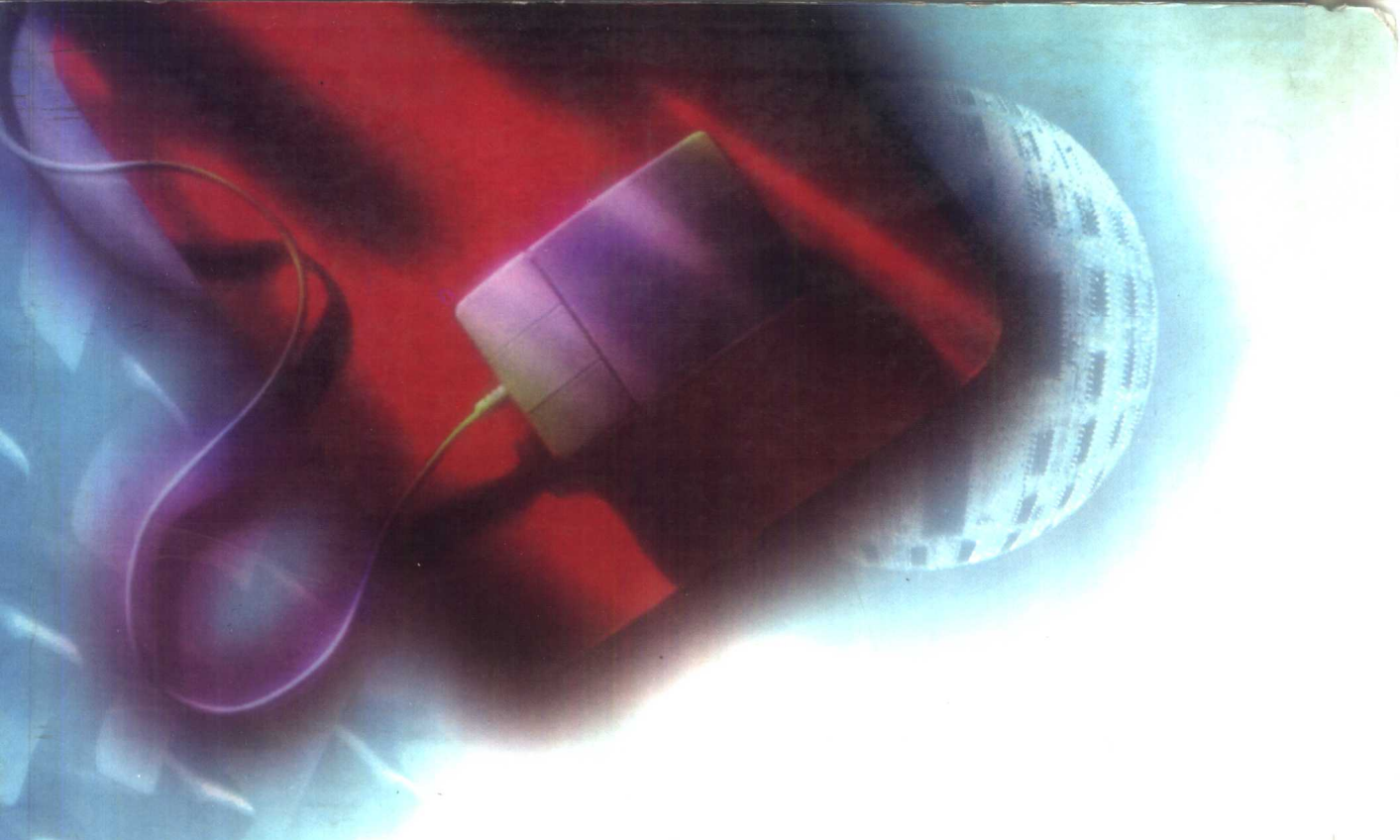

从问题到程序

——程序设计与C语言引论

裘宗燕 著

CONG WENTI DAO CHENGXU

CHENGXU SHEJI YU C YUYAN YINLUN

北京大学出版社

从问题到程序

——程序设计与C语言引论

裘宗燕 著

北京大学出版社
北 京

内 容 提 要

本书是面向理工类大学本科、专科第一门程序设计和C语言课程而编写的教材，也可供广大计算机工作者，电脑爱好者阅读参考，可以作为没有或只有很少程序设计经验的读者的入门读物或参考书。本书以程序设计为主线，介绍了C语言以及C语言程序设计方面的知识，对于如何从问题出发写出程序做了深入讨论，强调了分析问题、对问题的分解、确定主要步骤、确定函数抽象、找出循环、选择语言结构、最后写出程序的整个过程。书中特别注意强调"好的"C程序设计方式，坚持了ANSI C语言标准所倡导的正确写法，分析了一些不良程序设计习惯的危害。书中详细介绍了C语言的结构、机制及一些背景，以帮助读者理解问题的实质，还介绍了许多实用的C程序设计技术。总之，本书希望告诉读者如何写出正确、清晰、简洁、高效、可读、易修改的C语言程序。对于基本程序设计和C语言中反映出的程序设计、程序设计语言及计算机科学中的一般性问题也做了适当的介绍和解释。书中给出了许多程序实例，对不少实例还给出了在不同考虑下可能形成的多种解法，提出了一些请读者进一步思考的问题。每一章都附有较丰富的程序习题。

本书共分9章，分别是：程序设计、C语言与C程序；数据对象与计算；变量、函数和控制结构；循环与程序控制；C语言程序结构；数据对象的组合：数组；指针；结构及其他；文件与标准库函数。

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/裘宗燕著. —北京：北京大学出版社，1999.4
ISBN 7-301-04066-0

