



普通高等教育“十一五”国家级规划教材



21世纪大学本科  
计算机专业系列教材

李文新  
郭 炜 编著  
余华山  
李晓明 主审

# 程序设计导引及在线实践 (第2版)

<http://www.tup.com.cn>

- 根据教育部“高等学校计算机科学与技术专业规范”组织编写
- 与美国 ACM 和 IEEE CS *Computing Curricula* 最新进展同步
- 国家精品课程教材



清华大学出版社



普通高等教育“十一五”国家级规划教材

21世纪大学本科计算机专业系列教材

国家精品课程教材

# 程序设计导引及在线实践 (第2版)

李文新 郭炜 余华山 编著



清华大学出版社  
北京

## 内 容 简 介

本书是北京大学“程序设计实习”课程的内容和“北京大学程序在线评测系统”(POJ)的紧密结合,具有极强的实践性。本书的例题和习题精选自 POJ 题库,并且在叙述中穿插了许多精心编写的思考题,总结了学生在程序设计中易犯的错误。本书的作者均有丰富的工程软件开发经验和教学经验,因此本书中的程序代码均保持良好的风格。

本书可以作为高等学校理工科相关专业程序设计类课程的教材,也可作为以 ACM/ICPC 为代表的大学生程序设计竞赛的培训教材,还可供对程序设计感兴趣的读者学习参考。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13501256678 13801310933

## 图书在版编目(CIP)数据

程序设计导引及在线实践/李文新,郭炜,余华山编著. —2 版. —北京:清华大学出版社,2017  
(21 世纪大学本科计算机专业系列教材)  
ISBN 978-7-302-45234-8

I. ①程… II. ①李… ②郭… ③余… III. ①C 语言—程序设计—高等学校—教材 IV. ①TP312

中国版本图书馆 CIP 数据核字(2016)第 264024 号

责任编辑:张瑞庆

责任校对:李建庄

责任印制:何 芊

出版发行:清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址:北京清华大学学研大厦 A 座

邮 编:100084

社总机:010-62770175

邮 购:010-62786544

投稿与读者服务:010-62776969, [c-service@tup.tsinghua.edu.cn](mailto:c-service@tup.tsinghua.edu.cn)

质量反馈:010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

课件下载: <http://www.tup.com.cn>, 010-62795954

印 刷 者:三河市君旺印务有限公司

装 订 者:三河市新茂装订有限公司

经 销:全国新华书店

开 本:185mm×260mm 印 张:18

字 数:433 千字

版 次:2007 年 11 月第 1 版 2017 年 1 月第 2 版

印 次:2017 年 1 月第 1 次印刷

印 数:1~2000

定 价:35.00 元

产品编号:068170-01

## 21 世纪大学本科计算机专业系列教材编委会

主 任：李晓明

副 主 任：蒋宗礼 卢先和

委 员：（按姓氏笔画为序）

|     |     |     |     |     |
|-----|-----|-----|-----|-----|
| 马华东 | 马殿富 | 王志英 | 王晓东 | 宁 洪 |
| 刘 辰 | 孙茂松 | 李仁发 | 李文新 | 杨 波 |
| 吴朝晖 | 何炎祥 | 宋方敏 | 张 莉 | 金 海 |
| 周兴社 | 孟祥旭 | 袁晓洁 | 钱乐秋 | 黄国兴 |
| 曾 明 | 廖明宏 |     |     |     |

秘 书：张瑞庆

本书主审：李晓明

# 本书序

## PREFACE

本书是一本与众不同的程序设计入门教材,实践性极强,不论对于高等学校计算机专业的学生,还是非计算机专业的学生,都非常适用。

目前绝大部分程序设计入门教材的主要内容就是详细介绍一门程序设计语言,这对于高等学校计算机专业的学生是远远不够的;对于非计算机专业的学生也略显肤浅。许多大学本科计算机专业的课程设置,在程序设计语言和数据结构这两门课之间,并无空间进行基础算法的教学,这就容易导致学生由于基本技能缺失而在学习数据结构课程时产生困难,或难以学精。对于非计算机专业的学生来说,如果仅掌握一门程序设计语言的语法规则,写几个打印由星号组成的三角形之类的“玩具”程序,而对计算机科学的基础与灵魂——算法一无所知,不明白计算机到底是怎么解决问题的,那么在日后的工作中,不但不能自己编写实用程序,甚至不能敏感地及时意识到哪些问题适合用计算机处理,可以交给计算机专业人士来做。本书将程序设计语言和最基本的算法思想相结合,能够有效避免上述现象。

