掌握了语法，为什么还是不会编程？

翁惠玉教授携手计蒜客在线教育团队，教你掌握C编程！

第一行代码

C语言

视频讲解版

翁惠玉 ◎ 编著

由计蒜客提供在线学习支持

- 强调**算法**、**抽象**等重要的**程序设计思想**，培养初学者的**计算思维**。
- 录制了**94**个**微视频**，总时长超过**1000**分钟，对**重难点**知识、**典型**例题进行讲解，扫描书中**二维码**即可在线观看，也可通过**光盘**本地学习。
- 书中讲解了**171**道例题和**228**个程序样例，并提供了**118**道自测题和**141**道编程题，自测题的答案见附录，编程题的答案见光盘。
- 提供辅助教学的**资源包**，包括实验、试卷及答案、教学大纲、PPT等。

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

第一行代码

C语言

视频讲解版

翁惠玉 ◎ 编著

由计蒜客提供在线学习支持

人民邮电出版社

北京

图书在版编目（CIP）数据

第一行代码：C语言：视频讲解版 / 翁惠玉 编著. -- 北京：人民邮电出版社，2018.5
ISBN 978-7-115-47593-0

Ⅰ. ①第… Ⅱ. ①翁… Ⅲ. ①C语言－程序设计
Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2017)第322159号

内 容 提 要

本书以C语言为编程环境，由浅入深地介绍了C语言的完整内容以及过程化程序设计的思想和方法。全书共有13章。第1章介绍了什么是程序设计。第2章给出了一个完整的C语言程序，并介绍了如何在VS2010中输入、编译链接及调试程序。第3～5章分别介绍了C语言中支持结构化程序设计的3种结构：顺序、分支和循环所必需的工具。第6章介绍了如何编写及应用函数；第7章介绍了处理批量数据的工具，即数组。上述章节的内容都是C语言的核心知识，请务必掌握。第8～11章分别讲解了结构体、共用体、链表、位运算和文件等高级编程技术。第12章讲解了如何用结构化程序设计思想指导一个大程序的开发，以及软件开发的基本过程。该章中用“猜硬币”游戏介绍了自顶向下分解的过程，用“石头、剪刀、布”游戏介绍了模块划分，用“龟兔赛跑模拟”的例子介绍了如何建立一个自己的库以及如何应用自己创建的库，用学生管理系统和书店管理系统讲述了软件开发的过程。第13章介绍了通用算法设计技术，旨在让读者了解，当遇到一个问题时应该如何设计解决问题的算法。

本书内容翔实、讲解深入，每个知识点都提供了示例，全书共有171道例题和228个程序样例，所有程序样例都在VS2010中调试通过。为了方便读者自学，本书还提供了118道自测题和141道编程题，以及所有习题的答案，且配套了讲解主要知识点的视频。

本书适合C语言初学者，也适合有一定基础的读者。可作为各高等院校计算机专业的教材，也可供从事计算机软件开发的人员参考。

◆ 编　　著　翁惠玉
责任编辑　税梦玲
责任印制　焦志炜
◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市潮河印业有限公司印刷
◆ 开本：800×1000　1/16
印张：30.5　　2018年5月第1版
字数：742千字　　2018年5月河北第1次印刷

定价：69.80元（附光盘）

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字20170147号

前言

翻开这本书，你将开启对计算机科学的探索之旅。计算机科学这门学科已成为当今最具生气和活力的学科之一，它使很多领域中看似不可能的事情成为可能。互联网、无人机、AlphaGo、电子支付等曾经在科幻电影中出现的技术，现在已经融入了人们的日常生活，然而这些技术都离不开计算机程序设计。

要想了解计算机是如何工作的，并且学会如何编写计算机程序（以下简称“编程”），不可能一蹴而就，必须循序渐进和不断实践。对大多数人来说，迈出第一步可能是较难的。多数人能够熟练地应用许多现成的软件，但一开始并不相信自己也能开发软件。其实，编程并不需要高深的数学和电子学知识，核心在于能找到解决问题的思路。要想做到这一点，必须以逻辑方式考虑问题，训练自己以计算机能够理解的方式去表达自己的逻辑，即“计算思维”。本书的编写目的之一，也是希望能培养初学者的计算思维。

