重点大学计算机教材

HZ BOOKS

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

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作为讨论程序设计的语言，讨论了基本程序设计的各方面问题。书中给出程序实例时没有采用常见的“提出问题，给出解答，再加些解释”的简单三步形式，而是增加了许多对问题的分析和讨论，以帮助读者认识程序设计过程的实质，理解从问题到程序的思考过程。书中还尽可能详尽地解释了许多与C语言和程序设计有关的问题。

本书适合作为高等院校计算机及相关专业的教材，也可供其他学习C程序设计语言的读者阅读。

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. -北京：机械工业出版社，2005.9
（重点大学计算机教材）
ISBN 7-111-16756-2

Ⅰ.从… Ⅱ.裘… Ⅲ.① 程序设计-高等学校-教材 ② C语言-程序设计-高等学校-教材 Ⅳ.① TP311.1 ② TP312

中国版本图书馆CIP数据核字（2005）第070356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）
策划编辑：温莉芳
责任编辑：李云静
北京瑞德印刷有限公司印刷 · 新华书店北京发行所发行
2005年9月第1版第1次印刷
787mm × 1092mm 1/16 · 27印张
印数：0 001 - 5 000册
定价：36.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

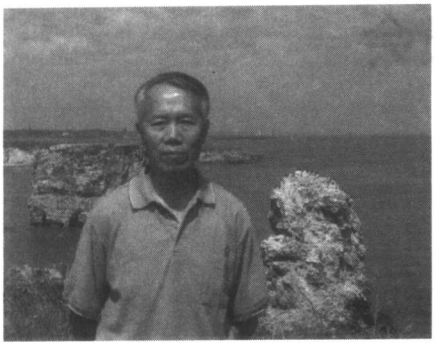
重点大学计算机教材系列

专家指导委员会

（以姓氏拼音为序）

联络人　温莉芳

本书作者简介



裘宗燕，1952年生，北京大学数学学院信息科学系教授。长期从事计算机软件和理论方面的研究和教学工作。专业兴趣领域包括：程序理论，软件形式化方法，复杂程序和系统的形式化模型，程序设计语言（语言特征和语义），程序设计技术，符号计算和计算机科学教育等。近年主持或者参加了多项与本人研究兴趣相关的项目，包括国家自然科学基金项目和973项目。目前主要的研究工作集中在具有复杂状态的程序和系统的语义模型、面向对象系统的理论、Web语言和Web服务的形式化模型以及一般的软件形式化方法领域。

多年来一直从事计算机科学技术课程的教学工作，承担过多门本科生和研究生课程的教学工作，包括：算法与数据结构、程序设计语言原理、计算概论（C语言程序设计）、程序设计技术与方法、算法设计与分析、数理逻辑、理论计算机科学基础、人工智能等。

近年在教学过程中编写了若干本计算机课程教材，包括本书和《C++基本程序设计》、《数据结构——C++和面向对象的途径》（与张乃孝老师合作）等。在工作之余翻译了多部重要计算机科学技术著作，包括《C++程序设计语言（特别版）》、《C++语言的设计和演化》、《程序设计实践》、《从规范出发的程序设计》、《B方法》、《计算机程序的构造和解释》、《程序设计语言——实践之路》等。

个人主页：www.is.pku.edu.cn/~qzy/

Email：qzy@math.pku.edu.cn

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是毋庸置疑的。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，介绍如何理解程序语言，如何学习程序设计，如何为计算机领域中的进一步学习和工作做好准备。

本书以C作为讨论程序设计的语言，讨论了基本程序设计的各方面问题。书中给出程序实例时没有采用常见的“提出问题，给出解答，再加些解释”的简单三步形式，而是增加了许多对问题的分析和讨论，以帮助读者认识程序设计过程的实质，理解从问题到程序的思考过程。书中还尽可能详尽地解释了许多与C语言和程序设计有关的问题。