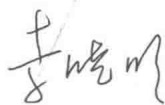
本书的最大特点是和“北京大学程序在线评测系统”紧密结合,具有极强的实践性。“北京大学程序在线评测系统”(Peking University Online Judge System, POJ)是一个免费的公益性网上程序设计题库,网址为 <http://acm.pku.edu.cn/JudgeOnline>(注意这里的网址区分大小写)。它包含 2000 多道饶有趣味的程序设计题目,题目大部分来自 ACM/ICPC 国际大学生程序设计竞赛,很多题目就反映工作和生活中的实际问题。这些题目有易有难,比如最简单的题 A+B Problem 就是给出两个数,输出它们的和。用户可以针对某个题目编写程序并提交,POJ 会自动判定程序的对错。本书的所有例题和课后习题大都精选自 POJ 题库,难度较低,学生做习题时可以将自己的程序提交给 POJ,几秒钟之内即可知道是对还是错。作为教学支持,每位学生在 POJ 上可以建立自己的账号,教师在 POJ 上一眼就能看到学生是否已经完成布置的习题,这几乎将教师评判学生作业的工作量减少到零。POJ 对于程序的正确性评判是极为严格的,学生的程序根据 POJ 给出的输入数据进行计算并输出结果,POJ 在服务器端编译、运行学生提交的程序,取得输出结果和标准答案对比,必须一个字节都不差,程序才能够通过。这对于培养严谨、周密的程序设计作风极为有效,学生必须考虑到每一个细节和特殊边界条件,而不是大体上正确就能够通过。传统的人工评判是难以做到这一点的。

本书的另一特点是在叙述中穿插了许多精心编制的思考题,特别适合教师进行启发式教学。思考题没有答案,以便教师引导学生进行讨论。

本书还有一个亮点,就是在许多例题后都会总结学生在完成该题时容易犯的典型错误,让学生少走弯路。这些错误都总结自学生在 POJ 上提交的程序,因而具有典型性。

本书中代码的风格也很值得一提。许多程序设计教程,其编写者虽有丰富的教学经验,但却不一定具有实际的软件开发经历,因而书中的例子程序往往在变量命名、代码效率等程序设计风格方面不是很在意,只求正确即可,教学代码的痕迹明显。而本书的作者除了具有多年的教学经验以外,还从事过多年的软件开发。李文新教授是国内第一个自主研制的地理信息系统开发环境 Geo-Union 的主要设计者和核心代码编写者之一,曾经担任过图原空间信息技术有限公司和长天科技有限公司的总工程师。她目前是中国计算机学会信息学奥林匹克竞赛科学委员会的科学委员,是 ACM/ICPC 竞赛北京大学代表队的原任教练和现任领队。余华山副教授多年来一直从事支持高性能计算的程序开发与运行环境的研究工作,是集群并行程序开发与运行平台 P\_HPF 系统的主要研制者之一,主持开发了计算网格协同平台 Harmonia 系统。在中国教育科研网格 China Grid 公共软件支撑平台 CGSP 的研制过程中,他是总体设计的主要负责人之一,并负责 CGSP 信息服务系统的设计和实现。郭炜老师的专业研究方向是计算机辅助教学,他独立开发了《我爱背单词》等系列著名英语学习软件,同时还担任教练,和李文新教授一起率领北京大学 ACM/ICPC 国际大学生程序设计竞赛队在国际竞赛中取得了较好名次。本书中的例子程序的代码风格优美、注释完备、可读性强,以此作为范例,对培养良好的程序设计风格,日后在团队开发中赢得同事的信任和喜爱十分有益。