Ⅰ.从… Ⅱ.裘 Ⅲ.C语言-程序设计 Ⅳ.TP312

书　　名：从问题到程序——程序设计与C语言引论
著作责任者：裘宗燕
责任编辑：沈承凤
标准书号：ISBN 7-301-04066-0/TP·444
出 版 者：北京大学出版社
地　　址：北京市海淀区中关村北京大学校内，100871
网　　址：http://cbs.pku.edu.cn/cbs.htm
电　　话：出版部 62752015　发行部 62754140　62752037
电子信箱：zpup@pup.pku.edu.cn
排 版 者：兴盛达激光照排中心
印 刷 者：北京银祥福利印刷厂印刷
发 行 者：北京大学出版社
经 销 者：新华书店
787毫米×1092毫米　16开本　17.875印张　443千字
1999年4月第一版　1999年4月第一次印刷
定　　价：21.00元

前　言

程序设计在计算机科学教育中的重要性是无庸置疑的。在多年讲授计算概论课的过程中，我一直在思考的一个问题是：学生在第一门程序设计课里应该学习什么？这引出了许多有关问题，比如：第一门程序设计课应该怎样讲？作为第一门程序设计课程的教材应该是怎样的？等等。这几年我们一直用C语言作为第一门程序设计课的教学语言，选用过不少教材，包括国内教材和国外的原版教材。虽然这些教材各有特点，但对于第一门程序设计课而言，用起来都有不少困难。因此我才动手编写了这个教材，可以说是对于自己教学的总结，也反映了目前自己对上面提出问题的认识。

在以往教学过程中学生们提出了许多问题，这使我看到，要编写一本适合第一门课程使用的新教材（或适合自学者的第一本程序设计书籍），对学习过程中遇到的各方面问题，无论是关于C语言本身的，还是关于程序设计过程的，都应该尽量给出合理而充分的解释，使读者、学生在阅读或复习时，能够从中了解问题的实质，而不是自己在模糊中去设法感悟。这对于帮助初学者尽快进入程序领域是非常重要的，也可能使教师在教学中有更大的自由度，更灵活地选择自己的例子和教学方式，有关具体问题的信息可以由学生自己由书本得到。另一方面，由于程序设计本身的性质，在基础书籍和课程中不应采用"提出问题，给出解答，再加一点解释"这样简单的三步循环程序，而应当更着重于帮助初学者认识程序设计活动的实质，理解从问题到程序的思考过程。本书在这些方面做了一些努力，希望能包含对许多有关问题的解释和比较详细的说明，供读者学习时参考。

本书编写中一个主要想法是要反映程序设计工作两个重要方面的特征。一方面是程序设计工作的科学性，另一方面是程序设计工作的工程性。关于科学性，我们指的是：程序本身的构造过程应当有充分的科学依据。对要解决的问题如何进行分解，怎样看清楚程序各部分的意义和互相联系，这些都需要编程序的人对程序实现过程有科学的认识，这方面研究的发展形成了程序的理论。虽然在初级课程或书籍里不可能深入讨论有关的理论及成果，但却必须反映程序理论的精神实质，使初学者从一开始就接触到程序及程序语言的一些本质性问题。这一点，无论从提高读者对计算机科学技术认识水平的角度看，还是从将来进一步深入学习的角度看，都是十分必要的。

由于计算机的日益普及，不少学生在进大学前就有了用计算机的经验，有些人计算机已经玩得很熟，有些人已经写过许多程序。但是也应该看到情况的另一面：在学习大学有关的课程之前，学生中对程序及程序设计已经建立了比较正确的认识的人仍是凤毛麟角。即使是那些对计算机已经很熟的学生，实际上也有许多观点需要重新建立，或者有一些错误观点需要纠正。这也是在这个课程里应该特别强调科学性的原因。

在教学中我发现，学生常有对计算机有关问题的实质的不正确认识，有些看法可能连学生自己也没有清楚的意识。例如，有些学生在思想深处把计算机看成一种很不驯服的东西（动物？），程序设计在其头脑深处的形式是设计一些计谋，"骗"计算机来帮人干活。有些学生实际上把程序设计看成像是玩一种计算机游戏，这里到处都是陷阱和妖魔，玩家的办法就是在一次

次试探和失败中总结经验，找一条绕过各种障碍的路。

今天日益完善的程序开发环境有时也成为学习程序设计的障碍。很好的环境也造就了这样一种学生：他们对系统玩得很熟，键盘也打得很快。遇到程序作业，这些学生不是先做细致的分析和设计，而是粗粗一想就动手，很快就写出一大堆代码，随后就进入简单的三步工作循环：运行、追踪、打补丁改错。对于简单问题，程序很快也就这样做出来了，但质量一般都很差，常常是一个简单问题就写出了一大篇，自己也说不清解决问题的细节过程。即使是对程序很熟的人，不花点工夫也难判断这种程序的对错，以及如果错，错在哪里。当面临的问题变得复杂时，这些同学就遇到了困难，写的程序往往很难通过测试，很难弄清错误究竟在那儿，应该怎样修改补救。这种程序就像基础没打好而摇摇欲坠的大楼，无论如何修补都难以解决问题。对于这种"工作成果"，最合理的建议就是推倒重来，把问题分析清楚后重新动手做。这些情况给人的提示是：从最简单的程序开始就应该注意程序设计的正确途径。