初学者遇到的问题

学习程序设计，初学者经常会遇到“一个问题”和“两个错误”。“一个问题”就是“书读懂了，但不会编程”，遇到的编程题不知该如何下手；“两个错误”就是犯了“只学不做”和“死抠语法”的毛病。

1. 不会编程

不会编程可能存在以下两种情况。

第一种情况：知道如何解决问题，但不会通过计算机来解决问题，即不会编写解决问题的程序。

第二种情况：虽然掌握了程序设计的基本语法，但编程者不具备解决某个特定问题的能力。

第一种情况是因为缺乏计算思维，而学习程序设计就是训练和培养计算思维。通过编程可以解决很多问题，但计算机能够提供的基本功能只有算术运算和逻辑运算。因此，要编写出解决某个问题的程序，必须将解决问题的过程分解成一系列的算术运算和逻辑运算，然后把这一系列的运算表示成程序设计语言的语句。如果需要解决的是数学问题，则分解过程比较容易；如果需要解决的问题是非数学问题，则分解过程就比较困难了。要形成计算思维，只有勤学多练。勤学就是多阅读别人的程序；多练就是多编程，初学者可以先编写解决数学计算的程序，从这些程序中熟悉程序设计语言，熟悉编写计算机程序的过程。本书中提供了大量的解决计算问题的程序，请大家熟悉了编程语言和编程过程后，再开始着手编写一些简单的非数值计算的程序，如字符串处理等。俗话说，熟读唐诗三百首，不会写诗也会凑。编程也是如此，大家可以多阅读书中提供的程序，也可多读一些标准程序，学习他人解决问题的方法和编程技巧。

第二种情况与有没有学好程序设计是没有关系的。要编写解决某个问题的程序，自己应该要能解决这个问题，然后才能把解决问题的过程分解成计算机能够完成的基本动作，设计出解决问题的算法。计算机科学是一门需要终生学习的学科，要想编写解决某个问题的程序，就需要先学习如何解决这个问题。

2. 只学不做

有些人在学习程序设计时，把书读了一遍又一遍，视频也听了一遍又一遍，书上都是密密麻麻的笔记，但就是不动手编程。这种学习方法是不可取的。

程序设计是知识，更是技能。学习程序设计一定要动手编程。编程的过程就是思维训练的过程，语言的很多奥妙之处也只有在编程的过程中才能深刻体会。程序编写好之后，不是自己感觉正确就可以了，一定要经过调试（debug）。如果编译器显示成功生成了目标代码，就说明程序设计的语法是正确的，接下来，当程序运行出预期的结果时，才能说明程序的逻辑是正确的。调试需要多年的经验积累。刚开始学习调试时，即使是简单的语法错误，即使编译器已经明确告诉你错误在哪儿、错误是什么，你可能还是完全搞不懂为什么错了，应该如何修改。当程序输出的结果没有达到你的预期，或是程序执行异常终止时，你也许都不知道为什么会出现这种情况。因此，学习编程就必须动手，当成功地编写了一个又一个的程序时，你离高手也就越来越近了。

3. 死抠语法

程序设计的目的是让计算机去解决某个问题，所以学习程序设计的重点在于设计出解决问题的方法。程序设计语言只是描述问题解决过程的一种工具，因此，你可以使用 C 语言、Python 语言、Java 语言等。每种程序设计语言都有自己的语法，但它们提供的功能是大同小异的。所以，在学习编程时，不要死抠语言的语法，而是要训练解决问题的思维方法，还要尽量使用简单明了的语句。例如，经常会有学生问到“printf("%d　%d", ++x, x++);”语句的输出结果，或者“x+++y”的值是什么？语句是你给计算机输入的一个命令，你完全可以给计算机一个清晰明了的命令，为什么要用“x+++y”这种命令呢？

如何使用本书进行学习

本书以介绍基本的程序设计思想、概念和方法为基础，强调了算法、抽象等重要的程序设计思想，并选择 C 语言作为编程语言。C 语言是业界非常流行的语言，它使用灵活、功能强大，可以很好地体现程序设计的思想和方法。同时，C 语言既具有低级语言的特性，又具有高级语言的特性，常用来编写系统软件或应用软件。笔者根据多年来在上海交通大学计算机系讲授“程序设计”课程的经验，并参考了近年来国内外主要的程序设计图书的基础上，编写了本书，希望能对初学者有所帮助，下面介绍如何使用本书进行学习。