程序构造过程应该有充分的科学性，并要有对程序实现过程的科学认识。有关研究的发展形成了程序的理论。入门书不可能讨论有关的理论成果，但必须反映其精神实质，使初学者从一开始就看到程序的一些本质性问题。本书特别强调问题的分析和分解，很早就介绍了函数抽象的概念，而后不断有进一步的讨论，实例中也特别注意这方面的问题。书中一些地方还通过实例介绍了一些更深入的理论问题，如通过对程序运行时间的统计，介绍计算过程的基本性质（复杂性）；通过分析循环过程是否完成了所需工作，介绍“循环不变关系”的概念和意义等。当然，这些只是希望帮助读者了解一些情况，作为思考程序问题的线索。

程序设计也是一种工程性的工作，需要分析问题，寻找可能的解决方案，在各种方案中做评价和选择，还要对所做选择有清醒的认识（优点和缺点，是否在某些方面有所偏向或不足）。工程中通常没有完美的选择，更多的是权衡和折中，在程序设计里也是如此。本书的许多实例包含了比较详细的分析过程，常对一个问题给出多个解答，比较它们的优劣，有时还指出其他可能性：还可以如何看问题，还可能如何做，等等。书中还常给读者提出一些问题，希望读者发挥自己的思维能力和主观能动性。各章练习也试图反映这些想法。

本书希望强调一种观点：程序问题并没有需要记住的标准答案。由于分析问题时的不同考虑，设计过程中的不同选择，人们对同一问题会得到许多合理而正确的**不同**程序。这些程序常常各有长短，可能各有侧重，也可能反映了人们对问题的不同认识。在学习中应该特别注意如何分析问题，如何把复杂问题分解为相对简单的部分，如何在可用的功能中做出选择等。这里的每一步都可能产生分支，我们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）给出的程序并不是“金科玉律”，它们不过是作者对问题的理解和分析的结果，实际上还有很多可能性。要学好程序设计，我们应该养成这样的习惯：在看程序时，应该特别注意分析其中隐含着作者的哪些考虑和选择，其中哪些是合理而有价值的（或不合理而无价值的），还可能有什么选择，沿其他选择做下去可能得到什么（或失去什么），等等。这样思考将使我们受益无穷。当然，这并不是说书中的示例不重要。恰恰相反，因为程序设计中有许多选择可能，所以书中应当给出好的分析和选择，供读者参考。入门书籍还应该说明选择

的理由，指出采取这些选择带来的问题（缺点、限制等）。

正如本书的书名所言，程序设计是“从问题到程序”的思考和工作过程。要很好地完成程序设计，我们既要充分发挥聪明才智，又要有细致认真、一丝不苟的工作态度。即使将来不从事程序设计工作，从这一课程中得到的锻炼也可能很重要。

从作为入门语言的角度看，没有一种语言具有无可比拟的优势，使用时都需要考虑其有利方面，也要克服其不利因素。选择C作为入门语言的主要理由有：C是使用最广的语言之一，包含了基本程序设计需要理解的主要机制，能满足讨论程序设计问题的需要。学生可以用它完成练习，得到有关的知识积累和能力锻炼，还能掌握一种实用工具，也能作为后续课程学习的基础。C语言适合（也正在）作为计算机领域许多课程的教学语言。

C是一种很灵活的语言，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。从另一方面看，通过用C做程序设计，也可能得到对程序设计的更多认识。C程序设计可以在比较低的层次上做，也可以在较高层次上做，学生可能从中了解更多有关程序设计过程的问题。此外，许多语言从C借鉴了一些想法和表达形式，有些就是C的扩充和发展，C语言知识对于进一步了解其他语言，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，同时深入介绍了C语言各方面的情况。这里强调的是如何认识程序、写程序和用C写程序。本书通过实例讨论了问题的分析和分解，找出主要步骤，确定函数抽象，找出循环，选择语言结构，最后写出程序的过程。不少实例给出了在不同考虑下可能形成的多种解法，以帮助读者理解程序设计的“真谛”。