在这个提倡创新的年代,本书是特别富有创意的,希望并相信读者能够喜欢。



北京大学教授  
原教育部高等学校计算机科学与技术教学指导委员会副主任  
21 世纪大学本科计算机专业系列教材编委会主任



计算机程序是通过在计算机内存中开辟一块存储空间,并用一个语句序列不断修改这块存储空间上的内容,最终得到问题的答案的方法来解决实际问题的。计算机程序一般需要用一种具体的程序设计语言表达出来。一种计算机语言通过定义变量的形式给出了申请内存的方式,并通过表达式和赋值语句给出了对内存中的数据进行运算和修改的方法,通过分支和循环语句提供了用不同方式安排语句序列的能力。大部分计算机语言还提供了基础函数库来完成一些常用的计算和数据处理的功能。

使用计算机程序解决实际问题,首先要能够将一个具体问题抽象成一个可计算的问题,并找出可行的计算过程;其次是掌握一门程序设计语言,将设计的计算过程写成具体的代码在机器上运行。

作者总结了多年计算机程序设计类课程的教学经验,认为在程序设计课程的教学应该把握 5 个基本的教学环节:第一,让学生充分理解计算机程序在内存中的运行原理和过程。在程序运行过程中任意时刻都清楚语句运行到了哪里,以及当前存储数据的内存区的内容是什么。只有清楚这些,才能在程序调试过程中及时地找到出错位置,并修改错误,最终让程序按照设计者的意图执行。第二,以一门高级程序设计语言为例,让学生了解该设计语言使用哪些语句定义变量,哪些语句修改变量,变量有哪些基本类型,每种类型的变量占多大的存储空间,不同类型的变量可以进行哪些运算,哪些语句用来控制语句序列的分支和循环,如何用简单变量组合出复杂变量(如数组或结构体),如何控制复杂的计算过程(如通过函数实现分而治之),有哪些库函数是可用的,等等。第三,讲授一些常用的、基本的计算过程,使得学生在解决复杂问题之前,手上有一些可用的基本方法。例如,如何通过分支和循环语句模拟一个手工计算的过程,进行不同数制转换时可以选定一个共同的基数进行转换,字符串处理的问题应该多使用库函数,处理日期问题时可以用一个数组来存储每个月的天数,这样可以很方便地处理不规则的数据,等等。第四,围绕一些具体的问题实例,让学生学会通过分析问题抽象出数学模型,从而设计出计算过程和中间数据的存储方式,最终实现代码并调试成功。学生只有通过这样一个完整的程序设计过程的训练,才能充分理解写程序是要干什么,并且学会判断什么样的问题适合用计算机来解决。第五,学生学习效果的检验方式直接决定了最终的教学效果。如果想让学生真正学会独立动手写出正确的程序,就必须采取上机考查的方式,要求学生针对实际问题写出最终可以正确运行并能解决问题的程序。

本书的内容安排充分体现了上述的教学理念。为了方便理解例题中的代码,本书先用

1/3 的篇幅简明扼要地介绍 C/C++ 语言的基本语法,包括变量的定义,变量的值的修改,基本的变量类型,用基本类型的变量构造数组、结构体等复杂的数据类型,定义表达式,控制语句序列,以及常用的 C 语言标准库函数。

之后所有的内容都采用以问题为中心的讲述方式。首先用近 1/3 的篇幅讲述面对不同类型的常见问题,应该如何抽象计算过程,并将计算过程写成具体代码。这些问题包括简单计算问题、数制转换问题、字符串处理问题、日期和时间处理问题、计算过程模拟问题等。

接着用近 1/4 的篇幅讲述了计算机程序中常用的但不同于数学计算方法的三种算法思想:枚举、递归和动态规划。

本书的最后两章讲述了如何用基本的数据类型构造一些稍微复杂的数据结构:链表和二叉树,作为本书向数据结构递进的序曲。

配合本书的教学,我们使用了北京大学在线评测系统,书中所有的例题和练习题都在该系统上,学生可以随时针对某一题目编写程序并提交给系统,几秒钟内就可以获得正确与否的回答。我们也利用该系统进行学生的期中、期末考试,学生必须现场在给定的时间内完成从问题分析到代码实现的全部过程才能通过考试。为了测试程序在不同数据输入下的正确性,该系统中的题目大部分采用输入多组测试数据的形式,所以在书中会看到每个程序都要读入多组数据进行处理。这些测试数据是彼此独立的,可以读入一组,处理一组并输出结果,然后再读入下一组。