1. 内容安排模块化，可先易后难

本书内容分为 5 个部分。

■ 第 1 部分：包含第 1～2 章，讲解程序设计所需要的基础知识及所用的开发环境。

■ 第 2 部分：包含第 3～6 章，讲解 C 语言中支持结构化程序设计的 3 种结构——顺序结构、分支结构和循环结构所需要的工具。

■ 第 3 部分：包含第 7～11 章，讲解 C 语言的一些高级工具，如数组、指针、结构体、链表、共用

体、位运算与位段、文件等。

■ 第 4 部分：包含第 12 章，讲解软件开发过程及相关技术。

■ 第 5 部分：包含第 13 章，讲解通用算法设计，即遇到一个问题时应该如何设计解决问题的算法。

书中内容覆盖面较广，虽然存在一定难度，但是某些难度较大的内容可以先略过（较难的、可以忽略的章节前面都用“*”标注），且不会影响整个知识的连贯性。如果你是一位初学者，可以先学习第一～三部分，并略过其中带*号的章节，这三部分内容可以帮助你熟悉程序设计语言和培养计算思维，等到能够较熟练地掌握程序设计及 C 语言后，再回过头来学习其中带*号的部分，以进一步加深对程序设计的理解，从而编写质量更高的程序。学完前三部分内容后，再根据自己的实际情况来选择学习后面两个部分的内容。如果想做一些工程性的项目，可以继续学习第四部分；如果重点在做研究，设计解决某个问题的算法，可以继续学习第五部分。

2. 理解计算机运行机制、掌握算法

书中在讲解知识点时，先讲解其基本用法，然后通过一系列的示例来加深对知识点的理解，最后介绍一些计算机内部的处理过程（如各类数据在计算机中的表示、函数的调用过程、变量的作用域、递归的处理等）。大家在学习时，既要掌握基础知识，还要尽可能地去理解计算机的运行机制，这对训练计算思维，编写高质量的程序，理解程序中的某些错误的出现原因很有帮助。

编写程序的基础是算法设计。当遇到一个问题时，如何着手设计算法呢？书中介绍了常用的算法设计方法，包括枚举法、贪婪法、分治法、动态规划和回溯法等，请大家务必掌握，工作中遇到的很多问题都可以通过这些常用算法来解决。另外，本书也引用了计算学科中的汉诺塔、“八皇后”问题等经典问题，这对解决实际工作中出现的问题也有很大帮助。

3. 程序编写要规范，培养良好的编程习惯

要想编写解决简单问题的程序并不难，但要想编写出工程化的、较复杂的程序，就必须掌握软件工程。书中介绍了结构化程序设计思想、软件的开发过程及如何保证程序正确和提高可维护性的一些方法，并通过 5 个示例讲解了结构化程序设计思想的应用及软件开发的完整过程。

在进行实际项目开发时，代码数量是繁杂的，且程序也需要不断地进行更新和维护。因此，在编写程序时，应该要培养工程化的意识，编写的代码要具有可维护性。书中对程序为什么要加注释，为什么要定义符号常量，为什么要用函数，以及如何提取函数等都做了解释，还给出了变量/函数的命名、程序的排版、常用语句的组合等规范，就是为了让大家能够养成良好的程序编写习惯。同时，在每一章都设置了“编程规范和常见问题”小节，请大家用心学习。

4. 纸上得来终觉浅，绝知此事要躬行

没有谁天生就会编程，每个程序设计的高手都会与代码“共度”无数个不眠之夜，痛并快乐着。要想成为程序设计高手，还有漫长的路需要走。第一步，可以先阅读他人的程序，学习他人解决问题的思想，

然后想想自己可以怎么去解决问题；反之也可，先自己思考，再去查看他人的思路。为此，书中给出了大量的例题和程序样例（171 道例题和 228 个程序样例，所有程序样例都在 VS2010 中调试通过），以帮助大家进行第一步练习。第二步，自己动手编写程序，可以动手编写例题程序，也可以完成书中每章预留的编程题（共 141 道，编程题的代码都在随书配套的光盘中，建议大家先独立解决，只有实在想不出解决办法时再去查看代码），同时，本书还设置了 118 道自测题，用以检查对知识点的掌握程度，请大家认真对待，自测题答案可在本书附录中查看。