书中特别强调好的程序设计风格，强调“好的”C程序设计：反复讨论了通过函数抽象建立清晰结构的重要性，特别关注程序的结构良好、可读、易修改，也尽可能指出了一些不良程序设计习惯及其危害。历史原因使C成为一个不太严格的语言，如果不注意，用C写的程序就有可能隐藏一些不易发现的错误[⊖]。ANSI C标准倡导了一套写“好的”C程序的写法。本书坚持这一正确方向，讨论了如何写出更可靠、不易包藏隐含错误的清晰、简洁、高效的C程序，通过实例说明了应该如何写和不应该如何写。书中还介绍了一些实用的C程序设计技术，详细解释了C语言的各种结构和机制，尽可能地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，并用一个小例子介绍C程序的形式，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3章，变量、函数和控制结构。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，基本程序设计技术。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语言的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程序的结构，函数

⊖ 任何语言都有弱点。有句名言说：“再好的语言也不能阻止人写出坏程序”。这不是说语言不重要，而是说任何语言都有合理使用、写好程序的问题。C语言在这方面的问题突出一点，读者应特别注意。

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，数组。介绍数组的概念、定义和使用，数组与函数的关系，两维和多维数组等。

第7章，指针。首先介绍指针的概念和指针变量的使用，C语言中指针与数组的关系，多维数组作为参数的通用函数，而后讨论动态存储管理、类型定义、指向函数的指针等问题。

第8章，文件和输入输出。讨论文件的概念，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，结构和其他数据机制。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。讨论程序设计及开发中的一般性问题和技术，包括分块开发问题等。

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。

最后有几个附录介绍了C语言的一些相关参考资料。

本书以ANSI标准C语言为背景，书中所有实例均按ANSI C标准书写，习题也不涉及任何具体的系统环境。读者可以用任何符合ANSI C标准的C系统作为编程环境，如国内使用较多的Turbo C系统，微机上可用的公开的免费的lcc、Dev-C++以及其他各种系统。

在此特别感谢北京大学理科试验班和数学学院参加C程序设计课程的同学们和参加辅导工作的研究生们，是他们的思考和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的工作中给予我的支持。

本书曾于1999年在北京大学出版社出版。这一新版对全书做了大幅度修改，反映了近年来我对许多问题的新认识，也使这个新版已经成为了另一本新书。由于机械工业出版社华章分社朋友们的大力支持，使得本书得以面世，我在这里向他们表示特别的谢意。虽然本书凝结着我多年的工作心得和深入的思考，但书中仍难免有或大或小的错误。希望读者能把发现的问题告诉我，也希望同行们对本书提出宝贵意见。

裘宗燕

北京大学数学学院信息科学系，2005年修订

目　录

前言
本书作者简介

第1章　程序设计和C语言 ……1
1.1　程序和程序语言 ……1
1.2　C语言简介 ……8
1.3　一个简单的C程序 ……10
1.4　程序开发过程 ……13
1.5　问题与程序设计 ……17
本章讨论的重要概念 ……18
练习 ……18
第2章　数据对象与计算 ……21
2.1　基本字符、名字表示、标识符和关键字 ……21
2.2　数据与类型 ……23
2.3　基本类型与数据表示 ……23
2.3.1　整数类型和整数的表示 ……23
2.3.2　实数类型和实数的表示 ……25
2.3.3　字符类型和字符的表示 ……26
2.3.4　数据的外部表示、内部表示与转换 ……27
2.4　运算符、表达式与计算 ……30
2.4.1　算术运算符 ……30
2.4.2　算术表达式 ……31
2.4.3　表达式的求值 ……32
2.4.4　计算和类型 ……34
2.5　数学函数库及其使用 ……37
2.5.1　函数、函数调用 ……37
2.5.2　数学函数及其使用 ……38
2.5.3　函数调用中的类型转换 ……39
问题解释 ……40
几个常用程序模式 ……40
本章讨论的重要概念 ……41
练习 ……41
第3章　变量、函数和控制结构 ……43
3.1　语句、复合结构 ……43
3.2　变量——概念、定义和使用 ……44
3.2.1　变量的定义 ……45
3.2.2　变量的赋值与取值 ……46
3.2.3　几个问题 ……48
3.3　定义函数（初步） ……50
3.3.1　函数定义 ……52
3.3.2　函数和程序 ……55
3.3.3　函数与类型 ……56
3.3.4　自定义输出函数 ……57
3.4　关系表达式、逻辑表达式、条件表达式 ……58
3.4.1　关系表达式和条件表达式 ……58
3.4.2　逻辑表达式 ……60
3.5　语句与控制结构 ……61
3.5.1　条件语句（if语句） ……62
3.5.2　循环语句（1）：while语句 ……65
3.5.3　循环语句（2）：for语句 ……68
3.6　若干常用结构和问题 ……70
3.6.1　增量和减量运算符（++、--） ……70
3.6.2　逗号运算符 ……71
3.6.3　实现二元运算符操作的赋值运算符 ……71
3.6.4　空语句 ……72
3.6.5　表达式和求值 ……73
问题解释 ……74
几个常用程序模式 ……74
本章讨论的重要概念 ……75
练习 ……75

