下各章和若干附录：

第 1 章，程序设计和 C 语言。介绍程序与程序语言的概念，C 语言的发展及其特点。用一个小例子介绍 C 程序形式，其加工和执行。最后介绍程序设计与开发过程。

第 2 章，数据与简单计算程序。讨论程序语言的许多最基本概念，包括：字符集、标识符和关键字，数据与类型，数据表示，运算符、表达式与计算过程，数学函数库的使用等。

第 3 章，变量、函数和流程控制。讨论程序设计的一些基本问题，包括语句与复合结构，变量及其使用，简单函数定义，逻辑条件的描述与使用等。最后介绍了几种基本控制结构。

第 4 章，基本编程技术。首先讨论循环程序设计的基本问题，通过一系列程序实例分析了循环的构造过程。此后介绍了 C 语言其他控制结构及其使用。

第 5 章，C 程序结构。讨论 C 语言的许多具有一般性的重要问题，主要是 C 程序结构，函数概念及有关的问题，预处理命令和预处理过程，递归的概念等。

第 6 章，顺序数据组织：数组。介绍数组概念、定义和相关程序设计技术。

第 7 章，指针的应用。介绍指针概念和指针变量的使用，指针与数组的关系，多维数组作为参数的通用函数，以及动态存储管理，类型定义，指向函数的指针等。

第 8 章，文件和输入输出程序设计。讨论了文件概念，与输入输出有关的各种问题，标准库的输入输出功能，以及输入输出的程序设计问题。

第 9 章，结构和复杂数据组织。介绍结构（struct）、联合（union）、枚举（enum）等数据定义机制的意义及在程序中的使用。随后简单介绍了链接结构的概念。

第 10 章，程序开发技术。介绍程序设计和开发的一般性问题和技术，分块开发等。

第 11 章，标准库。介绍标准库提供的各方面功能及其相关知识。

第 12 章，C99 导引。介绍 C99 的重要扩充，变长数组和标准库的几个新数值包。

最后是两个附录和进一步学习的建议。

本书以 ANSI 标准 C 语言为背景，参考了 C99 的新发展，内容不依赖任何具体 C 系统。读者可用任何符合标准的 C 系统作为编程环境，如国内常见的微机、工作站或其他计算机上的 C 系统。本书里的所有实例程序按 ANSI C 和 C99 标准书写，习题不涉及任何具体系统环境。读者可以用各种 C 系统作为学习工具，如国内使用较多的 TURBO C 系统，一些公开的免费 C 语言系统（如普通微机上可用 lcc、Dev-C++等）。

⊖ 任何语言都有弱点。有句话说：“最好的语言也不能防止人写出坏程序。”这不是说语言不重要，而是说任何语言都有正确使用的问题。C 在这方面的问题突出一点，读者应特别注意。

新版说明

本书最早于 1999 年在北京大学出版社出版，2005 年在机械工业出版社重新出版时做了大幅度修改，反映了对许多问题的新认识。现在这个新版对内容和结构进行了全面修订，特别加强了针对近年日益受到业界和学术界重视的问题的讨论。C 语言系统和相关教育正在向 C99 标准转变，但本书并不强调这一转变，因为 C99 是 ANSI C 的一个合理扩充，任何好的 ANSI C 程序都是好的 C99 程序。本书前一版里的示例程序早已贯彻了 C99 的精神，符合 C99 标准。本版新增的第 12 章是上一版附录 C 的扩充和增强。这一版的主要修改包括：