5. 利用好微视频、光盘资源以及在线学习平台

如果你是初学者，那么在自学的过程中，肯定会遇到无数个想不明白的问题；如果你有一定基础，也会存在对知识点理解不够透彻的问题；如果你是程序设计的高手，那就请帮助身边的“菜鸟”吧！我作为一名高校教师，希望能够帮助到选用本书进行学习的读者，为此，我用心录制了微视频，挑选了一些知识点进行讲解。因时间精力有限，这个版本完成了 94 个微视频的录制，后续我会继续录制，并在图书重印或再版时补充。大家可以扫描书中的二维码进行在线查看，也可以通过光盘本地查看。光盘中还提供了书中所有程序样例的源代码、实战训练的答案，以及试卷、实验指导、教学大纲、PPT 等资源，以便于教师教学。另外，计蒜客为本书提供了在线学习的支持服务，购买了本书的读者，可在计蒜客平台中开通专属的 C 语言程序设计在线课程，采用伴随式的方式来学习 C 语言编程，并可从计蒜客平台的题库中选择题目进行练习。

致谢

本书得以顺利地编写完成和出版，我要感谢上海交通大学电信学院程序设计课程组的各位老师，我们经常在一起讨论，使我能不断加深对程序设计的理解；我还要感谢那些可爱的学生们，他们与我在课上和课后的互动，使我知道了他们的困惑和学习难点。另外，感谢计蒜客为本书读者提供在线学习 C 语言的平台支持。

若读者在学习过程中遇到困惑，也可以通过邮件 hyweng@sjtu.edu.cn 与我联系。

编　者

2018 年 1 月

目 录

第1章 程序设计概述 1

1.1 什么是程序设计 1
1.2 计算机的基本组成 2
1.2.1 计算机硬件 2
1.2.2 计算机软件 4
1.3 程序设计语言 4
1.3.1 机器语言 4
1.3.2 汇编语言 5
1.3.3 高级语言 6
1.3.4 智能语言 7
1.3.5 C语言 7
1.4 程序设计过程 7
1.4.1 算法设计 8
1.4.2 编码 11
1.4.3 编译与链接 11
1.4.4 调试与维护 12
1.5 编程规范及常见问题 13
1.5.1 真的需要算法设计阶段吗 13
1.5.2 为什么不用自然语言编程 13
1.5.3 寄存器、主存储器和外存储器有什么不同 13
1.5.4 所有的计算机能够执行的指令都是相同的吗 13
1.5.5 为什么需要编译和链接 13
1.5.6 为什么在不同类型的计算机上运行C语言程序需要使用不同的编译器 14
1.5.7 为什么不同类型的计算机不能运行同一个汇编程序 14
1.6 小结 14
1.7 自测题 14
1.8 实战训练 15

第2章 初识C语言 16

2.1 一个完整的C语言程序 16
2.1.1 注释 17
2.1.2 预编译 17
2.1.3 主程序 18
2.2 C语言的开发环境 20
2.2.1 VS2010的安装 20
2.2.2 程序输入 20
2.2.3 编译链接 24
2.2.4 程序的运行 25
2.2.5 程序的调试 28
2.3 编程规范及常见问题 30
2.3.1 注意注释 30

2.3.2 良好的排版习惯 30
2.3.3 为什么要学 C 语言 31
2.3.4 如何学习程序设计 31
2.3.5 什么是库 31
2.4 小结 32
2.5 自测题 32
2.6 实战训练 32

第 3 章 顺序程序设计 33

3.1 常量与变量 33
3.1.1 变量定义 33
3.1.2 数据类型 35
3.1.3 常量与符号常量 39
3.2 数据的输入/输出 44
3.2.1 字符的输入/输出 44
3.2.2 格式化输入/输出 46
3.3 算术运算 54
3.3.1 算术表达式 54
3.3.2 不同类型数据间的混合运算 56
3.3.3 强制类型转换 57
3.3.4 数学函数库 59
3.4 赋值运算 60
3.4.1 赋值表达式 60
3.4.2 赋值的嵌套 62
3.4.3 复合赋值运算 62
3.4.4 自增和自减运算符 63
*3.5 信息表示 64
3.5.1 数制间的转换 64
3.5.2 整数的表示 66
3.5.3 实数的表示 69
3.6 顺序程序设计示例 69
3.7 程序规范及常见问题 71
3.7.1 变量命名 71
3.7.2 运算符的优先级 71
3.7.3 数据运算时的注意事项 71
3.7.4 为什么要定义符号常量 72
3.7.5 变量定义后且对它赋值前的值是什么 72
3.7.6 不要在表达式中插入有副作用的子表达式 72
3.8 小结 72
3.9 自测题 73
3.10 实战训练 74