本书作者分工如下:李文新编写第 1 章中的 1.1、1.2、1.4、1.7、1.8、1.9 节,第 2 章,第 5 章,第 9 章中的 9.3、9.4、9.6、9.10 节,以及附录 A 和附录 B。郭炜编写第 1 章中的 1.3、1.5、1.6、1.10~1.19 节,第 6 章,第 7 章,第 9 章中的 9.1、9.2、9.5、9.7、9.8 和 9.9 节,以及第 10 章。余华山编写第 3 章、第 4 章、第 8 章、第 11 章和第 12 章。

由于水平和精力所限,书中难免存在不当之处,恳请专家和读者批评指正。

作者

2016 年 9 月于燕园



# 目 录

## CONTENTS

|                         |    |
|-------------------------|----|
| 第 1 章 C/C++ 语言概述 .....  | 1  |
| 1.1 程序的基本框架 .....       | 1  |
| 1.2 变量 .....            | 2  |
| 1.2.1 变量的定义 .....       | 2  |
| 1.2.2 变量的赋值 .....       | 3  |
| 1.2.3 变量的引用 .....       | 3  |
| 1.3 C/C++ 语言的数据类型 ..... | 3  |
| 1.4 常量 .....            | 5  |
| 1.5 运算符和表达式 .....       | 6  |
| 1.5.1 算术运算符 .....       | 6  |
| 1.5.2 赋值运算符 .....       | 8  |
| 1.5.3 关系运算符 .....       | 8  |
| 1.5.4 逻辑运算符 .....       | 9  |
| 1.5.5 位运算符 .....        | 9  |
| 1.5.6 sizeof 运算符 .....  | 12 |
| 1.5.7 类型强制转换运算符 .....   | 13 |
| 1.5.8 运算符的优先级 .....     | 13 |
| 1.6 注释 .....            | 14 |
| 1.7 分支语句 .....          | 15 |
| 1.7.1 if 语句 .....       | 15 |
| 1.7.2 switch 语句 .....   | 17 |
| 1.8 循环语句 .....          | 19 |
| 1.8.1 for 语句 .....      | 19 |
| 1.8.2 while 语句 .....    | 20 |
| 1.8.3 do-while 语句 ..... | 20 |
| 1.8.4 break 语句 .....    | 21 |
| 1.8.5 continue 语句 ..... | 21 |
| 1.9 函数 .....            | 22 |

|         |                                  |    |
|---------|----------------------------------|----|
| 1.9.1   | 函数的定义 .....                      | 22 |
| 1.9.2   | 函数的调用 .....                      | 23 |
| 1.9.3   | 参数传递和返回值 .....                   | 24 |
| 1.9.4   | 库函数和头文件 .....                    | 25 |
| 1.10    | 标准输入输出 .....                     | 25 |
| 1.10.1  | printf 函数(标准输出函数) .....          | 25 |
| 1.10.2  | scanf 函数(标准输入函数) .....           | 26 |
| 1.11    | 全局变量和局部变量 .....                  | 27 |
| 1.12    | 数组 .....                         | 28 |
| 1.12.1  | 一维数组 .....                       | 28 |
| 1.12.2  | 二维数组 .....                       | 30 |
| 1.12.3  | 数组的初始化 .....                     | 31 |
| 1.12.4  | 数组越界 .....                       | 32 |
| 1.13    | 字符串 .....                        | 33 |
| 1.13.1  | 字符串常量 .....                      | 33 |
| 1.13.2  | 用字符数组存放的字符串 .....                | 34 |
| 1.14    | 指针 .....                         | 36 |
| 1.14.1  | 指针的基本概念 .....                    | 36 |
| 1.14.2  | 指针运算 .....                       | 39 |
| 1.14.3  | 空指针 .....                        | 40 |
| 1.14.4  | 指向指针的指针 .....                    | 41 |
| 1.14.5  | 指针和数组 .....                      | 41 |
| 1.14.6  | 字符串和指针 .....                     | 43 |
| 1.14.7  | void 指针 .....                    | 44 |
| 1.14.8  | 函数指针 .....                       | 45 |
| 1.14.9  | 指针和动态内存分配 .....                  | 47 |
| 1.14.10 | 误用无效指针 .....                     | 50 |
| 1.15    | 结构 .....                         | 50 |
| 1.15.1  | “结构”的概念 .....                    | 50 |
| 1.15.2  | 结构变量的定义 .....                    | 51 |
| 1.15.3  | 访问结构变量的成员变量 .....                | 52 |
| 1.15.4  | 结构变量的初始化 .....                   | 52 |
| 1.15.5  | 结构数组 .....                       | 52 |
| 1.15.6  | 指向结构变量的指针 .....                  | 53 |
| 1.15.7  | 动态分配结构变量和结构数组 .....              | 55 |
| 1.16    | 文件读写 .....                       | 55 |
| 1.16.1  | 用 fopen 打开文件 .....               | 56 |
| 1.16.2  | 用 fclose 关闭文件 .....              | 56 |
| 1.16.3  | 用 fscanf 读文件,用 fprintf 写文件 ..... | 57 |

