



普通高等教育“十三五”规划教材

C语言程序设计

——面向工程应用实践（第2版）

© 牛连强 冯海文 侯春光 编著



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

普通高等教育“十三五”规划教材

C 语言程序设计

——面向工程应用实践

(第2版)

牛连强 冯海文 侯春光 编著

电子工业出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书全面介绍了 C 语言的语法知识和利用其进行程序设计的相关技术,包括 C 语言及程序设计的基础知识、数据组织、流程控制、函数封装、指针、字符串操作、构造数据类型及文件操作等。

在内容编排和结构组织上,本书力求精炼合理,难点概念合理分散,问题讲解简明生动,更重要的是将工程应用和强化实践能力的宗旨贯穿于每个部分。本书不同于一般的语法书籍,从基本习惯、说明案例至编程实战,都在不同程度上融入了工程元素,详细介绍了实际软件开发中的重点问题和注意事项,并说明哪些是工程应用中的合适技术,如何对代码进行不断的优化,使学习者能够进行“近距离”的工程应用知识学习。

本书可作为普通高等学校计算机科学与技术、软件工程、网络工程及其他相关专业 C 语言课程的教材,也适合 C 语言的初学者和相关培训机构使用。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

C 语言程序设计:面向工程应用实践 / 牛连强, 冯海文, 侯春光编著. — 2 版. — 北京: 电子工业出版社, 2017.2
ISBN 978-7-121-30674-7

I. ①C… II. ①牛… ②冯… ③侯… III. ①C 语言—程序设计—高等学校—教材 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2016)第 311325 号

策划编辑: 王羽佳

责任编辑: 周宏敏

印 刷: 三河市双峰印刷装订有限公司

装 订: 三河市双峰印刷装订有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787×1092 1/16 印张: 17.25 字数: 510 千字

版 次: 2013 年 7 月第 1 版

2017 年 2 月第 2 版

印 次: 2017 年 2 月第 1 次印刷

定 价: 43.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888, 88258888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式: (010) 88254535, wylj@phei.com.cn。

前言

C 语言以其独特的魅力吸引了众多的软件开发者,也被作为程序设计的一种基本语言来学习。它兼有高级语言和低级语言二者之长,代码简洁高效,功能强大。尽管以 C 语言为基础发展起来的 C++、Java 等语言已逐渐赶上甚至超过了 C 语言本身,但从实用性、易用性和学习的难度等多种角度看, C 语言仍是不可多得的语言,而“C 语言程序设计”也是大部分高校计算机及相关专业的必修课程。

十年前,本书第一作者应邀为辽宁省计算机基础教育学会编写了“C 语言程序设计”教材,并在多所高校使用,取得了良好的教学效果。该书的主要目的是实现普及性的程序设计知识教育,并兼顾国家的程序设计等级考试。随着高校向工程化实用型人才培养目标的转化,以及企业对高校毕业生动手实践能力的要求,促使我们对以前的教材进行整理和修改,从面向工程、面向应用的观点出发,结合新的 C 语言标准重新编撰了此书。

与现存的普及性教材和程序设计技巧和实例类的书籍不同,本书的读者对象是高校的在读学生,重点是在掌握语言及程序设计的一般知识基础上,突出面向工程应用的知识积累和解决实际问题的能力培养,强化学习的实践性和实用性。因此,本书紧密围绕着上述目标进行组织,并体现了如下主要特点:

■ **精心安排的体系结构。**尽管 C 语言本身是一种很小的语言,但设计一个完整程序所涉及到的知识点是很多的,且很多知识点存在着交叉、重叠。根据难点的合理拆分,知识点结合的紧密性以及完整项目与程序的组织原则等多种因素,确定了自己的体系结构,使前后内容自然衔接,循序渐进,也使读者能够更容易建立知识体系。

■ **采取面向工程 and 实际应用的原则。**虽然掌握程序设计的基本方法是计算机语言教学的目的之一,但作为一本大学生使用的教材,应该充分考虑解决工程中的实际应用所面临的问题,要使其了解哪些是工程应该中应该使用的技术,怎样的程序更符合工程应用的要求,哪些问题在实际工作中必须避免,等等。因此,除了对相关知识增加质量上的评价外,本书对重点问题、工程应用经验、应特别注意的事项等进行了详细总结和提炼,并以提示的方式贯穿于知识点的讲解过程。