第 4 章 分支程序设计 76

4.1 关系表达式 76
4.2 逻辑表达式 78
4.2.1 逻辑运算 78
4.2.2 短路求值 80
4.3 if 语句 82
4.3.1 if 语句的形式 82
4.3.2 if 语句的嵌套 87
4.3.3 条件表达式 90
4.4 switch 语句及其应用 92
4.5 程序规范及常见问题 102
4.5.1 条件语句程序的排版 102
4.5.2 不要连用关系运算符 102
4.5.3 注意短路求值 102
4.5.4 常见错误 102

4.6 小结 103
4.7 自测题 103
4.8 实战训练 104

第5章 循环程序设计 106

5.1 计数循环 106
5.1.1 for 语句 106
5.1.2 for 语句的进一步讨论 115
5.1.3 for 循环的嵌套 116
5.2 break 和 continue 语句 118
5.2.1 break 语句 118
5.2.2 continue 语句 120
5.3 基于哨兵的循环 121
5.3.1 while 语句 122
5.3.2 do…while 循环 128
5.4 循环的中途退出 130
5.5 编程规范和常见问题 134
5.5.1 循环语句程序的排版 134
5.5.2 优化循环体 134
5.5.3 使用 for 循环的注意事项 134
5.5.4 常见错误 134
5.5.5 三个循环语句之间的关系 135
5.6 小结 135
5.7 自测题 135
5.8 实战训练 136

第6章 过程封装——函数 139

6.1 函数的定义 140
6.1.1 函数的基本结构 140
6.1.2 return 语句 141
6.1.3 函数示例 141
6.2 函数的使用 144
6.2.1 函数原型的声明 144
6.2.2 函数调用 146
6.2.3 将函数与主程序放在一起 147
6.2.4 函数调用过程 151
6.3 带参数的宏 154
6.4 变量的作用域 155
6.4.1 局部变量 155
6.4.2 全局变量 156
6.5 变量的存储类别 158
6.5.1 自动变量 158
6.5.2 静态变量 159
6.5.3 寄存器变量 160
6.5.4 外部变量 161
*6.6 多源文件程序的编译链接 163
6.7 递归程序设计 164
6.7.1 递归的基本概念 165
6.7.2 递归函数的应用 167
6.8 编程规范及常见问题 173
6.8.1 使用函数时的建议 173
6.8.2 函数命名 174
6.8.3 没有返回值的函数是否需要 return 语句 174
6.8.4 尽量避免使用全局变量 174
6.8.5 尽量避免实际参数表达式有副作用 175
6.8.6 常见错误 175
6.9 小结 175
6.10 自测题 176

6.11 实战训练 177

第 7 章 批量数据处理——数组 179

7.1 一维数组 179
7.1.1 一维数组的定义 179
7.1.2 数组元素的引用 180
7.1.3 一维数组的内存映像 184
7.1.4 一维数组的应用 185
7.2 数组作为函数的参数 190
7.3 查找算法 195
7.3.1 顺序查找 196
7.3.2 二分查找 197
7.4 排序算法 201
7.4.1 直接选择排序法 201
7.4.2 冒泡排序法 203
7.5 二维数组 205
7.5.1 二维数组的定义 206
7.5.2 二维数组元素的引用 207
7.5.3 二维数组的内存映像 208
7.5.4 二维数组的应用 208
7.5.5 二维数组作为函数的参数 218
7.6 字符串 220
7.6.1 字符串的存储及初始化 220
7.6.2 字符串的输入/输出 221
7.6.3 字符串作为函数参数 225
7.6.4 字符串处理函数 230
7.6.5 字符串的应用 232
7.7 程序规范及常见问题 234
7.7.1 数组下标必须从 0 开始吗 234
7.7.2 能用表达式 des = src 将字符串 src 赋给字符串 des 吗 234
7.7.3 为什么存放字符串的数组长度比字符串的实际长度多一个字符 235
7.7.4 有了 scanf 函数为什么还需要 gets 函数 235
7.7.5 传递字符串为什么只需要一个参数 235
7.7.6 传递二维数组时形式参数中第二个方括号中的值为什么必须指定 235
7.8 小结 235
7.9 自测题 236
7.10 实战训练 237