上述问题说明了在书籍和课程中进一步强调程序设计过程的科学性是非常必要的，基础课程中更是这样。本书注意了这方面的讨论，从开始就强调对问题的分析和分解，对函数抽象方法做了许多讨论，讨论程序实例时特别注意这些问题，后面章节又不断有进一步的论述。书中一些地方还通过程序实例介绍了程序理论中一些更深入的问题，如通过对程序运行时间的测试，介绍计算过程的一些基本性质（复杂性）；通过对循环过程是否完成了所希望的工作的分析，介绍"循环不变关系"的概念和意义等。当然，对这些问题都没有进行深入讨论，只是希望读者看到有关情况，作为自己思考程序问题的线索。

程序设计工作另一方面特征是其工程性。这里说的工程性不是指规模大，参加人员多这类性质，而是从另一个角度看问题。在工程计划设计中需要有对问题的分析和理解，设法寻找可能的解决方案，对各种可能解决方案做出评价和选择。应当对所做的选择有清楚的认识（例如有哪些优点和缺点，是否对要解决问题的某些方面有所偏向或不足等等），并进一步确定具体的实施方法。这些问题在程序设计过程中都有直接反映。如果我们不是把学习程序设计的目的定位在解决几道典型的程序题目上，而是放在帮助学习者提高能力、真正理解程序设计过程方面的话，这些问题都是需要特别强调的。

为帮助读者认识程序设计在这方面的实质，本书的叙述和实例都力图强调这样的观点：对于一个程序问题，并没有什么标准答案需要记住。由于对问题分析时的不同考虑，由于在程序设计过程中的不同决定或选择，对于同一个问题，人们常常能够得到许多合理的"能解决问题"的程序。这些程序常是各有长短，可能有侧重点的不同，也可能确实反映了人们对问题的不同认识。学习时应特别注意解决问题的方法，包括如何分析问题，如何逐步把要解决的问题弄得更清楚明确，如何寻找可能的求解途径，把复杂问题分解为相对简单的步骤，如何确定可使用程序语言结构并从中选出合用的，等等。这里的每一步都可能产生分支，而在做选择时，则应该弄清选择的后果，无论是收获还是损失。

工程中往往没有十全十美的选择，要做的更多是平衡和折衷。这些对于程序设计同样是本质性的。本书对许多实例给出了较详细的分析过程，有时对一个问题给出了多个解答，有时指出了其他的可能性：还可能如何看问题，还可能如何做，等等。对于有的实例，书中还给读者提出了许多问题，希望读者发挥自己的思维能力和主观能动性。各章节后面的练习也试图反映这些看法。

这样做的目的就是希望读者能建立一种认识：程序设计就是这样一种工作，这里并没有什

么"标准答案"。程序设计过程中要追求的是比较好的"正确"答案,而各种书籍上给出的答案(包括本书里给出的)不过是作者对于问题理解和分析的反映。其他可能的选择也很多。进一步地,我们还希望读者养成这样的习惯:看到别人的一个样例程序时,应该想一想这个程序中隐含了程序作者的哪些考虑和选择,其中哪些是合理的有价值的(或者是不合理的没有价值的),在哪些地方还可能有什么选择,沿着其他选择走下去可以得到什么(或失去什么),如此等等。如果读者能够养成这种思考习惯,并且坚持这样做,必将从中受益无穷。当然,这并不是说书籍中给出的例子不重要。恰恰相反。正因为在程序设计过程中有许多选择的可能性,书籍中的实例应当给出合理的好选择,供读者参考。对于初等的入门书籍,应当同时说明为什么这样做的理由。如果可能,还应当指出采取这些选择带来的问题(缺点、造成的限制等)。

正如本书的书名所言,程序设计是"从问题到程序"的一个工作过程。这个工作既要求遵循严格的科学方法,又要求谨慎灵活的工程能力。要很好地完成程序设计工作,编程序的人既需要充分发挥自己的聪明才智,又需要有细致认真、一丝不苟的工作态度。即使是将来不从事计算机程序方面的工作,通过这个课程得到的锻炼也可能非常重要,尤其对于理科学生,这个课程可能对于弥补其工程方面训练的不足有所帮助。

前些年,第一门程序设计课常用 PASCAL、FORTRAN 或 BASIC 等语言讲授。由于这几个语言在一些方面的不同弱点,目前第一门程序设计课的教学语言正在向 C 语言或者其他类 C 语言(如 C++等)转移。在目前,从作为第一门程序设计课程教学用语言的角度看,没有任何一种程序语言具有无可比拟的天然优势,选择任何语言都需要考虑其有利方面,也需要努力克服这种选择带来的不利之处。本书是为把 C 作为第一门程序设计课的基础语言而编写的教材,书中不假定读者(学生)已学过某种程序设计语言、有了一定程序设计经验。本书的编写反映了作者在把 C 语言作为教学语言时的许多考虑。

选择 C 语言作为程序设计基础课的教学语言的主要理由有:C 语言是目前实际程序设计工作中使用最广泛的语言之一,它包含了进行程序设计需要理解和使用的基本程序机理和重要机制。正确地掌握这些机制能够理解程序与程序设计的主要问题,完成各种作为练习的程序,并得到有关方面的知识积累和能力锻炼。目前有许多简单或复杂的软件系统是用 C 语言编写的,或者基本上是用 C 语言编写的。以 C 语言作为基本语言,不但能够学习程序设计,同时也能掌握一种实用的程序设计工具。另一方面,程序设计是计算机方面的一门基础课程,学习了 C 语言程序设计,进一步学习后续课程都比较方便。C 语言适合(也正在被)作为计算机领域许多后续课程的教学语言。第三,C 语言是一种非常灵活的语言,用 C 语言写程序,人往往需要了解一些实现细节,这是人们对用 C 语言作为基础课程语言的一个主要疑虑。但问题也有另一面,通过对 C 语言程序设计的较好理解,学习者可能对计算机领域的许多问题有更深的认识。这个问题实际上对课程组织、教材和教师提出了较高的要求。此外,用 C 语言写程序,既可以在比较高级的层次上做,也可以在比较低级的层次上做,学生通过这个学习可能更多地了解认识程序设计过程中的各种问题。此外,许多后来的程序语言从 C 语言借鉴了想法和描述方式,有些本身就是 C 语言的扩充和发展,C 语言的知识对于进一步了解掌握许多其他语言,包括未来的新语言都是有价值的。