■ **追求优秀的技术和高质量的程序。**为了使学习更靠近应用,首先,强化有应用背景的材料,并据此对实用技术做透彻的讲解。其次,利用后续逐步展开的新技术,不断修正前面已设计的程序,提高其质量,引导学生学会用更合适的技术来解决实际问题。

本书的内容共分 10 章。第 1 章介绍程序设计的基本概念、常识以及 C 语言的初步知识,可完成对 C 语言、程序设计及设计环境的基本了解。第 2 章和第 3 章介绍数据、运算、输入/输出、顺序与分支流程控制,可满足设计简单但完整的 C 语言程序的要求。第 4 章介绍循环结构以及数组的基本应用方法,可以解决稍微复杂的实际问题。第 5 章介绍函数,能够较全面地了解 C 语言程序的组织结构和模块化的程序设计技术。第 6 章介绍指针,这是 C 语言“高级”能力的体现。第 7 章介绍字符串,这是应用广泛且能够体现 C 语言程序设计技巧的内容。第 8 章介绍利用指针访问数组的方法、动态内存管理、指向函数的指针以及类型识别与描述方法,集中讨论了几种与指针相关的复杂概念和技术。第 9 章介绍自定义的数据类型,包括枚举、结构体和共用体,使程序能够顺利描述复杂的对象,建立复杂的数据结构。第 10 章介绍文件操作,完成数据在内存与磁盘文件之间的交互。

本书提供电子课件、习题参考答案和程序代码等,请登录华信教育资源网免费注册下载(<http://www.hxedu.com.cn>)。

在内容讲解上，本书尽量做到简明扼要，重点突出，采用更直观和直接的方法解释复杂的概念，以使读者容易理解并认清问题的本质。为了辅助学习和强化实践，每章除了正常的例题和习题之外，还特别安排了一定数量的编程实战题目。同时，本书还简要介绍了几种主要的 C (C++) 编程环境的使用方法，以便读者上机实习时选用。

我们希望本书能够高质量地满足工科学校计算机相关专业的教学需要，也特别希望读者能够不吝指出书中的缺点和错误，以便再版时能够加以改进。

作者的电子邮箱：niulq@sut.edu.cn。

编著者

2017 年 1 月

目 录

第1章 概述	1	2.5.1 运算符和表达式	34
1.1 程序设计基础	1	2.5.2 算术运算	35
1.1.1 问题的求解过程	1	2.5.3 赋值运算	35
1.1.2 算法及其描述	2	2.5.4 自加和自减运算	38
1.1.3 模块化与结构化	4	2.6 关系运算和逻辑运算	39
1.2 C语言及其特点	5	2.6.1 逻辑值	39
1.2.1 C语言的产生和发展	5	2.6.2 关系运算	40
1.2.2 C语言的主要特点	5	2.6.3 逻辑运算	41
1.3 C语言程序的基本结构	6	2.7 位运算	44
1.4 高级语言程序的处理过程	10	2.7.1 位运算符及表达式	44
1.5 利用 Visual C++ 6.0 环境编写和运行		2.7.2 位运算操作	44
C语言程序	11	2.8 sizeof 运算与逗号运算	48
1.5.1 VC6 环境的安装与运行	12	2.8.1 sizeof 运算符	48
1.5.2 编制一个(控制台)源程序	13	2.8.2 逗号运算符	49
1.5.3 编译、链接与运行程序	14	2.9 数据类型转换	50
1.5.4 程序调试技术	15	2.9.1 隐式类型转换	50
1.5.5 简单的程序调试与纠错	17	2.9.2 显式类型转换	51
1.6 习题	19	2.10 习题	52
1.7 编程实战	20	2.11 编程实战	54
第2章 数据与运算	21	第3章 简单程序设计	55
2.1 标识符与关键字	21	3.1 C语言语句概述	55
2.1.1 标识符	21	3.1.1 语句分类	55
2.1.2 关键字	22	3.1.2 语句的形式	56
2.2 数据类型	22	3.2 数据输出	58
2.3 常量	23	3.2.1 输出一个字符	58
2.3.1 直接常量与符号常量	23	3.2.2 按自定义格式输出数据	59
2.3.2 整型常量	24	3.3 数据输入	62
2.3.3 字符型常量与字符串常量	25	3.3.1 输入一个字符	62
2.3.4 浮点型常量	28	3.3.2 按自定义格式输入数据	63
2.4 变量	29	3.4 分支结构	66
2.4.1 变量定义与初始化	29	3.4.1 条件运算符与条件表达式	66
2.4.2 整型变量	31	3.4.2 if 语句	67
2.4.3 字符型变量	32	3.4.3 switch 语句与多分支处理	72
2.4.4 浮点型变量	33	3.5 习题	76
2.5 算术运算与赋值运算	33	3.6 编程实战	77