第 8 章 指针 240

8.1 指针的概念 240
8.1.1 指针与间接访问 240
8.1.2 指针变量的定义 241
8.1.3 指针变量的操作 242
8.2 指针与数组 246
8.2.1 指向数组元素的指针 246
8.2.2 指针运算与数组访问 246
8.3 指针与函数 250
8.3.1 指针作为参数 250
8.3.2 返回指针的函数 254
8.3.3 数组作为函数参数的进一步讨论 256
8.4 动态内存分配 259
8.4.1 动态变量 259
8.4.2 动态变量的创建 259

8.4.3 动态变量的消亡 261
8.4.4 内存泄露 261
8.4.5 查找 malloc 和 calloc 的失误 261
8.4.6 动态变量应用 263
8.5 指针与字符串 265
8.5.1 用指向字符的指针变量表示字符串 265
8.5.2 字符串作为函数的参数 266
8.5.3 返回字符串的函数 268
8.6 指针数组与多级指针 270
8.6.1 指针数组 270
*8.6.2 main 函数的参数 272
*8.6.3 多级指针 276
*8.6.4 二维数组与指向一维数组的指针 277
*8.6.5 动态二维数组 279
*8.7 函数指针 280
8.7.1 指向函数的指针 280
8.7.2 函数指针作为函数参数 281
8.7.3 函数指针用于菜单选择 284
8.8 编程规范与常见问题 285
8.8.1 int x, *p = &x;有错吗 285
8.8.2 避免使用悬空指针和未初始化的指针 286
8.8.3 不同类型的指针之间为什么不能赋值 286
8.8.4 指针与数组等价吗 286
8.8.5 值传递和指针传递的区别是什么 286
8.8.6 返回指针的函数必须确保返回值指向的变量在函数执行结束时依然存在 287
8.8.7 使用动态变量时必须严格防止内存泄露 287
8.9 小结 287
8.10 自测题 288
8.11 实战训练 289

第 9 章 更多的数据类型 290

9.1 枚举类型 290
9.2 类型别名 293
9.3 结构体 294
9.3.1 结构体的概念 294
9.3.2 结构体类型的定义 295
9.3.3 结构体类型变量的定义 296
9.3.4 结构体类型变量的使用 299
9.3.5 结构体与函数 302
9.4 链表 308
9.4.1 链表的概念 308
9.4.2 单链表的存储 310
9.4.3 单链表的操作 311
9.4.4 带头结点的单链表 312
9.4.5 单链表实例 313
9.5 共用体 316
9.5.1 共用体概念和共用体类型的定义 316
9.5.2 共用体类型变量的定义及初始化 318
9.5.3 共用体变量的使用 318

9.6 编程规范及常见问题 324
9.6.1 结构体中每个字段的类型都不相同吗 324
9.6.2 单链表中为什么要引入头结点 324
9.6.3 引入结构体有什么用处 324
9.6.4 结构体和共用体的区别 324
9.6.5 结构体和共用体类型定义时能否省略类型名 324
9.6.6 结构体类型定义与结构体变量定义 325
9.7 小结 325
9.8 自测题 325
9.9 实战训练 326
第 10 章 位运算与位段 328
10.1 位运算 328
10.1.1 “按位与”运算 328
10.1.2 “按位或”运算 330
10.1.3 “按位异或”运算 331
10.1.4 “按位取反”运算 333
10.1.5 “左移”运算 335
10.1.6 “右移”运算 335
10.1.7 位运算与赋值运算 337
10.1.8 不同长度的数据进行位运算 337
10.2 位段 338
10.2.1 位段的概念及定义 338
10.2.2 位段的引用 339
10.3 编程规范及常见问题 341
10.3.1 检验某数中指定位的值 341
10.3.2 将数据中的某一位的值置成 0 341
10.3.3 将数据中的某一位的值置成 1 342
10.3.4 将数据中的某一位的值取反 342
10.4 小结 342
10.5 自测题 342
10.6 实战训练 343
第 11 章 文件 344
11.1 内存与外存 344
11.2 文件的概念 345
11.2.1 什么是文件 345
11.2.2 ASCII 文件与二进制文件 346
11.3 文件缓冲与文件指针 346
11.4 文件的打开与关闭 347
11.4.1 打开文件 347
11.4.2 关闭文件 349
11.5 ASCII 文件的读写 349
11.5.1 字符读写函数 350
11.5.2 字符串读写函数 352
11.5.3 数值读写函数 354
11.6 二进制文件的读写 358
11.6.1 fwrite 函数 358
11.6.2 fread 函数 360
11.7 文件的顺序访问 362
11.7.1 什么是文件的顺序访问 362
11.7.2 feof 函数 363