1. 进一步贯彻了以程序设计的思想和技术为中心的编写思想。本书原本就不打算作为一本“考试用书”，其目标是培养有能力、有发展潜力的计算机程序工作者。这一基本想法在新版的内容和组织中得到了更好的反映。
2. 书中各章的基本部分只介绍 C 语言中在一般程序设计里使用最广泛的重要内容，以便更集中地讨论相关的程序设计技术和方法。有关 C 语言一些细节问题的解释集中放在各章最后，单独列为一节。这种做法既避免了过多的语言细节干扰讨论的主干内容，也使课程教学内容的安排有了更多选择。
3. 近年计算机的应用飞速发展，连接全世界大量计算机的互联网形成了一个全球性的信息环境，越来越多的计算机系统嵌入在各种机器、设备、仪器、交通工具、基础设施里。在这种大背景下，计算机系统的安全性变得越来越重要。与此同时，计算机软件的脆弱性带来的危害和威胁也越来越严重。程序的强健和安全是计算机系统安全的基础。但以往的编程教科书基本上不讨论程序安全性，不考虑程序输入或函数参数不满足需求时的程序行为问题。一些专家认为，计算机基础教育中的这种缺失是今天计算机系统脆弱的一个重要根源。本书大大加强了安全性方面的讨论，在讨论各种程序设计技术的同时，认真分析了相关 C 程序结构中的脆弱点和可能的编程缺陷，提出了提高程序强健性的技术手段。作者相信，这些讨论对于培养明天合格的计算机程序工作者是非常重要的。希望这一考虑能受到教授相关基础课程的老师和本书自学者的重视。

作者特别感谢北京大学理科试验班和数学学院参加相关课程学习的同学们和参加辅导工作的研究生们，他们的思考和问题给了作者许多启示，使作者更深入地理解了许多问题。作者还要感谢家人与同事多年的支持。华章公司有关人员为本书做了大量工作，在此致以特别的谢意。虽然本书凝结着作者多年的思考，但仍难免包含错误或不足，这些都由作者个人负责，也希望得到读者的指正和同行的宝贵意见。

裘宗燕
北京大学数学学院信息科学系

本书中的编程实例和模式

目　录

第 1 章　程序设计和 C 语言

在开始学习程序设计时，初学者首先遇到的问题是：什么是程序？什么是程序设计语言？本章首先讨论这些问题，帮助读者比较直观地建立起对程序、程序设计和程序设计语言的基本认识，而后简单介绍本书中讨论程序设计问题时所用的程序设计语言——C 语言，并通过一个简单实例介绍 C 语言程序的一些基本情况和有关概念。最后介绍在实际做程序设计中必然要遇到的一些问题。

1.1　程序和程序语言

*程序*一词来自生活，通常指完成某项事务的一套既定活动方式或者活动过程。从表述上看，我们可以把程序看成对一系列动作的进行过程的描述。日常生活中也可以找到许多“程序”实例。例如，一个学生早上起床后的行为可能描述为：

1. 起床
2. 刷牙
3. 洗脸
4. 吃饭
5. 早自习

这是一个直线形的程序，由一系列更简单的活动（基本步骤）组成。这就是最简单的程序形式。描述这种程序，也就是给出一个包含某些基本步骤的序列。如果按顺序实施这些步骤所描述的动作，其整体效果就完成了一项事务或者工作。

考虑另一个复杂些的过程：到图书馆借教学参考书。这一常见过程可以描述为：

1. 进入图书馆
2. 查书目
3. 填写索书单
4. 交图书馆工作人员取书
5. 如果该书已经借出，可以有两种选择
   5.1. 回到第 2 步（重新查书目借其他参考书）
   5.2. 放弃借书，离开图书馆
6. （如果图书还在，工作人员找到要借的书）办理借书手续
7. 离开图书馆

这个程序比前一个复杂一些。主要的复杂性就在于，这一程序不是个平铺直叙的动作序列，其中步骤更多，有时需要根据遇到的情况处理，还可能出现重复动作。

仔细探究这个实例，还可以认识到这一程序可能需要进一步细化，以处理一些实际中可能需要考虑的复杂情况。不难找出许多在上面描述中未处理的情况。譬如说：查找图书目录时没有找到所需的书籍；填写好索书单时已经到了图书馆的下班时间；借书时发现自己没有带借书证；工作人员查到该读者的借书册数已经达到限额，或发现该读者有逾期未还的图书，因此拒绝出借；等等。

从这些现实生活中的例子里，可以了解“程序”概念的一些直观特征。现实生活中有许许多多这样的程序性的活动，当我们身处其中时，通常需要按部就班地一步步完成一系列动作。对这种工作（事务、活动）过程的细节动作描述就是一个“程序”。