第4章 循环结构与数组	78	5.6.1 变量的生存期与作用域	120
4.1 while 语句与 do while 语句	78	5.6.2 局部变量	121
4.1.1 while 语句	78	5.6.3 外部变量	125
4.1.2 do while 语句	80	5.6.4 static 修饰、变量屏蔽和外部变量 的使用	127
4.2 for 语句	82	5.7 编译预处理指令	129
4.2.1 for 语句的语法	82	5.7.1 宏定义	129
4.2.2 for 语句的特殊形式	83	5.7.2 文件包含	131
4.3 流程转移语句	85	5.7.3 条件编译	132
4.3.1 break 语句	85	5.8 习题	132
4.3.2 continue 语句	86	5.9 编程实战	135
4.3.3 goto 语句	87	第6章 指针	138
4.4 循环结构的应用	88	6.1 指针与指针变量	138
4.5 一维数组的定义和引用	91	6.1.1 指针是经过包装的地址	138
4.5.1 一维数组的定义	91	6.1.2 指针变量	140
4.5.2 一维数组的引用	92	6.2 指针变量的赋值与指针的间接 引用	141
4.5.3 一维数组的定义初始化	93	6.2.1 指针变量的赋值	141
4.6 一维数组的应用	94	6.2.2 *运算符与间接访问	142
4.7 二维数组	97	6.2.3 指针变量的初始化	144
4.7.1 二维数组的定义与引用	97	6.3 指针的运算	145
4.7.2 二维数组的定义初始化	98	6.3.1 指针的加减算术运算	145
4.7.3 二维数组的应用	99	6.3.2 指针的自加和自减运算	147
4.8 习题	101	6.3.3 指针的比较	147
4.9 编程实战	103	6.4 指针作函数的参数	148
第5章 函数	104	6.4.1 修改实参变量的值	148
5.1 函数的定义与声明	104	6.4.2 从被调用函数取得信息	150
5.1.1 函数定义	104	6.5 利用指针访问一维数组	151
5.1.2 函数声明	107	6.5.1 利用指针实现快速数组元素 访问	151
5.2 函数调用及返回	108	6.5.2 一维数组名的指针含义	152
5.2.1 函数的调用过程	108	6.5.3 指针与数组的一致性	153
5.2.2 用 return 语句控制函数返回	109	6.6 向函数传递一维数组	154
5.3 形参与实参	111	6.7 习题	158
5.3.1 函数的形式参数	111	6.8 编程实战	161
5.3.2 实参与形参的匹配	112	第7章 字符串	162
5.3.3 函数调用表达式	112	7.1 用字符数组作字符串变量	162
5.4 参数的传值调用规则	114	7.1.1 对字符数组的特殊处理	162
5.5 递归调用	116	7.1.2 用作字符串变量的字符数组 初始化	164
5.5.1 递归调用过程	116		
5.5.2 典型的递归问题与函数示例	118		
5.6 变量的存储属性	120		