|              |                                 |            |
|--------------|---------------------------------|------------|
| 1.16.4       | 用 fgetc 读文件,用 fputc 写文件 .....   | 58         |
| 1.16.5       | 用 fgets 函数读文件,fputs 函数写文件 ..... | 59         |
| 1.16.6       | 用 fread 读文件,用 fwrite 写文件 .....  | 60         |
| 1.16.7       | 用 fseek 改变文件读写的当前位置 .....       | 63         |
| 1.17         | C 语言标准库函数 .....                 | 64         |
| 1.17.1       | 数学函数 .....                      | 64         |
| 1.17.2       | 字符处理函数 .....                    | 65         |
| 1.17.3       | 字符串处理和内存操作函数 .....              | 65         |
| 1.17.4       | 字符串转换函数 .....                   | 66         |
| 1.18         | 命令行参数 .....                     | 66         |
| 1.19         | C/C++ 编码规范 .....                | 67         |
| 1.19.1       | 标识符命名注意事项 .....                 | 68         |
| 1.19.2       | 程序的书写格式 .....                   | 68         |
| 1.19.3       | 注释的写法 .....                     | 70         |
| 1.19.4       | 一些好的编程习惯 .....                  | 71         |
| <b>第 2 章</b> | <b>简单计算题 .....</b>              | <b>73</b>  |
| 2.1          | 例题:鸡兔同笼 .....                   | 73         |
| 2.2          | 例题:棋盘上的距离 .....                 | 74         |
| 2.3          | 例题:校门外的树 .....                  | 77         |
| 2.4          | 例题:填词 .....                     | 78         |
| 2.5          | 例题:装箱问题 .....                   | 80         |
|              | 练习题 .....                       | 82         |
| <b>第 3 章</b> | <b>数制转换问题 .....</b>             | <b>85</b>  |
| 3.1          | 相邻数字的基数等比:确定进制 .....            | 85         |
| 3.2          | 相邻数字的基数不等比:skew 数 .....         | 87         |
|              | 练习题 .....                       | 89         |
| <b>第 4 章</b> | <b>字符串处理 .....</b>              | <b>90</b>  |
| 4.1          | 简单的字符串操作示例 .....                | 90         |
| 4.2          | 例题:统计字符数 .....                  | 91         |
| 4.3          | 例题:487-3279 .....               | 93         |
| 4.4          | 例题:子串 .....                     | 96         |
| 4.5          | 例题:Caesar 密码 .....              | 99         |
|              | 练习题 .....                       | 101        |
| <b>第 5 章</b> | <b>日期和时间处理 .....</b>            | <b>104</b> |
| 5.1          | 例题:判断闰年 .....                   | 104        |