在一个程序描述中，总有一批预先假定的“基本动作”，这些基本动作是执行程序者能理解和直接完成的。例如，在上面有关借书的程序描述中，我们把“查书目”当做一个基本动作。如果来图书馆的读者不知道如何查书目（例如刚进学校的新生），那么，在给这种读者的程序描述中，就需要把“查书目”动作进一步细化，描述查书目的细节过程。例如，通过图书馆的计算机检索系统查图书目录的过程可以描述为：

通过图书馆的计算机进入书目检索系统；
输入有关要检索图书的信息；
从检索到的图书列表中选择

这就是程序的进一步细化，或者叫做功能分解。如果借书的人不知道如何启动计算机检索系统，那就需要对上述程序做进一步分解。这种逐步细化或者分解的过程，也是本书后面有关计算机程序设计的讨论中最本质的东西。

一个程序通常都有开始与结束。在执行一个程序的过程中，动作者（无论是人还是机器）需要按程序的描述执行一系列动作。在到达结束位置时整个工作完成。

本书将要深入讨论的计算机程序同样具有这些特征。

计算机、程序与程序设计

日常生活中的程序性活动与计算机里的程序执行类似，这一情况可以帮助我们理解计算机的活动方式。当然，日常生活中的程序性活动里有更多变数，许多事情并不要求完全按程序做，可以有许多“灵活性”。而计算机对程序的执行则完全是严格、一丝不苟的，计算机将一步步按程序中的指令行事，没有一点“商量”的余地。

计算机是人类发明的一种自动机器，它能完成的工作被称为“计算”。实际上，计算机的最基本功能是可以执行一组基本操作，每个操作能完成一件很简单的工作，例如做一次整数的加减乘除运算，或者把一个数从这里搬到那里，或者比较两个数的大小等等。为使人们能指挥计算机工作，每种计算机都提供了一套指令，其中每种指令对应于计算机能执行的一个基本动作。而所谓的“计算机程序”，也就是一组这种指令形成的序列。

作为看得见、摸得着的物理实体，计算机的基本原理非常简单，其最本质的特征就是能自动地按程序（作为计算机能执行的基本动作序列）工作。人与计算机打交道的基本方式就是根据自己的需要，用计算机规定的形式写出一个程序，而后把这个程序提供给计算机，命令它去执行。此后计算机就会按程序的规定，一丝不苟地执行其中的指令，直至程序结束。人们常把计算机执行程序的过程简单地说成是程序的执行过程。

计算机是一种通用的计算机器，给了它一个或者一组程序后，它就变成了处理某个专门问题、完成某种特殊工作的专用机器。这种通用性与专用性的统一非常重要。这样，一方面，计算机可以在大工厂里采用现代化生产方式大量生产；另一方面，通过运行不同程序，一台计算机可以在不同时候处理不同问题，甚至同时处理许多不同的问题。这就是计算机威力的真谛。人们描述（编制）计算机程序的工作被称为程序设计或者编程，这种工作的产品就是程序。由于计算机的本质特征，从计算机诞生之初就有了程序设计工作。

今天，计算机的发展及其在各领域的广泛应用，对人类社会生活各方面的深刻影响已经是众所周知的事实了。计算机之所以能产生这样大的影响，其原因不仅在于人发明并大量制造了这样一种令人敬畏的奇妙机器，更重要的是人们为计算机开发出了数量宏大、五彩缤纷、能指挥计算机完成各种简单或复杂工作的程序。目前正在使用的计算机没有多少种，而正是数量繁多、功能丰富多彩的程序给了计算机无穷无尽的“生命力”。

程序设计语言及其发展

由上面介绍可以看到程序和程序描述的另一些问题。例如，对于上面学生借书的例子，我们提出了许多需要考虑的细节，提到了基本动作的问题。一台计算机有着预先确定的一组基本动作，每个动作只能完成很少的一点工作。如果需要用计算机去处理复杂问题，可能需要写出很长很长的程序，而且必须精确描述执行所有动作的细节过程，不能有一点错误，不能有一点含糊其辞之处。这些就是程序设计的所有困难的根源之所在。