11.8 文件的随机访问 364
11.8.1 文件定位指针 365
11.8.2 rewind 函数 365
11.8.3 fseek 函数 367
11.8.4 ftell 函数 369
*11.9 文件操作与控制台操作 370
11.10 编程规范及常见问题 371
11.10.1 良好的文件使用习惯 371
11.10.2 文件打开方式选择 372
11.10.3 文件指针与文件定位指针 372
11.10.4 流与文件 372
11.11 小结 372
11.12 自测题 373
11.13 实战训练 373

第 12 章 软件开发过程 374

12.1 结构化程序设计思想 374
12.2 自顶向下分解示例："猜硬币"游戏 375
12.2.1 顶层分解 375
12.2.2 prn_instruction 函数的实现 376
12.2.3 play 函数的实现 376
12.2.4 get_call_from_user 函数的实现 378
12.3 模块划分示例："石头、剪刀、布"游戏 380
12.3.1 自顶向下分解 381
12.3.2 模块划分 382
12.3.3 头文件的设计 383
12.3.4 模块实现 386
12.4 设计自己的库示例：随机函数库的设计和实现 389
12.4.1 随机函数库的功能设计 390
12.4.2 接口文件的设计 390
12.4.3 实现文件的设计 391
12.5 随机函数库的应用示例：模拟龟兔赛跑 392
12.5.1 自顶向下分解 392
12.5.2 模块划分及实现 393
12.6 软件开发过程 395
12.6.1 软件危机 395
12.6.2 软件工程 396
12.7 软件开发过程示例：学生管理系统的设计与实现 397
12.7.1 需求分析 397
12.7.2 概要设计 398
12.7.3 详细设计 400
12.7.4 编码与测试 402
12.8 软件开发示例：网上书店的设计 407
12.8.1 需求分析 407
12.8.2 概要设计 408
12.8.3 详细设计 411
12.9 编程规范及常见问题 413
12.9.1 头文件的格式 413
12.9.2 实现一个库为什么需要两个文件 413
12.9.3 慎用全局变量 414
12.10 小结 414
12.11 自测题 414

12.12 实战训练 414

第 13 章 通用算法设计 416

13.1 枚举法 416
13.2 贪婪法 423
13.3 分治法 427
13.4 动态规划 431
13.5 回溯法 435
13.6 小结 442
13.7 实战训练 442

附录 443

附录 1 第 1 章自测题答案 443
附录 2 第 2 章自测题答案 445
附录 3 第 3 章自测题答案 446
附录 4 第 4 章自测题答案 451
附录 5 第 5 章自测题答案 453
附录 6 第 6 章自测题答案 455
附录 7 第 7 章自测题答案 459
附录 8 第 8 章自测题答案 463
附录 9 第 9 章自测题答案 466
附录 10 第 10 章自测题答案 467
附录 11 第 11 章自测题答案 470
附录 12 第 12 章自测题答案 472
附录 13 ASCII 编码表 474

第1章

程序设计概述

自从第一台计算机问世以来，计算机技术发展得非常迅速，特别是微型计算机的出现，使得计算机的应用从早期单纯的数学计算发展到处理各种媒体的信息。计算机本身也从象牙塔进入了千家万户。编写程序也不再是象牙塔中的工作，而是各个专业的技术人员都需要具备的技能。

本章将介绍一些计算机和程序设计的背景知识，具体包括：

- 什么是程序设计；
- 计算机的基本组成；
- 程序设计语言；
- 程序设计过程。

1.1 什么是程序设计

程序设计就是教会计算机如何去完成某一特定的任务，即编写出完成某个任务的正确的程序。学习程序设计就是学习当老师，你的学生就是计算机。老师上课前先要备课，然后再去上课，最后检查学生的学习情况是否达到了预期效果。对应于这三个阶段，程序设计也包括三步：第一步是算法设计，第二步是编码，第三步是编译与调试。