在撰写本书时,我自己心目中有几个要努力追求的目标,把它们列在这里,供读者和用这本书作为教材的同行参考:

1. 假定读者(学生)原来没有程序设计经验,或者只有很少的经验。因此书中对在学习程

序设计可能遇到的各种基本问题，各种概念和观点都应该尽量给出清楚的解释。这符合目前参加第一门程序设计基础课学习的人的基本情况。这种考虑的目的是希望本书适合作为第一门C程序设计的教材。

2. 以讲授程序和程序设计为基本线索，同时也对C语言各方面特征做深入细致的介绍和解释。本书希望强调的是如何认识程序、写程序，如何用C语言写程序。因此对于从问题出发，经过分析逐步写出程序的过程做了许多深入的讨论。对于程序实例，应当强调分析问题，对问题进行分解，设计求解过程，找出其中主要步骤，确定函数抽象，找出循环，选择所用语言结构，最后写出程序的整个过程。本书对不少实例给出了在不同考虑下可能形成的多种解法，希望能更好地帮助读者理解程序设计的真谛。
3. 应该特别强调好的程序设计风格，强调通过函数抽象分解建立程序清晰结构的重要性。书中很早就开始介绍函数概念，从库函数的使用开始，到简单函数的书写，到函数的确切定义等。书中特别希望强调程序的结构性、可读性、易修改性，给出程序实例时也努力遵循这些原则，使其简洁清楚易读。书中还根据讨论的进展和遇到的问题分析了一些不良程序设计习惯及其危害。
4. 应该特别注意强调"好的"C程序设计及C语言描述方式。由于历史原因C语言成为一个不太严格的语言。如不注意，用C写的程序常常隐含不易发现的错误，这是把C作为第一个语言时需要解决的问题*。在ANSI C标准的基础上，实际上可以形成一套写"好的"C程序的方式。本书力图始终坚持ANSI C所倡导的正确的C程序写法，强调如何写更可靠的、不易包藏隐含错误的C程序的各方面问题，并通过许多实际例子说明了应该如何写和不应该如何写等等。在坚持了上面这些原则的基础上，书中也介绍了C语言的许多实用程序设计技术。总之，本书希望强调的是如何写出正确、清晰、简洁、高效的C程序。
5. 应该对C语言各方面的结构、机制都做较细致介绍，因为这其中的不少问题反应了计算机一些相关领域的普遍性知识和情况。本书力图对许多问题给出较细致的解释，不仅讲了它们是怎样的，还解释了这样规定的背景理由，以期帮助读者理解问题的实质。书中对程序设计和C语言中反映出来的程序设计语言的一般性问题，程序设计中的一般性问题，以及计算机科学中的一般性问题也做了适度的介绍和解释。

本书包括九章和若干附录，章节内容如下：

第一章，程序设计，C语言与C程序。这章首先介绍了程序与程序语言的概念，C语言的发展及其特点。用一个小例子介绍了C程序的形式，C程序的加工和执行。最后介绍程序设计与开发过程，其中的各种问题，如调试、追踪、错误的排除等。

第二章，数据对象与计算。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。

第三章，变量、函数和控制结构。讨论基本程序设计的一些问题，包括基本语句与复合结构，变量的概念和使用，简单函数的定义，程序中逻辑条件的描述与使用等。并进一步讨论了函

---

* 实际上任何语言都有其自身的弱点。计算机工作者有一句很有意思的话："再好的语言也不能阻止人写出坏的程序"。当然这并不是说程序语言不重要，否则为什么人们还在努力开发"更好"的新语言呢？这里我们想说的是，*对任何语言都有如何合理恰当地使用、如何写出好程序的问题*。对于C语言，这方面的问题可能更突出一点，这也是读者在学习中应当特别注意的。

数的递归定义，最后介绍了几种基本的控制结构。

第四章，循环与程序控制。本章首先分析了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。

第五章，C语言程序结构。本章讨论了C语言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等。

第六章，数据对象的组合：数组。介绍数组概念、定义和使用，数组的处理，数组与函数的关系，两维和多维数组等。

第七章，指针。首先介绍指针的概念和指针变量的使用，介绍了C语言中指针与数组的关系等，多维数组作为参数的通用函数等。而后讨论了动态存储管理，类型定义，指向函数的指针等一系列问题。

第八章，结构及其他。介绍了结构(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。随后介绍了链接结构的概念以及C程序的分块开发问题。