第4章 基本程序设计技术 ……79
4.1 循环程序设计 ……79
4.1.1 基本循环方式 ……80
4.1.2 求一系列完全平方数 ……81
4.1.3 判断素数（谓词函数）……82
4.1.4 艰难的旅程（浮点误差）……83
4.1.5 求立方根（迭代和逼进）……85
4.1.6 求sin函数值（通项计算） ……86
4.2 循环程序的问题 ……87
4.2.1 从循环中退出 ……87
4.2.2 循环中的几种变量 ……89
4.3 循环与递归 ……90
4.3.1 阶乘和乘幂（循环，递归）……90
4.3.2 Fibonacci序列（计算与时间）……93
4.3.3 为计算过程计时 ……94
4.3.4 Fibonacci序列的迭代计算（程序正确性与循环不变式）……95
4.3.5 最大公约数 ……97
4.3.6 河内塔（梵塔）问题 ……100
4.4 基本输入输出 ……102
4.4.1 格式输入函数scanf ……102
4.4.2 字符输入输出函数 ……109
4.4.3 输入函数的返回值及其作用 ……112
4.5 控制结构和控制语句 ……114
4.5.1 do-while循环结构 ……114
4.5.2 流程控制语句 ……115
4.5.3 开关语句 ……117
4.6 程序设计实例 ……119
4.6.1 一个简单计算器 ……119
4.6.2 定义枚举常量 ……119
4.6.3 单词计数问题 ……120
4.7 程序测试和排错 ……123
4.7.1 测试 ……123
4.7.2 白箱测试 ……124
4.7.3 黑箱测试 ……125
4.7.4 排除程序里的错误 ……126
问题解释 ……127
几个常用程序模式 ……128
本章讨论的重要概念 ……128
练习 ……128
第5章 C程序结构 ……133
5.1 数值类型 ……133
5.1.1 实数类型和整数类型 ……133
5.1.2 字符类型 ……133
5.1.3 整数类型 ……134
5.1.4 基本数据类型的选择 ……135
5.2 函数和标准库函数 ……136
5.2.1 C语言的库函数 ……137
5.2.2 字符分类函数 ……137
5.2.3 随机数生成函数 ……138
5.3 函数定义和程序的函数分解 ……139
5.3.1 主函数 ……140
5.3.2 程序的函数分解 ……141
5.3.3 对函数的两种观点 ……142
5.3.4 函数原型 ……146
5.4 C程序结构与变量 ……149
5.4.1 外部定义的变量 ……150
5.4.2 作用域与生存期 ……151
5.4.3 外部变量和自动变量 ……151
5.4.4 变量定义的嵌套 ……153
5.4.5 静态局部变量 ……154
5.4.6 变量的其他问题 ……155
5.4.7 一个实例 ……157
5.5 预处理 ……159
5.5.1 文件包含命令 ……159
5.5.2 宏定义与宏替换 ……160
5.5.3 条件编译命令 ……163
5.6 定义常量 ……164
5.7 字位运算符 ……165
5.8 编程实例 ……168
5.8.1 一个简单的猜数游戏 ……168
5.8.2 加密与解密 ……170
本章讨论的重要概念 ……172
练习 ……172