要说明一个程序在执行中需要做些什么，就需要给出这一程序性活动的一步步的细节过程，需要描述程序执行中的各种动作及其执行顺序。为了做这件事，需要有一种适当的描述方式。一

套系统的描述方式就形成了一个语言。

语言一词通常指人生活中使用的自然语言，如汉语、英语等。这些语言随着人类发展进步而自然形成，是人们交流信息的工具和媒介。人们用口头语言传播见闻、表达看法和想法；用书面语言写文章、书籍，实现更大范围的信息交流。在前面描述的现实生活中的程序实例中，我们就是用汉语作为描述程序的语言，描述的程序是为了给人看，要人做的。

为了与计算机交流，指挥计算机工作，同样需要有与之交流的方式，需要一种意义清晰、人用起来比较方便、计算机也能处理的描述方式。也就是说，需要有一种适用的描述程序的语言。供人编写计算机程序用的语言就是程序设计语言，这是一类人造语言。程序设计语言也常被称为编程语言，本书中常常简称为程序语言，在上下文清楚之处简称为语言。

有人可能说：小学生就开始学数学，数学的一个基本部分是计算，小学生已经会用数学方式（或者说，用数学语言）描述计算过程，程序设计语言还有什么特殊之处吗？确实有！程序语言的一个突出特点就在于不仅人能懂得和掌握它，能用它描述所需的计算过程，而且计算机也可以“懂得”它，可以按程序语言给出的关于计算过程的描述去行动，完成人们所需要的计算工作。程序设计语言是人描述计算的工具，也是人与计算机交流信息的媒介：通过用程序语言写程序，人能指挥计算机完成各种特定工作，完成各种计算。

在计算机内部，一切信息都以二进制编码的形式存在。需要放进计算机、要求计算机去执行的程序也不例外。计算机总规定了一套描述其指令的二进制编码形式，这种描述形式称为机器语言，计算机能直接“理解”和执行它。用机器语言写出的程序称为机器语言程序。计算机诞生之初，人们只能直接用机器语言写程序。对人的使用而言，二进制的机器语言很不方便，用它写程序非常困难，工作效率极低，写出的程序难以理解，正确性很难保证，发现程序有错误也很难辨认和改正。下面是一台假想计算机上的指令系列：

```
0000000100000001000  -- 将单元1000的数据装入寄存器0
0000000100010001010  -- 将单元1010的数据装入寄存器1
0000010100000000001  -- 将寄存器1的数据乘到寄存器0原有数据上
0000000100010001100  -- 将单元1100的数据装入寄存器1
0000010000000000001  -- 将寄存器1的数据加到寄存器0原有数据上
0000001000000001110  -- 将寄存器0里的数据存入单元1110
```

这里想描述的是计算算术表达式a × b + c（符号a、b、c分别代表地址为1000、1010和1100的存储单元），而后将结果存入单元1110的计算过程（程序）。

一个复杂程序里的指令可能有成百万、成千万条，或者更多，程序中的执行流程错综复杂，在二进制机器语言的层面上理解一个复杂程序到底做了什么，很容易变成人力所不能及的事情。为缓解这一问题，人们很快就发展出符号形式的、使用相对容易些的汇编语言。用汇编语言写的程序需要用专门软件（汇编系统）加工，翻译成机器语言后才能送给计算机去执行。下面是用某种假想的汇编语言写的程序，它完成与上面程序同样的工作：

```
load 0 a -- 将单元a的数据装入寄存器0
load 1 b -- 将单元b的数据装入寄存器1
mult 0 1 -- 将寄存器1的数据乘到寄存器0原有数据上
load 1 c -- 将单元c的数据装入寄存器1
add  0 1 -- 将寄存器1的数据加到寄存器0原有数据上
save 0 d -- 将寄存器0里的数据存入单元d
```

汇编语言的每条指令对应于一条机器语言指令，但采用了助记的符号名，存储单元也用符号形式的名字表示。这样，每条指令的意义都更容易理解和把握了，理解整个程序也容易了许多。但是，汇编语言里没有高层结构，只是基本汇编指令堆积形成的长长的序列，是一盘散沙。用汇编语言写复杂程序仍然很困难，写出的程序难以理解，其中的错误难以检查。