7.2 指向字符串常量的指针变量	165	第 9 章 自定义数据类型	212
7.2.1 界限模糊的字符串常量与变量	165	9.1 枚举	212
7.2.2 利用 const 限制指针的行为	166	9.2 结构体类型	214
7.3 字符串的输出与输入	167	9.2.1 结构体类型的定义	214
7.3.1 字符串输出	167	9.2.2 结构体类型的变量定义	216
7.3.2 字符串输入	168	9.3 结构体变量的引用	217
7.3.3 内存格式化	169	9.3.1 引用结构体成员	217
7.4 字符串操作	170	9.3.2 结构体变量的初始化	218
7.4.1 向函数传递字符串	170	9.3.3 整体引用结构体变量	218
7.4.2 返回指针的函数	171	9.3.4 结构体对象在函数间的传递	219
7.4.3 字符串运算函数	173	9.4 结构体数组和指针	220
7.4.4 字符串处理函数的设计	176	9.4.1 结构体数组	220
7.5 指针数组	178	9.4.2 结构体指针	221
7.5.1 指针数组的定义与引用	178	9.5 结构体指针的应用——链表	224
7.5.2 字符串数组	179	9.5.1 单向链表及其支撑结构	224
7.6 指向指针的指针	181	9.5.2 链表的创建与访问	225
7.6.1 指向指针的指针常量与变量	181	9.5.3 链表结点的查找、插入与删除	227
7.6.2 指针数组作函数参数	183	9.6 位段	231
7.7 习题	184	9.7 共用体	232
7.8 编程实战	187	9.8 习题	233
第 8 章 与指针相关的其他技术	189	9.9 编程实战	235
8.1 二维数组的指针访问	189	第 10 章 文件	236
8.1.1 二维数组的一维表示	189	10.1 文件概述	236
8.1.2 二维数组名的指针含义	190	10.1.1 文件的概念	236
8.1.3 指向一维数组的指针变量	192	10.1.2 文本流与二进制流	236
8.1.4 二维数组作函数参数	193	10.1.3 标准 I/O 和系统 I/O	237
8.1.5 利用二维数组实现的字符串 数组	195	10.2 文件的打开与关闭	238
8.2 动态内存管理	196	10.2.1 文件类型与文件指针	238
8.2.1 动态内存的申请与使用	197	10.2.2 文件的打开	238
8.2.2 calloc 函数与 realloc 函数	199	10.2.3 文件的关闭	241
8.3 指向函数的指针	201	10.3 文件的顺序读写	242
8.3.1 指向函数的指针常量与变量	202	10.3.1 字符读写函数 fgetc 和 fputc	242
8.3.2 函数指针的应用	203	10.3.2 文件尾检测	243
8.4 定义的识别与数据类型的显式 描述	205	10.3.3 getw 函数和 putw 函数	245
8.4.1 由运算识别复杂的定义	205	10.3.4 读写字符串函数 fgets 和 fputs	246
8.4.2 用 typedef 显式描述数据类型	207	10.3.5 格式化读写函数 fscanf 和 fprintf	247
8.5 习题	209	10.3.6 按块读写函数 fread 和 fwrite	248
8.6 编程实战	211	10.4 文件的随机读写	251
		10.4.1 以读/写方式打开文件	251

10.4.2	fseek 函数与读写位置的随机定位	252	附录 B	运算符的优先级与结合性	258
10.5	相关函数	253	附录 C	C++ Builder 编程环境的使用	259
10.6	习题	254	附录 D	DEV-C++编程环境简介	264
10.7	编程实战	255	参考文献		267
附录 A	常用字符与 ASCII 码对照表	257			

第1章 概述

计算机与信息技术的快速发展与普及应用,使得更多的问题可用计算机得到有效的解决。越来越多的人希望了解与程序设计相关的知识和技术,以便利用程序更高效地处理自己工作中遇到的问题。本章简要介绍与程序设计和C语言有关的一些基础知识,说明C语言程序的基本结构框架,并完成设计简单C语言程序的环境准备。

1.1 程序设计基础

1.1.1 问题的求解过程

简单地说,用计算机程序处理一个问题,就是把解题的技术和方法描述成计算机可以执行的一系列操作,并实施这些操作步骤,通常需要包含如下过程:

- ① 分析问题,建立数学模型;
- ② 设计解决问题的方法,即设计算法;
- ③ 确定程序的流程;
- ④ 编制和调试程序;
- ⑤ 运行程序,得到结果。

上述过程的表述并不十分严格,也略显抽象,以下将通过一个例子来说明。

我国古代数学家张丘建在《算经》一书中提出了一个“百鸡问题”,描述为:鸡翁一值钱五,鸡母一值钱三,鸡雏三值钱一。以百钱买百鸡,问鸡翁、母、雏各几何?以下是为了在计算机上求解此问题所要经历的主要过程:

(1) 建立数学模型

若用 x 、 y 和 z 分别表示鸡翁、母、雏的数量,可得到如下的数学关系:

$$\begin{cases} x+y+z=100 \\ 5x+3y+z/3=100 \end{cases}$$

这是一个不定方程。因为 x 、 y 和 z 都是整数,并有一定的取值范围,这就决定了用计算机来描述这些数据时所采用的数据类型,也确定了存储这些值所占用的存储空间大小。这两个等式建立了数据之间的关系,而这些相互依存的数据及其关系构成了数据结构。

(2) 设计求解算法

此问题较简单,可以采用“穷举法”来解决。由于 x 、 y 和 z 都是0~100之间的整数,可以将(0,0,0)至(100,100,100)之间的任意一组整数值代入这两个等式进行测试。如果某一组值使二者都成立,就得到了原问题的一个解。很明显,这样的算法不适合手工计算,它说明了计算机求解算法与手算解法之间是有一定差异的,尽管存在简单的方法会更好。

(3) 描述算法的流程

长期以来,人们使用了各种技术和工具来描述一个算法(程序)的轮廓,主要是说明实现该算法所要经历的操作步骤,包括伪语言、程序流程图、PAD图,以及建模工具UML(Unified Modeling Language,统一建模语言)图等。为了简单起见,本书中使用程序流程图作为描述工具。

（4）编制程序

这是学习计算机语言的主要目的。所谓计算机语言就是为了描述计算机操作步骤而制定的一整套标记符号、表达格式和语法规则。“程序”（Program）就是指为实现特定目标或解决特定问题而用计算机语言编写的命令序列集合，或者说是为实现预期目的而执行的一系列语句和指令，其核心是对数据的描述和处理。

计算机语言分为低级语言和高级语言两类。最低级的语言是机器语言，它能被计算机直接识别，而其他语言则不能。因此，几乎每一种语言都配有一个相应的“翻译”程序，称之为“编译程序”，用于将自己的代码翻译为机器指令。

目前仍在广泛使用的一种低级语言是汇编语言，它是通过直接将机器指令符号化后形成的语言，功能强，且代码效率高，但更接近于计算机硬件，编程困难，软件开发效率低，不易被非专业人员所接受。相比之下，高级语言是为了更好、更直观地描述算法，方便程序设计而发展起来的通用型语言，如 BASIC、FORTRAN、COBOL、PASCAL、C、C++、JAVA、C#和 PYTHON 等。高级语言的语法比较接近数学表达式描述，容易理解和编程。大部分高级语言支持结构化、模块化的程序设计方法，新兴高级语言还提供了对面向对象设计技术的支持，被广泛地用于科学计算和事务处理等诸多方面。

计算机程序可以大体上归结为系统程序和应用程序两大类。其中，“系统程序”是指为维持计算机和网络系统正常运行而设计的程序，典型的系统程序是操作系统中所包含的程序。相对地，“应用程序”是指为某些特殊目的和解决某些特定问题而编制的程序，如数据库管理、绘图、文字处理以及图像处理等。各种程序及其相关的技术文档的总称即为“软件”。

在解决一个问题时，原则上可以使用任何一种语言来编制程序，但每种语言都有自己的特性和适合的领域，所编制程序的效率也有较大差异。长期以来，人们期望能有一种既具有高级语言特点，又兼有低级语言的高效和强硬件操作能力的语言，这导致了 C 语言的诞生。

（5）运行程序

程序编制后，需要进行多方测试以修改其中存在的错误。测试和证实程序的正确性也是一个专门的技术领域。通常，需要使用多组数据集来运行程序进行观察，直到没有错误为止。确信程序无误后，再将其真正投入运行，得到所需结果。

1.1.2 算法及其描述

程序设计主要包括数据结构和算法设计。“数据结构”用于指定数据及其相互关系，“算法”则是对特定问题求解步骤的一种逻辑描述，二者密不可分。合适的数据结构可以高效地支持插入、查找和排序等基本操作，有利于简化算法的设计，而每种事务的顺利处理则要依赖有效的算法。

著名计算机科学家沃思（Niklaus Wirth，图灵奖获得者）曾经提出了一个公式，即“数据结构+算法=程序”。实际上，一个程序除了以上两个关键要素外，还应当采用适当的程序设计方法进行设计，并且用一种计算机语言来表示。因此，算法、数据结构、程序设计方法和语言工具是一个程序员必须具备的四个方面知识。