第九章，文件与标准库函数。本章首先讨论了文件的概念，以及与输入输出有关的各种问题，随后介绍了标准库提供的各方面功能及其相关知识。

最后有三个附录和一个比较详细的索引。

本书以标准C语言为基础，有关内容并不依赖于任何具体的C语言系统。因此在这个课程的学习(这本书的阅读中)可以使用任何符合ANSI C标准的C语言系统作为程序工作环境。目前国内常用的微机上的各种C语言系统，或者工作站或其他计算机上的C系统都可以用。虽然目前软件厂商在不断推出新版本的程序开发环境，尤其是在竞争比较激烈的微机软件市场上。但是应该看到，这更多的是为了有关厂商自己的市场需要。新版本的开发环境未必比早一些的版本有多大改善，但通常是更复杂、速度也更慢，尤其是对于初学者入门更加困难。由于本书的内容是非常基本的，所有例子程序都按照ANSI C标准书写，习题中不牵涉到任何具体的系统环境。因此本书的学习并不要求复杂的环境支持，建议读者使用比较简单基本的系统作为学习工具，例如国内使用较多的TURBO C 2.0系统等。本书的最后附了对TURBO C 2.0系统的简单使用说明。

这里我要特别感谢北京大学数学学院和北京大学理科试验班参加过我的课程的同学们和参加过课程辅导的研究生们，是他们的思考和提出的问题给了我许多启示，使我更深入地理解了许多问题。我也要感谢我的家人与同事在这些年工作中给我的支持。北京大学出版社的沈承凤老师为本书的出版做了许多深入而细致的工作，在此特别表示感谢。

虽然本书包含了我几年的工作和思考，但书中难免有一些大的或小的错误，这些都只能由我自己承担责任。希望本书的读者能够把发现的问题告诉我，也希望各方面的同行就本书有关的问题提出宝贵意见。

裘宗燕

北京大学数学学院信息科学系，1999年

目　录

第一章　程序设计、C 语言与 C 程序

1.1　程序与程序语言

在开始学习程序设计时，初学者首先遇到的问题可能是："什么是程序"？"什么是程序设计语言"？语言这个词一般用来指人们在生活工作中使用的自然语言，如汉语、英语等。这些自然语言是随着人类的发展和进步而逐渐自然形成的，是人们互相之间交流信息的工具和媒介。人们用口头语言向别人传播消息，传播自己的见闻，表达自己的看法和想法；用书面语言写文章、书籍，以便在更大范围中实现信息交流。程序设计语言则完全不同，这是一类完全人造的语言，是人与计算机打交道时交流信息的一类媒介和工具。程序设计语言也常被称作"编程语言"，本书后面常把它们简称为"程序语言"，在上下文清楚的地方也把它们简称为"语言"。

计算机是人类发明的一种自动机器，它能够完成的工作就是计算；程序设计语言可以看作是人们用来描述计算的工具。有人可能说：从小学开始我就学习数学，数学的一个基本部分是计算，从小学起我们就已经开始用数学的方式（或者说，用数学的"语言"）描述计算过程，程序设计语言难道有什么特殊的地方吗？确实有！程序语言的一个突出特点就在于：不仅人（能够）懂得它，掌握它，能够用它来描述自己所需要的计算过程，而且计算机也可以"懂得"它，可以按照人们用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。所以我们把程序设计语言看作是人与计算机之间"交流"的媒介：人可以用程序语言来书写程序，用这种方式指挥计算机完成各种各样的特定工作，或者说，完成各种各样的计算。

今天，计算机的发展，计算机在各种领域中的广泛应用，计算机对人类社会生活各方面的深刻影响已经是人所共知的事实了。计算机之所以能够产生这样大的影响，其原因不仅在于人发明了并且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、能够指挥计算机完成各种简单或者复杂工作的程序。计算机，作为一种看得见、摸得着的物理实体，其基本工作原理实际上非常简单，目前人们正在使用的计算机也没有多少种。而正是那些数量繁多、功能丰富多彩的程序给了计算机无穷无尽的"生命力"。

计算机的最本质特征就是可以编程序，可以自动地按照程序的指令进行工作。所以，从计算机诞生之初就有了程序设计工作。计算机本身是一种通用的计算机器，加上一个（或者一组）程序后，它就可以转变为一台处理某种专门问题的特殊处理机器。计算机的这种通用性与专用性的统一是非常重要的。这样，一方面，计算机可以通过现代化的大工厂，采用现代化方式大量地生产出来；另一方面，使用同一台计算机，通过运行不同的程序，在不同时候就可以处理不同的问题，甚至同时处理不同的问题。这就是计算机威力的真谛。显然，程序在这个问题上扮演了核心的角色，而这些程序就是程序设计工作的产品。

进行程序设计需要有确定的描述方式，说得更准确一点，需要的是一种计算机能够处理的、意义清晰的，而且人使用起来也比较方便的描述计算的方式。也就是说，需要有好的描述程序的语言。在计算机诞生后的一段时间里，人们需要直接用二进制形式的机器语言描述程序。二进制形式的语言对于人的使用而言非常不方便，用它书写程序非常困难，需要花费许多时

间，不但工作效率极低，而且程序正确性难以保证，即使知道其中有错误也很难辨认和改正。后来，人们发展了符号形式的，人使用起来相对容易一些的汇编语言。用汇编语言书写的程序，计算机不能直接执行，需要用专门的软件（“汇编系统”）进行加工，把这种程序翻译成二进制的计算机指令形式，然后才能在计算机上使用。1954 年诞生的第一个高级程序设计语言，FORTRAN，宣告了程序设计的一个新时代的开始。FORTRAN 语言采用了完全符号化的描述形式，用类似数学表达式的形式描述对数据的计算过程。语言中提供了有类型的变量，作为计算机存储在语言层次上的反映。此外，语言里还提供了高级的程序流程控制机制，如循环和子程序等。这些高级机制使写程序的人可以摆脱程序具体实现的许多细节，使复杂程序的书写方便了许多，写出的程序也更容易阅读，若有错误也更容易辨认和改正。FORTRAN 语言诞生以后受到广泛欢迎。