1954年诞生了第一个高级程序语言FORTRAN，宣告了程序设计的一个新时代的开始。FORTRAN采用完全符号化的描述形式，用类似数学表达式的形式描述数据的计算。语言中提供

了有类型的变量，作为计算机存储单元的抽象模型。此外还提供了一些流程控制机制，如循环和子程序等。这些高级机制使编程者可以把复杂的程序分解为一些较小、较容易把握的部分，在工作中摆脱许多具体细节，方便了复杂程序的书写，写出的程序更容易阅读，错误也更容易辨认和改正。FORTRAN 语言诞生后受到广泛欢迎。

高级语言及其实现

高级程序语言具有人更容易习惯的描述形式，更容易使用，这也使更多的人能够并乐于加入到程序设计活动中。用高级语言写程序的工作效率更高，它使人们能开发出更多的应用系统，这种情况反过来推动了计算机应用的发展。应用发展又推动了计算机工业的大发展。可以说，高级程序设计语言的诞生和发展，对计算机发展到今天的程度起了极其重要的作用。

从 FORTRAN 语言诞生至今，人们提出的语言已经有数千种，其中大部分只是试验性语言，只有少数语言得到广泛使用。随着时代的发展，今天绝大部分程序都是用高级语言写的，人们也已习惯于用*程序设计语言*特指各种高级程序语言了。

在高级语言（例如 C 语言）的层面上，描述前面同样的程序片段只需要一行：

```
d = a * b + c;
```

这表示要求计算机算出等于符号右边的表达式的值，而后把计算结果存入由 d 代表的存储单元里。这种表示方式很接近人们所熟悉的数学形式，明显地更容易阅读和理解。高级语言程序完全采用这种符号形式，摆脱了难用的二进制形式和具体计算机的细节。此外，高级语言中还提供了许多高级的程序结构，供编写程序的人们用于组织复杂的程序。与机器语言和汇编语言的程序相比，情况确实大大改善了。

当然，计算机也不能直接执行高级语言描述的程序。人们在设计好一个语言后，还需要开发出一套实现这一语言的软件，这种软件被称为*高级语言系统*，也常被说成是这一高级语言的实现。在研究和开发各种高级语言的过程中，人们也研究了各种语言实现技术。高级语言的基本实现方式有两种，分别称为*编译*和*解释*：

1. 采用编译方式实现高级语言：人们首先针对具体语言（例如 C 语言）开发出一个翻译软件，其功能就是把用该种高级语言写的程序翻译成所用计算机的机器语言的等价程序。我们用这种高级语言写好一个程序后，可以把它送给这个翻译程序处理，得到与之对应的机器语言程序。在此之后，我们只要命令计算机去执行这个机器语言程序，计算机就能完成所需要的工作了。
2. 采用解释方式实现高级语言：人们首先针对具体高级语言开发一个解释软件（系统），该软件的功能就是直接读入这种高级语言写的程序，并能一步步按照程序的要求工作，完成该程序所描述的计算。有了这种解释软件，我们只要直接把写好的程序送给运行着这个软件的计算机，就可以完成该程序所描述的工作了。

在目前的实际计算机系统中，采用第一种方式实现高级语言的情况更为常见，也有许多采用第二种方式实现语言的系统。本书后面还会进一步介绍前一实现方式的一些具体情况。

随着计算机科学技术的发展，人们不断提出新的程序语言，老语言逐渐被淘汰，仍在使用的老语言也在不断变化。以 FORTRAN 语言为例，它在过去的 50 年已经过多次大的改版，其新版本（如 FORTRAN 90 和 95）与初始的 FORTRAN 相比早已面目全非了。其他历史较长的语言也都如此。推动语言发展的因素很多。随着程序设计工作的长期实践，人们对程序设计应该怎样做、需要什么东西去描述程序等不断产生新的认识，其中许多重要认识后来凝结到新语言里。推动语言发展的另一原因是应用。新的应用领域常对描述工具提出新要求，这些认识和要求促使人们改造已有的语言，或者提出新语言。

目前世界上使用较广的语言有 FORTRAN、C、C++、PASCAL、Ada、Java 等，这些语言通