一些常用的基本数据结构和算法可以在“数据结构”、“算法设计与分析”等知识领域中找到，只有了解这些基础知识，才能借助它们构建更复杂有效的解题方法。在实际设计一个算法时需要注意到如下问题：

（1）有穷性

这是指一个算法必须在执行有限个步骤之后结束。

（2）确定性

任何算法的每一步必须被确切定义，不能出现模糊性、多义性或不确定性，对于相同的输入必须得到相同的结果。

(3) 可行性

算法的每一步都是能够实现的，或者说是可操作的。

(4) 有输出

在算法开始时，可以没有输入量或者有多个输入量，但算法执行完毕后，必须有一个或多个输出量。

设计一个好的算法通常要考虑到正确性、易读性、健壮性、高效性和低存储量等诸多方面的因素。由于本书以介绍语言的语法和程序构建方法为主要目的，较少涉及复杂的算法，一般也只是给出算法的简单流程或语言描述。

采用流程图描述算法时要使用一些标准化的符号。图 1.1 列出了国标 GB1526-89 所推荐的一套流程图标准化符号，这些符号与国际标准化组织 ISO 提出的流程图符号一致。

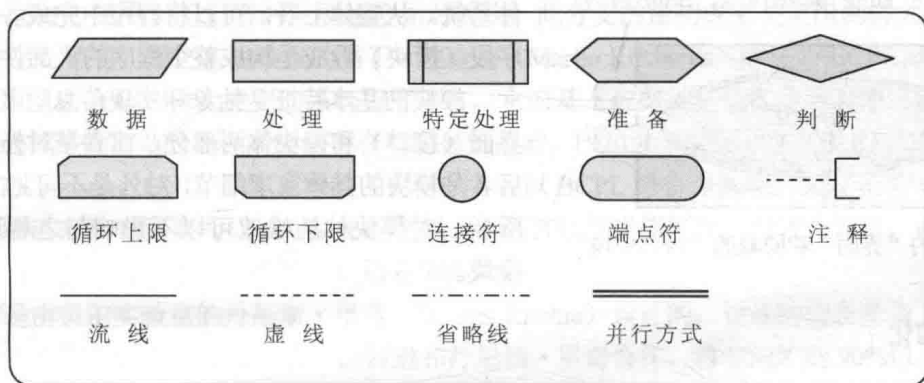


图 1.1 流程图符号

图 1.2 利用流程图描述了 1.1.1 节中的“百鸡问题”的求解算法。注意到流线的箭头画法为：若方向向右或向下，可以标出箭头也可以不标，否则必须标出箭头。开始和结束符号不是圆角矩形。此外，还可以采用循环符号来描述循环结构。