由于高级程序语言更接近人们所习惯的描述形式，更容易被使用者接受，这使得更多的人能够（并且乐于）参加到程序设计活动中来。人们用高级语言书写程序，工作效率更高，这样就大大推动了计算机应用的发展，开发出了更多的应用系统。应用的发展反过来又推动了计算机工业的大发展。因此，可以说，高级程序设计语言的诞生和发展对于计算机领域发展到今天这样的程度起了非常重要的作用。从 FORTRAN 语言诞生至今，人们已经提出的各种语言超过千种，其中的大部分可以说只是试验性的，只有少数语言得到了广泛使用。随着时代的发展，今天绝大部分程序实际上都是用高级语言书写的，人们也已经习惯于用“程序设计语言”这个词特指各种高级程序语言了。

在计算机科学技术发展过程中，新的程序语言不断被提出，一些老程序语言慢慢地被淘汰。有些老的语言虽然仍然在使用，但它们本身也发生了许多变化。以 FORTRAN 语言为例，这个语言在过去 40 多年时间里经过了四五次大改版，目前最新的版本（FORTRAN 90）与开始的 FORTRAN 语言相比，也早已是面目全非了。对于其他具有较长历史的语言，情况也类似。推动程序语言发展的因素很多，一个重要的原因是人们在程序工作中的新认识。随着程序设计的实践越来越丰富，人们对程序设计这个工作应该怎样做，需要什么样的东西去描述等，不断地产生新的认识。推动程序语言发展的另一个原因是计算机应用的发展。新的应用领域常常对描述工具提出新的要求，这些新认识和新要求促使人们改造已有的语言，或者提出新的语言。

全世界目前使用较广泛的语言主要有 FORTRAN、C、C＋＋、PASCAL、Ada、Java 等，这些语言通常被认为是常规的语言，因为它们有许多共同的性质。另外还有一些语言比较特殊，在形式、编程方式等方面与常规语言差别很大，这些语言互相之间通常也大相径庭。这些非常规的语言一般是各有各的特点或者应用领域，甚至有着特殊的使用人群。属于这类语言的如 LISP、Smalltalk、PROLOG、ML 等等。这些语言虽然不如常规语言使用广泛，但也是非常重要的，它们都曾经在程序语言或者计算机的发展历史上起过（其中有些仍然正在起着）极其重要的作用。

在本章的下面几节里，我们将首先介绍本书讨论程序设计问题时所使用的程序设计语言——C 语言的一些基本情况，并从一个简单的程序例子出发，介绍 C 语言本身以及程序设计过程中的一些重要概念和问题。

1.2 C 语言简介

C 语言是美国贝尔实验室的 Dennis Ritchie 在 1973 年设计的一种程序设计语言，当时的设计目标是用于书写操作系统和其他系统程序。C 语言首先被用来写 UNIX 操作系统，在当时的 PDP-11 计算机上运行。到 70 年代后期，C 语言作为 UNIX 系统上标准的系统开发语言，随着 UNIX 系统的流行而得到越来越广泛的接受和应用。80 年代以后 C 语言被逐渐搬到包括大型机、工作站等在内的其他机型的操作系统上，逐渐成为各种计算机上开发系统程序和复杂软件系统的一种通用语言。

随着微型计算机的蓬勃发展、处理能力的提高和应用的日益广泛，越来越多的人需要从事微机上应用系统的开发工作，人们需要有适合于各种系统软件和应用软件开发的程序语言。C 语言由于其本身各方面的特点，能够比较好地满足人们的这些需要，在微机软件开发方面的应用也就日益广泛起来，目前已经成为应用最广泛的微机系统开发的语言工具之一，被用来开发各种微型机上的各种大小程序，直至非常复杂的软件系统。

对于目前世界上使用最多的以 Intel 及其兼容芯片为基础的微型计算机，已经有了许多性能良好的商品 C 语言系统可以使用。这其中包括：Borland 公司早期的 TURBO C 系列和后续的 Borland C/C++系列产品；Microsoft（微软）公司的 Microsoft C 和后续的 Visual C/C++系列产品。还有一些其他 C/C++语言系统产品，其中使用较广泛的有 Watcom C/C++系列和 Symantic C/C++系列产品等。此外还有一些廉价的和免费的 C 语言系统。其他微型计算机也都有相应的 C 语言系统。目前各种高档工作站都采用 UNIX 操作系统，C 语言是它们标准的系统开发语言。各种大型计算机上也有自己的 C 语言系统。

C 语言之所以能够被世界计算机界广泛接受，正是由于它自身的特点。总体上讲，因为 C 语言在设计上把直到 70 年代为止人们对于程序语言的认识和开发实现复杂系统程序（例如操作系统等）的需要成功地联系在一起。C 语言的主要特点包括：

C 语言的定义比较简单，是一个比较小的语言。这样，学习 C 语言时入门相对较容易，不需要知道多少东西就可以开始编程序。C 语言很好地总结了其他语言提出的子程序库的经验，把许多程序设计中需要的功能放在程序库（在 C 语言里称为“函数库”）里实现，例如程序的输入输出功能等。这样就使语言本身比较简单，编译程序的实现比较容易。人们早已用 C 语言本身写了它自己的编译程序，这种程序很容易移植到各种不同的计算机上，促进了 C 语言的传播。