第6章 数组 ……175
6.1 数组的概念、定义和使用 ……175
6.1.1 定义数组变量 ……176
6.1.2 数组的使用 ……177
6.1.3 数组的初始化 ……179
6.1.4 数组的存储实现 ……180
6.2 数组程序实例 ……181
6.2.1 从字符到下标 ……181
6.2.2 筛法求素数 ……182
6.2.3 成绩分类 ……183
6.2.4 多项式求值 ……184
6.2.5 定义数组的问题 ……185
6.3 数组作为函数参数 ……186
6.3.1 一个例子 ……186
6.3.2 修改实参数组的元素 ……188
6.4 字符数组与字符串 ……189
6.4.1 字符数组 ……189
6.4.2 字符串 ……189
6.4.3 程序实例 ……191
6.4.4 标准库字符串处理函数 ……193
6.4.5 输出文本里的最长行 ……194
6.5 两维和多维数组 ……198
6.5.1 多维数组的初始化 ……198
6.5.2 多维数组的表示和使用 ……199
6.5.3 多维数组作为函数的参数 ……200
6.6 编程实例 ……201
6.6.1 成绩直方图 ……201
6.6.2 一个通用带检查的整数输入函数 ……205
6.6.3 “计算”数组变量的大小 ……206
6.6.4 统计C程序里的关键字 ……208
6.6.5 数组的划分 ……211
6.6.6 数组的排序 ……213
问题解释 ……215
几个常用程序模式 ……215
本章讨论的重要概念 ……215
练习 ……215
第7章 指针 ……217
7.1 地址与指针 ……217
7.2 指针变量的定义和使用 ……218
7.2.1 指针操作 ……218
7.2.2 指针作为函数的参数 ……219
7.2.3 与指针有关的一些问题 ……222
7.3 指针与数组 ……224
7.3.1 指向数组元素的指针 ……224
7.3.2 基于指针运算的数组程序设计 ……227
7.3.3 数组参数与指针 ……228
7.3.4 指针与数组操作的程序实例 ……229
7.3.5 字符指针与字符数组 ……231
7.4 指针数组 ……233
7.4.1 指针数组与两维数组 ……234
7.4.2 命令行参数及其处理 ……235
7.5 多维数组作为参数的通用函数 ……238
7.6 动态存储管理 ……240
7.6.1 为什么需要动态存储管理 ……240
7.6.2 C语言的动态存储管理机制 ……241
7.6.3 两个程序实例 ……244
7.6.4 函数、指针和动态存储 ……247
7.7 定义类型 ……250
7.7.1 定义数组类型和指针类型 ……251
7.7.2 复杂类型描述与解读 ……252
7.8 指向函数的指针 ……254
7.8.1 函数指针的定义和使用 ……254
7.8.2 函数指针作为函数的参数 ……255
7.8.3 数值积分函数 ……257
7.8.4 若干以函数指针为参数的数组操作实用函数 ……259
几个常用程序模式 ……260
本章讨论的重要概念 ……260
练习 ……261
第8章 文件和输入输出 ……263
8.1 文件的概念 ……263
8.1.1 流和文件指针 ……263
8.1.2 缓冲式输入输出 ……264
8.2 文件的使用 ……265