|            |                           |            |
|------------|---------------------------|------------|
| 5.2        | 例题: 细菌繁殖 .....            | 107        |
| 5.3        | 例题: 日历问题 .....            | 112        |
| 5.4        | 例题: 玛雅历 .....             | 113        |
| 5.5        | 例题: 时区间时间的转换 .....        | 116        |
|            | 练习题 .....                 | 120        |
| <b>第6章</b> | <b>模拟 .....</b>           | <b>121</b> |
| 6.1        | 例题: 约瑟夫问题 .....           | 121        |
| 6.2        | 例题: 花生问题 .....            | 123        |
| 6.3        | 例题: 显示器 .....             | 126        |
| 6.4        | 例题: 排列 .....              | 130        |
|            | 练习题 .....                 | 133        |
| <b>第7章</b> | <b>高精度计算 .....</b>        | <b>136</b> |
| 7.1        | 例题: 大整数加法 .....           | 136        |
| 7.2        | 例题: 大整数乘法 .....           | 138        |
| 7.3        | 例题: 大整数除法 .....           | 141        |
| 7.4        | 例题: 麦森数 .....             | 145        |
|            | 练习题 .....                 | 148        |
| <b>第8章</b> | <b>枚举 .....</b>           | <b>149</b> |
| 8.1        | 枚举的基本思想 .....             | 149        |
| 8.2        | 简单枚举的例子: 生理周期 .....       | 150        |
| 8.3        | 数学模型中包括多个变量的例子: 称硬币 ..... | 151        |
| 8.4        | 搜索空间中解不唯一的例子: 完美立方 .....  | 154        |
| 8.5        | 遍历搜索空间的例子: 熄灯问题 .....     | 156        |
| 8.6        | 优化判断条件的例子: 讨厌的青蛙 .....    | 160        |
|            | 练习题 .....                 | 165        |
| <b>第9章</b> | <b>递归 .....</b>           | <b>167</b> |
| 9.1        | 递归的基本思想 .....             | 167        |
| 9.2        | 例题: 全排列 .....             | 168        |
| 9.3        | 例题: 八皇后问题 .....           | 170        |
| 9.4        | 例题: 逆波兰表达式 .....          | 174        |
| 9.5        | 例题: 四则运算表达式求值 .....       | 175        |
| 9.6        | 例题: 放苹果 .....             | 179        |
| 9.7        | 例题: 简单的整数划分问题 .....       | 180        |
| 9.8        | 例题: 算 24 .....            | 181        |
| 9.9        | 例题: 红与黑 .....             | 184        |

|                                  |     |
|----------------------------------|-----|
| 9.10 例题: 二叉树 .....               | 186 |
| 9.11 例题: 拯救少林神棍 .....            | 187 |
| 练习题 .....                        | 193 |
| <b>第 10 章 动态规划</b> .....         | 196 |
| 10.1 什么是动态规划 .....               | 196 |
| 10.2 动态规划解题的一般思路 .....           | 199 |
| 10.3 例题: 最长上升子序列 .....           | 200 |
| 10.4 例题: 帮助 Jimmy .....          | 202 |
| 10.5 例题: 最长公共子序列 .....           | 206 |
| 10.6 例题: 神奇口袋 .....              | 208 |
| 10.7 例题: 灌溉草场 .....              | 210 |
| 10.8 例题: 方盒游戏 .....              | 215 |
| 10.9 例题: 美妙栅栏 .....              | 221 |
| 练习题 .....                        | 225 |
| <b>第 11 章 链表</b> .....           | 228 |
| 11.1 单向链表、链表结点的插入 .....          | 228 |
| 11.2 带表头的单向链表、链表的搜索 .....        | 232 |
| 11.3 双向链表、链表结点的排序 .....          | 235 |
| 11.4 循环链表、链表结点的删除 .....          | 238 |
| 11.5 链表的应用: 计算每个作业的运行时间 .....    | 241 |
| 练习题 .....                        | 247 |
| <b>第 12 章 二叉树</b> .....          | 249 |
| 12.1 二叉树的建立 .....                | 250 |
| 12.2 基于递归的二叉树遍历 .....            | 254 |
| 12.3 平衡二叉树 .....                 | 257 |
| 练习题 .....                        | 263 |
| <b>附录 A 北京大学程序在线评测系统介绍</b> ..... | 264 |
| A.1 POJ 的使用情况 .....              | 264 |
| A.2 POJ 的主要功能 .....              | 265 |
| A.3 使用本书结合 POJ 进行教学时的用法 .....    | 266 |
| <b>附录 B 本书题目在 POJ 上的编号</b> ..... | 267 |
| <b>致谢</b> .....                  | 271 |

# 第 1 章

## C/C++ 语言概述

本书将介绍 C/C++ 语言的基本语法,但由于篇幅所限,不涉及面向对象的内容。本书中的所有程序,都没有使用面向对象的编程方法,但都应保存为 .cpp 文件,按 C++ 的语法进行编译。实际上,如果不涉及面向对象的部分,那么 C++ 语言和 C 语言的语法 90% 以上是一样的,只不过略有扩充,用起来更为方便而已。因此,当提及的某项语法特性在 C 语言和 C++ 语言中都适用时,我们就会说:“在 C/C++ 语言中……”。