C 语言提供了丰富的高级程序机制，包括各种程序控制机制和数据定义机制，这些机制能够满足人们构造复杂程序时的各种需要。C 语言提供了功能强大的函数定义和使用机制，使人可以把复杂的程序分解成一个个具有一定独立性的程序片段——函数，这种做法分解了程序系统的复杂性，使之更容易控制和把握。C 语言还提供了一组与硬件操作比较接近的低级操作，使人可以用 C 语言写那些比较低级的、直接与计算机硬件打交道的程序或者程序的组成部分。这种功能使 C 语言在许多情况下被作为汇编语言的高级“替代物”，大大提高开发这些程序的效率。

C 语言提供了一套“预处理”命令，支持对程序或软件系统进行分块开发。有了这些机制，一个软件系统可以比较方便地首先由几个人或几个工作小组分别进行开发，然后把做好的部分集成到一起，构成最后的系统。这种工作方式对于开发大的软件系统是必需的。人们已经用

C 语言开发了许多规模很大的系统。

C 语言的另一个特点是可以写出执行效率很高的程序。人们之所以在许多地方用汇编语言，一个原因就是高级语言写出的程序效率相对比较低。这样，要开发一些效率要求特别高的程序时就只能使用汇编语言。有了 C 语言之后情况发生了变化，目前 C 语言也被广泛应用在这类程序的开发方面。

C 语言的工作得到了全世界计算机界的广泛赞许。一方面，C 语言在程序语言研究领域具有一定价值，由它引出了不少后继语言，还有许多新语言也从 C 语言中汲取了营养，吸收了 C 语言中的不少东西。另一方面，C 语言对整个计算机工业和应用的发展都起了很重要的推动作用。正是由于这些情况，C 语言的设计者获得世界计算机科学技术界的最高奖——图灵奖。

随着 C 语言及其应用的发展，人们也有了更强烈的要求，希望 C 语言能够成为一种更加安全可靠的、既不依赖于具体计算机也不依赖于具体操作系统（如 UNIX）的标准化的程序设计语言。美国国家标准局（ANSI）在 80 年代建立了专门小组研究 C 语言的标准化问题，这个工作的结果是 1988 年颁布的 ANSI C 标准语言。C 语言的这个标准被国际标准化组织和各国标准化机构接受，同样也被采纳为中国国家标准。

关于 ANSI C 语言参考书，除了 ANSI C 语言标准本身外，最重要的参考书之一就是由 C 语言设计者 B. W. Kernighan 和 D. M. Ritchie 合著的《C 程序设计语言》（第二版）。该书英文版已由清华大学出版社作为“大学计算机教育丛书（影印版）”之一出版，这本书可以作为本书学习过程中使用的参考书。在这本书里不仅包括了关于 C 语言程序设计的有关讨论，书后还附了许多有关材料，包括改写的语言参考手册（由于原标准手册是 ANSI 的财产，不能直接抄录）和 ANSI C 标准程序库各方面功能的详细介绍。书中还有一个附录，总结了 ANSI C 与过去的 C 语言之间的所有不同点，供熟悉老式 C 语言的人参考。但是，该书的书写方式实际上假定读者在阅读之前就理解程序设计过程，已有这方面的经验并至少熟悉一种程序语言，因此不适合作为程序设计与 C 语言的初学者使用的教材。但它的确是一本很好的参考书，可惜的是该书至今还没有正式的中文译本。

本书中关于 C 语言的全部讨论都以 ANSI C 标准为依据，在后面各个章节里提出许多使用 C 语言写程序时应当注意的问题，其中一些就是针对 C 语言的过时形式的。应当看到，目前图书市场上仍然有不少书籍是以老的 C 语言形式作为程序例子，这些书籍里有些说法和解释也已经过时。本书里将坚持始终用 ANSI C 标准的新形式写程序，我们这样做是有充分理由的。

在 C 语言诞生之初，设计者的主要想法是把它作为汇编语言的一个替代品，首先是作为设计者本人自己用来写操作系统的语言工具，因此那时更多的是强调灵活性和方便性。这样发展起来的语言对于程序形式的要求很不严格，给写程序的人留下很大余地，使人可以用许多不太合规矩的方式写程序。这实际上就在 C 语言里留下了许多不安全因素，要求写程序的人自己注意可能出现的问题，程序的正确性主要靠写程序的人自己保证，语言的处理系统（编译程序）不能提供多少帮助。随着应用范围越来越广，使用 C 语言的人越来越多（显然，这其中的大部分人对语言的理解远不如设计者清楚），C 语言在这方面的缺点就日益突出起来。由此造成的后果是，人们用 C 语言开发的复杂程序里常常带有隐藏很深的错误，难以发现和改正。建立 ANSI C 标准的一个重要工作就是设法在这些方面对 C 语言做一些弥补和修正。

但是语言的改造是一件很困难的事情。虽然人们已经认识到原来 C 语言中的一些东西是

不好的，最终应该丢掉。但那些已有的东西，主要是在新标准出现之前人们已经开发的各种程序和软件系统，是一笔巨大的财富，绝不能轻易丢掉。要彻底改造这些系统要耗费极大的人力和物力，实际上是不可能做到的。另一方面，已有的一批C语言使用者也养成了使用老形式的习惯，改变这种习惯也不可能在一朝一夕实现。因此，即使想建立一个新标准，也需要保留与原来老形式尽量多的兼容性，ANSI C标准仍然基本上容许原来C程序形式的存在，作为对现实情况的一种让步。但是在另一方面，新标准中更强调：这些老的东西最终将被抛弃，希望写程序的人尽量不要再使用它们。