8.2.1 文件的打开和关闭……265
8.2.2 输入输出函数……267
8.2.3 程序实例……268
8.2.4 直接输入输出函数……270
8.3 标准流输入输出与格式控制……271
8.3.1 行式输入和输出……271
8.3.2 输入格式控制……272
8.3.3 输出格式控制……276
8.3.4 以字符串作为格式化输入输出对象……278
8.3.5 标准错误流……278
8.4 程序实例……279
8.4.1 求文件数据的平均值……279
8.4.2 一个背单词程序……281
8.4.3 资金账目系统……285
练习……288
第9章 结构和其他数据机制……291
9.1 结构（struct）……291
9.1.1 结构声明与变量定义……291
9.1.2 结构变量的初始化和使用……296
9.1.3 结构、数组与指针……297
9.2 结构与函数……299
9.2.1 处理结构的函数……299
9.2.2 程序实例……302
9.3 联合（union）……306
9.4 枚举（enum）……308
9.5 编程实例……310
9.5.1 数据组的排序……310
9.5.2 复数的表示和处理……312
9.6 链接结构（自引用结构）……315
9.6.1 链接结构……315
9.6.2 自引用结构的定义……317
9.6.3 程序实现……318
9.6.4 数据与查找……321
9.7 字段……322
问题解释……323
本章讨论的重要概念……323
练习……323
第10章 程序开发技术……327
10.1 分别编译和C程序的分块开发……327
10.1.1 分块开发的问题和方法……327
10.1.2 程序实例：学生成绩处理……328
10.1.3 分块重整……333
10.1.4 其他安排和考虑……336
10.1.5 模块化思想和技术……338
10.1.6 单一头文件结构和多个头文件结构…342
10.2 功能模块和程序库……345
10.2.1 复数模块……345
10.2.2 目标文件和库……348
10.2.3 防止重复包含……349
10.3 错误报告和处理……349
10.3.1 建立统一的错误报告机制……349
10.3.2 定义变参数的错误报告函数……350
10.3.3 运行中错误的检查和处理……352
10.4 程序的配置……358
10.4.1 程序的行为参数和启动时的配置…358
10.4.2 交互式配置……360
10.4.3 通过命令行参数……361
10.4.4 采用配置文件……362
10.5 程序开发过程……362
10.5.1 自上而下的开发……363
10.5.2 自下而上的开发……365
10.5.3 实际开发过程……365
练习……367
第11章 标准库……371
11.1 标准库结构……371
11.1.1 标准定义（<stddef.h>）……372
11.1.2 错误信息（<errno.h>）……372
11.2 几个已经介绍过的头文件……373
11.2.1 数学函数（<math.h>）……373
11.2.2 字符处理函数（<ctype.h>）……374
11.3 字符串函数（<string.h>）……375
11.3.1 一些字符串函数……375
11.3.2 存储区操作……378

11.4 功能函数（<stdlib.h>）……379
11.4.1 几个整数函数 ……380
11.4.2 数值转换 ……380
11.4.3 执行控制 ……381
11.4.4 与执行环境交互 ……381
11.4.5 常用函数bsearch和qsort ……382
11.5 日期和时间（<time.h>）……383
11.6 实现特征（<limits.h>和<float.h>）……385
11.6.1 整数类型特征 ……385
11.6.2 浮点数类型特征 ……386
11.7 其他与输入输出有关的函数（<stdio.h>）……386
11.7.1 符号常量和类型 ……386
11.7.2 文件操作函数 ……387
11.7.3 流缓冲区操作函数 ……388
11.7.4 文件定位及定位函数 ……389
11.7.5 其他有关函数 ……390
11.7.6 采用va_list参数的输出函数 ……391
11.8 定义变长度参数表（<stdarg.h>）……392
11.9 非局部控制转移（<setjmp.h>）……395
11.10 调试断言和信号处理（<assert.h>和<signal.h>）……397
11.11 标准库的其他功能 ……398
11.11.1 本地化 ……398
11.11.2 多字节字符 ……400
本章讨论的重要概念 ……400
练习 ……400
附录A C语言运算符表 ……401
附录B C语言速查 ……403
附录C C99简介 ……409
进一步学习的建议 ……413
参考文献 ……416

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”？本章首先讨论这方面的问题，帮助读者在比较直观的基础上建立起对程序、程序设计、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

程序一词也来自生活，通常指完成某项事务的一套既定活动方式或者活动过程。从表述方面看，我们可以把程序看成对一系列动作的执行过程的描述。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其中各个基本步骤的序列。如果按顺序实施这些步骤的动作，其整体效果就完成了该项事务或者工作。