本书提到的 C/C++ 语言特性,以目前流行的 32 位计算机和操作系统上的情况为准。

本书的重点是通过一些编程实例介绍程序设计中常用的思想方法和实现手段,不侧重介绍某种高级程序设计语言的语法细节。本章对将要使用的 C/C++ 语言的相关内容作概要介绍,主要包括变量、常量、表达式、赋值语句、分支语句、循环语句、数组、指针和函数等内容。

每个程序都描述了一个计算过程,计算过程的输入数据、中间结果和最终结果都存储在程序的变量中。计算的每一步用一个表达式来描述,即用运算符对一些变量的值、常量进行处理。这种运算符可以是加、减、乘、除等算术运算;也可以是大于、小于、等于等关系运算;或者是与、或、非等逻辑运算符。表达式的结果可以存储在变量中。一个程序的基本组成单位是语句。连续的多个语句可以构成一个语句组。最基本的语句有变量定义语句和变量赋值语句。在程序执行过程中,语句按其出现的先后被顺序执行。分支语句可以根据不同的情况执行不同的语句组,而循环语句可以重复执行同一个语句组。当一个程序由很多语句组成时,可以将其中与某个功能相关的一组语句抽象出来定义成函数,并用函数名来代替原来的多个语句,这样可以隐蔽程序中的一些细节,使得程序逻辑更简单清晰。

### 1.1 程序的基本框架

下面以简单程序 Hello World 为例说明程序的基本框架。此程序在屏幕上输出一行“Hello World!”:

```
#include<stdio.h>
void main(){
    printf("Hello World!\n");
}
```

这段程序包括以下两个部分。



```
(1) #include<stdio.h>
```

#include 是 C 语言的保留字,表示要把另一个文件中的内容包含在本文件中。<stdio.h>是被包含的文件的文件名。C 语言中提供了一些可以被直接拿来使用、能够完成某些特定功能的库函数,分别声明于不同的头文件中。例如,stdio.h 中定义了一些与输入输出有关的函数。printf 就是一个能往屏幕上输出一串字符的库函数。

```
(2) void main(){
    printf("Hello World!\n");
}
```

程序的主函数。每个程序都必须包含这个 main() 函数。程序运行时,从 void main(){...} 的第一个语句开始执行。用户编写的程序的主要框架写在 main 函数里。

```
printf("Hello World!\n");
```

这条语句的作用是在屏幕上输出一串字符“Hello Word!”然后换行。其中的\n 的作用就是换行。换行后,如果以后再用 printf 语句来输出,那么输出的内容就会出现在屏幕的下一行。

## 1.2 变 量

变量是内存中的一块区域,在程序运行过程中可以修改这块区域中存放的数值。变量由两个要素构成:变量的名称和变量的类型。变量的名称是这个内存区域的唯一标识。变量的类型决定了这个内存区域的大小、对所存储数值的类型要求。在程序中,有三种与变量有关的语句:变量的定义、变量的赋值和变量的引用。

### 1.2.1 变量的定义

如下的语句定义了一个变量:

```
int number;
```

这里 number 是变量名,int 代表该变量是整数类型的变量,分号(;)表示定义语句结束。

在目前流行的机器配置下,整型变量一般占 4 个字节的内存空间。变量的名字是由编写程序的人确定的,它一般是一个单词或用下划线连接起来的一个词组,用于说明变量的用途。在 C/C++ 语言中,变量名是满足如下规定的一个符号序列:①由字母、数字或(和)下划线组成;②第一个符号为字母或下划线。需要指出的是,同一个字母的大写和小写是两个不同的符号。所以,team 和 TEAM 是两个不同的变量名。

定义变量时,也可以给它指定一个初始值。例如:

```
int numberOfStudents=80;
```

对于没有指定初始值的变量,它里面的内容可能是任意一个数值。

变量一定要先定义,然后才能使用。

### 1.2.2 变量的赋值

给变量指定一个新值的过程称为变量的赋值,通过赋值语句完成。例如:

```
number=36;
```

表示把 36 写入变量 number 中。

下面给出一些变量赋值语句的例子:

```
int temp;  
int count;  
temp=15;  
count=temp;  
count=count+1;  
temp=count;
```