在这种情况下，今天我们来学习C语言，学习用C语言进行程序设计，那么理所当然地应该采用新的形式，而不应该学习那些旧的过时的形式。这里的原因主要有两个：一是这些旧东西终归将被抛弃，养成使用它的习惯将来还要改，那时将更加费劲，无益地耗费时日，显然很不合理；二是这些过时的东西确实是不好的，虽然有时采用这些形式能使人在写程序时少写几个字符，但那样做的结果往往会阻碍编译系统对程序的检查。由于人是容易犯错误的，在从事写程序这样的复杂工作时尤其是这样，所以阻止编译检查实际上是轻率地拒绝计算机的帮助，这种做法的实际后果无法预料：可能是没有任何影响（如果写出的程序实际上并没有错误）；也可能非常惨重，后来要为程序中实际存在的隐藏错误耗费更多的时间和精力。

实际上，与学习写程序有关的问题不仅包括采用C语言的什么形式，还包括人们在几十年的程序设计工作所总结出的许多经验，包括程序书写形式，许多情况中程序的具体写法等。这里要对阅读本书的读者提出的建议是：在学习程序设计的过程中，不仅要学习程序语言和程序设计的方法，而且应该注意学习如何写程序，还应该注意养成良好的程序设计习惯，这也是学习程序语言与程序设计过程中应当特别重视的一个方面。书中许多地方提出了对这些方面的建议，希望读者一定要重视这些建议。

1.3 一个简单的C程序

从下一章开始，本书将详细地讨论C语言的各种结构，讨论程序设计各个方面的情况和问题。在开始这些讨论之前，我们首先请读者看一个简单的C语言程序的例子，看一看用C语言写出的程序是什么样子。然后从这个例子出发解释程序开发过程中的一些问题。下面是一个简单的C语言程序：

```
#include <stdio.h>

main () {
    printf("good morning! \n");
}
```

用C语言写出来的程序也被简称为“C程序”，上面给出的只是一个非常简单的例子。这个例子程序可以分成两个基本部分：其中第一行是个特殊的行，它说明在这个程序里要用到C语言系统提供的一些标准功能，因此需要参考输入输出函数库文件stdio.h，关于这个方面的情况在第五章有详细介绍。例子中空行之后的三行是程序的基本部分，这部分描述了程序所要完成的工作。注意，我们把写在一对花括号里面的部分看作是下一层次的内容，把它们后退四格写出来，为的是能够反映程序的结构情况。这个程序的意义就是产生一行字符输出“good

morning!",详细情况在下一节讨论。

一个实际的C程序可能比这个例子长得多。一般地说,一个C程序由一系列普通的可打印(可显示)字符构成,组成程序的字符序列通常被按照人读起来方便的形式分为一些行(实际上,就是在字符序列中插进了一些换行符号),每个行的长度不一定相同。这样做的主要目的就是希望程序的表面形式能较好地反映C程序本身各层次的结构。人们一般用普通的编辑器编写程序的内容,或者是使用专门的程序开发系统写程序,修改程序。

C语言被称为一种"自由格式"的语言,除了若干简单限制外,写程序的人完全可以根据自己的想法和需要选择程序的格式,例如在哪里换行,在哪里增加几个空格,等等。这些格式写法上的变化并不影响程序本身的意义。

应该特别指出,C语言对程序格式没有规定并不说明程序的格式不重要。程序的一个重要作用是给人看,首先是写程序的人自己需要看。对于人的阅读而言,程序的格式就显得非常重要。根据多年的程序设计实践,人们在这方面取得了统一的认识:由于一个程序可能很长,程序的结构可能很复杂,因此程序在书写上必须采用良好的格式,所用格式应该很好地体现程序的层次结构,反映程序中各个部分之间的关系。

关于程序的格式问题,人们普遍采用的基本方式是:① 在程序里适当地加入空行,用于分隔程序中处在同一层次的不同部分;② 同一层次的各个不同部分应当互相对齐排列,下一层次的东西通过适当退格(在一行开始处加若干空格)的方式,使程序的结构更容易看清楚;③ 在程序里增加一些必要的说明信息,这方面的情况后面还要介绍。上面的程序例子采用了符合这些看法的书写形式。

对于初学者,从开始学习程序设计时就应当养成注意程序格式的习惯。虽然对刚开始写的小程序,即使没有采用好的格式,自己(或别人)也不难看懂,但对于稍微大一点的程序,情况就很不一样了。有些人在学习中为了一时方便,根本不关心程序的格式,想的只是少输入几个空格或换行,这样做结果是使自己在随后的程序调试检查过程中遇到更多的麻烦。所以,在这里要特别提醒读者:注意程序格式,从一开始写最简单的程序时就要注意养成好习惯。由于目前多数的程序设计语言(包括C语言)都是自由格式语言,这就使人能够方便地根据自己的需要和习惯写出具有良好格式的程序来。

1.4 C程序的加工和执行

C语言是一种高级程序设计语言,人比较容易使用、书写和阅读。用C语言写出的程序通常被称作"源程序",这种程序计算机不能直接执行,实际的计算机只能够识别和执行具有特定二进制代码形式的机器语言程序。为使计算机能够完成某个C语言程序所描述的工作,必须首先把这个源程序(例如上面的简单程序)转换成二进制代码形式的机器语言程序,这种转换过程由C语言系统完成。由C语言源程序到机器语言可执行程序的转换过程称为"C程序的加工",每个C语言系统都包含了加工C语言程序的功能。C程序加工由"编译程序"、"连接程序"等完成,语言系统里还有一些其他相关程序或程序模块。

程序加工通常分成两步完成:在第一步,由编译程序对源程序文件进行分析和处理,生成相应的机器语言目标模块,由目标模块构成的代码文件被称作"目标文件"。这种目标文件还不是可以执行的程序,因为其中缺少每个C程序运行都需要的一个公共部分,这个部分被称为