什么是程序设计

上课前首先要知道学生的知识背景，然后才能有的放矢地去教，学习程序设计首先也要了解计算机能做什么。备课就是把所要教授的知识用学生能够理解的方式表达出来。算法设计也就是把解决问题的过程分解成一系列计算机能够完成的基本动作。上课是把备课的内容用某种学生能够理解的语言描述出来。如果给中国学生讲课，就把备课的内容用中文讲出来。如果给美国学生讲课，就把备课的内容用英文讲出来。编码阶段也是如此，如果你的计算机支持 C 语言，就把算法用 C 语言表示出来；如果支持 Pascal 语言，就用 Pascal 语言描述。算法中的每一步都能与程序设计语言的某个语句相对应。上完课后要检查教学的效果，如果没有达到预期的结果，需要检查备课或上课中哪个环节出了问题，修改这些问题，重新再试。同样，编码后要运行程序，检查程序的结果是否符合预期的效果，如果没有，则需要检查算法和程序代码，找出问题所在，修改程序，然后重新运行。

为此，在学习程序设计之前，需要先了解一下我们的学生——计算机的基本功能，然后研究如何教会它各种新的技能。

1.2 计算机的基本组成

计算机系统由硬件和软件两部分组成。硬件是计算机的物理构成，是计算机的物质基础，是看得见摸得着的，相当于人的躯体。计算机软件是计算机程序及相关文档，是计算机的灵魂，相当于人们具备的知识与技能。

1.2.1 计算机硬件

我们常用的计算机称为个人计算机，它通常有一个键盘、一个显示器和一个主机，如图 1-1 所示。

打开主机，可以发现里面有主板、CPU、内存条等。通常计算机的硬件由 5 大部分组成：运算器、控制器、存储器、输入设备和输出设备，这些部分通过总线或其他设备互相连接，如图 1-2 所示。这 5 大部分协同完成计算任务。在现代计算机系统中，运算器和控制器通常集成在一块称为 CPU（中央处理器）的芯片上。这个硬件的结构是由计算机的鼻祖冯·诺依曼（图 1-3）提出的，因此被称为冯·诺依曼体系结构。

图 1-1 个人计算机

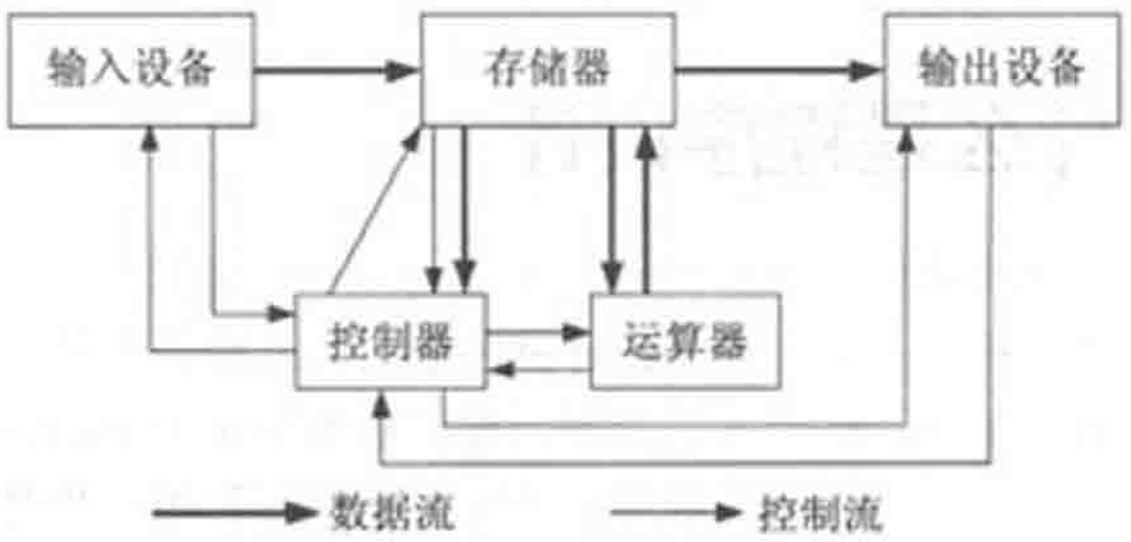

图 1-2 计算机硬件系统的组成

图 1-3 冯·诺依曼