### 1.2.3 变量的引用

变量里存储的数据可以参与表达式的运算,或者赋值给其他变量。这一过程称为变量的引用。例如:

```
int total=0;  
int p1=5000;  
int p2=300;  
int p3=1000;  
int p4=1000;  
total=p1+p2+p3+p4;
```

最后一个赋值语句表示把变量 p1、p2、p3 和 p4 的值取出来相加,得到的和赋给变量 total。最后一句执行后,total 的值变为 7300。

## 1.3 C/C++ 语言的数据类型

前面介绍了变量的定义语句:

```
int nNumber;
```

此处的 int 表示了变量 nNumber 的“数据类型”,它说明 nNumber 是一个“整型变量”,即 nNumber 中存放的是一个整数。“数据类型”能够说明一个变量表示什么样的数据(整数、浮点数或字符等)。不同数据类型的变量,占用的存储空间大小不同。除了 int 以外,C/C++ 中还有其他一些基本的数据类型,下面列举其中几个。

int: 整型。int 型变量表示一个整数,其范围是  $-2^{31} \sim 2^{31}-1$ ,占用 4 个字节。

long: 长整型。和 int 类型一样,也占用 4 个字节。

short: 短整型。short 型变量表示一个整数,但它占用 2 个字节,因而能表示的数的范围是  $-2^{15} \sim 2^{15}-1$ 。

unsigned int: 无符号整型。unsigned int 类型的变量表示一个非负整数, 占用 4 个字节, 能表示的数的范围是  $0 \sim 2^{32} - 1$ 。

unsigned long: 无符号长整型。其变量类型及占用存储空间大小和 unsigned int 一样。

unsigned short: 无符号短整型。unsigned short 类型的变量表示一个非负整数, 占用 2 个字节, 能表示的数的范围是  $0 \sim 2^{16} - 1$ 。

本书中将上面几种类型统称为“整数类型”。

char: 字符型。char 类型的变量表示一个字符, 如 'a' 和 '0' 等。占用 1 个字节。字符型变量存放的实际上是字符的 ASCII 码。例如, 'a' 的 ASCII 码是 97, 即十六进制的 0x61, 那么如果有

```
char c='a';
```

则实际上 c 中就存放着十六进制数 0x61, 或二进制数 01100001。

unsigned char: 无符号字符型。unsigned char 类型的变量表示一个字符, 占用 1 个字节。

float: 单精度浮点型。float 类型的变量表示一个浮点数(实数), 占用 4 个字节。

double: 双精度浮点型。double 类型的变量也表示一个浮点数, 但它占用 8 个字节, 因而精度比 float 类型高。

以上的 int、double、short、unsigned char 等标识符, 都是“类型名”。C++ 中的“类型名”可以由用户定义, 本书 1.15 节会进一步阐述。

在赋值语句中, 如果等号左边的变量类型为 T1, 等号右边的变量或常量类型为 T2, T1 和 T2 不相同, 那么编译器会将等号右边的变量或常量的值自动转换为一个 T1 类型的值, 再将此值赋给等号左边的变量, 这个过程称为“自动类型转换”。自动类型转换不会改变等号右边的变量。能进行自动类型转换的前提是: T1 和 T2 是两个兼容的类型。上面提到的所有类型正好都是两两互相兼容的, 但是后面会碰到一些类型, 比如指针类型、结构类型, 它们和上述所有的类型都不兼容。如果等号左边是一个整型变量, 等号右边是一个“结构类型”的变量, 这样的赋值语句在编译的时候就会报错。

下面以一个程序来说明上述数据类型之间的自动转换:

```
1.  #include<stdio.h>
2.  int main()
3.  {
4.      int n1=1378;
5.      short n2;
6.      char c='a';
7.      double d1=7.809;
8.      double d2;
9.      n2=c;           //n2 变为 97
10.     printf("c=%c,n2=%d\n", c, n2);
11.     c=n1;           //c 变为 'b'
12.     printf("c=%c,n1=%d\n", c, n1);
13.     n1=d1;          //n1 变为 7
14.     printf("n1=%d\n", n1);
```