现在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（进一步查找其他参考书的书目）
   5.2.放弃借书，离开图书馆
6. （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得多。可以看到，这一程序不是一个平铺直叙的动作序列，其中步骤更多，有时还需要根据当时遇到的情况处理，以及可能出现的重复动作。

如果仔细探究这个实例，还可以认识到这一程序可能需要进一步细化。例如读者可能找出

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；填写好索书单后已经到了下班时间；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。

由这些现实生活中的例子，我们可以初步看到程序的一些直观特征。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够理解和直接完成的。例如，在上面有关借书的程序描述中，我们把“查书目”作为一个基本动作。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”动作进一步细分，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：

从计算机进入图书馆的书目检索系统；
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，程序就需要进一步分解。这种逐步细化或者分解的过程，也是下面有关计算机程序设计的讨论中是最本质的东西。

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（无论是不是人）需要按照程序的描述执行一系列的动作。在达到结束位置时工作就完成了。

本书中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

日常生活中程序性活动的情况与计算机里的程序执行很相似。这一情况可以帮助我们理解计算机的活动方式。当然，在人们日常生活中的程序性工作中有更多变数，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的、一丝不苟的，计算机将一步步按程序中的指令办事，一点“商量”的余地也没有。

计算机是人类发明的一种自动机器，它能完成的工作就是“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，把一个数从这里搬到那里，比较两个数的大小等。为使计算机能按人的指挥工作，每种计算机都提供了一套指令，每种指令对应计算机能执行的一个基本动作。所谓的计算机程序，也就是一系列的这种指令。

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。因此，人与计算机打交道的基本方式就是把要求计算机执行的程序提供给它，而后命令它去执行这个程序。此后，计算机就会按照程序的规定，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之后，它就变为处理某个专门问

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人所共知的事实了。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

由上面的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。计算机有着预先确定的基本动作，每个动作只能完成很少的一点点工作。如果需要用计算机去处理复杂的问题，就可能需要写出很长很长的程序，而且必须精确地描述执行所有动作的细节过程，不能有一点错误，也不能有一点含糊其辞之处。这些也就是程序设计之所以困难的根源所在。

要说明一个程序在执行中需要做什么，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工作中使用的自然语言，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，需要有一种适宜的描述程序的语言。可供人编程序用的语言就是*程序设计语言*，这是一类人造的语言。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。需要存入计算机，要求计算机去执行的程序也不例外。这种计算机可以直接执行的二进制程序形式，称为*机器语言*的程序。在计算机诞生之初人们只能直接用机器语言写程序。对于人的使用而言，二进制的机器语言很不方

便，用它书写程序非常困难，工作效率极低，程序的正确性难以保证。发现了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:

```
0000000010000000001000      -- 将单元1000的数据装入寄存器0
0000000010001000001010      -- 将单元1010的数据装入寄存器1
0000010100000000000001      -- 将寄存器1的数据乘到寄存器0的原有数据上
0000000010001000001100      -- 将单元1100的数据装入寄存器1
0000010000000000000001      -- 将寄存器1的数据加到寄存器0的原有数据上
0000001000000000001110      -- 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（这里的符号a、b、c分别代表地址为1000、1010和1100的存储单元），而后将结果保存到单元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，在二进制机器指令的层面上理解一个复杂程序到底做了什么，很容易变成人力所根本不能及的事情。为缓解这一问题，人们发展出符号形式的、使用相对容易一些的汇编语言。用汇编语言写的程序需要用专门软件（汇编系统）加工，翻译成二进制机器指令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出的程序，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的原有数据上
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的原有数据上
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。但是，汇编语言的程序完全没有高层次的结构，只是许多基本的汇编语言指令堆积形成的长长序列，是一团散沙。复杂程序仍然难以开发，难以理解，程序中的错误难以检查。

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对数据的计算。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，语言里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的容易把握的部分，在工作中可摆脱许多具体细节，方便复杂程序的书写，写出的程序也更容易阅读，有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

高级程序语言更接近人习惯的描述形式，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。用高级语言书写程序的工作效率更高，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。应用的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，其中的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书写的，人