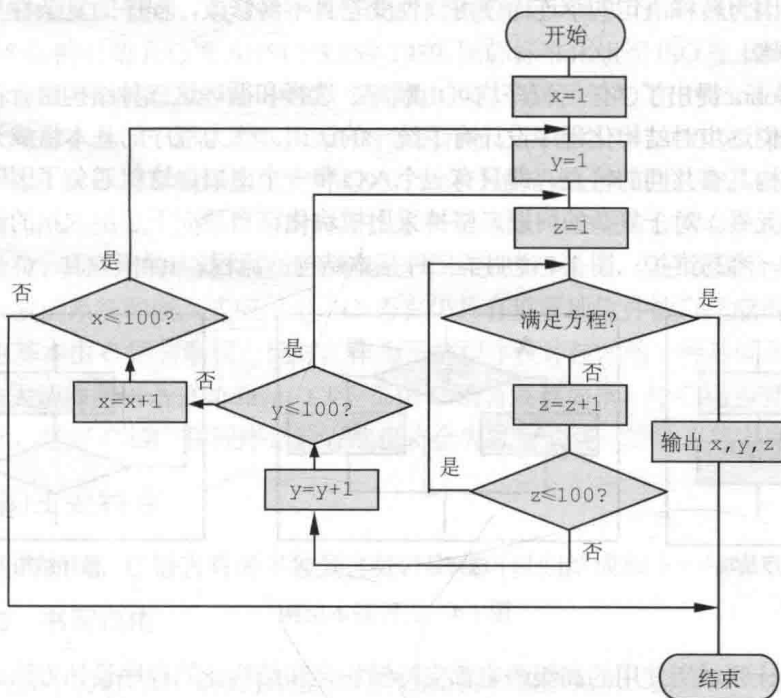


图 1.2 百鸡问题的流程图

1.1.3 模块化与结构化

在软件设计的发展过程中,模块化和结构化是已被证明的有效设计技术和方法。

1. 模块化

模块化的概念在计算机软件设计领域中已经沿用了 40 多年,是处理复杂问题所应采取的关键技术,也是使得软件能够被有效管理和维护所应具备的关键特性。

在一个复杂的软件中,任务被划分成若干个可单独命名和编址的部分,这些部分被称为“模块”,

界面: `double func(double x, double y)`



图 1.3 公开的“界面”和隐藏的“内部实现”

它们之间通过有目的的相互连接组成满足应用需要的软件系统。从整体上看,可以将程序中完成各种处理的程序段(模块)看成是构成整个程序的“部件”。

模块的基本特征是抽象和实现信息隐藏,分为模块界面(接口)和模块体两部分,前者是对外的“窗口”,后者是模块的具体实现细节,对外是不可见的,如图 1.3 所示。对模块体的修改可以不影响与之相连接的其他模块。

2. 结构化

在一个模块的内部要实现其应该完成任务的细节,即规定机器在实施时应遵循的处理流程。在软件设计发展的初期,程序中可以随时根据处理的需要确定流程的走向,从一个局部结构跳转到另一个局部结构,这种跳转称为“无条件转向”或“转移”。早期的高级语言如 BASIC 等就依赖这样的指令(语句)。

1963 年,一些计算机专家提出,程序中大量、无限制地使用无条件转移语句会使程序结构杂乱无章,各部分之间会因为转移语句的牵连而使可读性变差且不易修改,程序的复杂性与所用的无条件转向语句的数量成正比。

1966 年, C. Bohra 提出了“任何程序均可由顺序、选择和循环这三种结构组合而成”的观点并得到证明。由此,人们逐步对结构化程序设计有了统一的认识。作为程序的基本组成元素,顺序结构、选择结构和循环结构具有共同的特点,即只有一个入口和一个出口,这就避免了因随意转移流程的走向而破坏这些基本元素。对于复杂的问题,坚持采用模块化、自顶向下逐步求精的设计原则,可以使程序结构清楚易读,容易维护。图 1.4 说明了三种基本结构的流程。

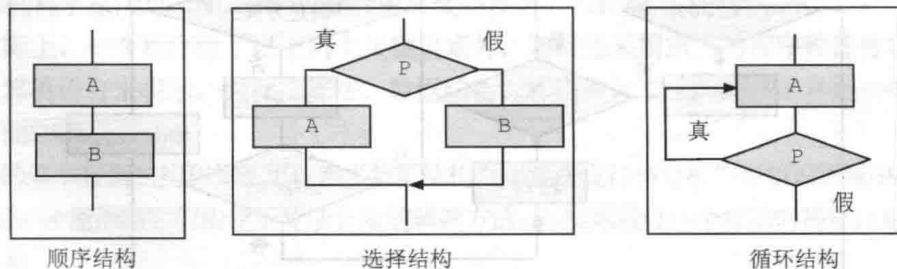


图 1.4 三种基本结构

目前,软件设计领域所使用的高级语言都支持模块化和结构化的程序设计方法。

1.2 C 语言及其特点

C 语言是国际上极为流行且被广泛应用的一种高级语言,既可用于编制系统软件,也可用来编写应用软件。

1.2.1 C 语言的产生和发展

C 语言起源于 ALGOL 60 语言。ALGOL 60 语言曾在 20 世纪 60 年代流行一时,但因距离系统硬件较远,难以编写系统软件而逐渐淡出程序设计领域。后来,该语言逐渐发展成二个分支,分别是 PASCAL 语言和 C 语言,且后者的发展速度相对较快。此分支的最初成果是英国剑桥大学在 1963 年研制出来的 CPL 语言,随之,由该校的马丁·理查德(Martin Ritchards)改进为 BCPL 语言并于 1967 年发表。BCPL 语言是一种无类型的系统程序语言,它的基本数据类型是机器字,使用较多的指针和地址运算,这与其他常见的各种高级语言有着明显的差异。1970 年,美国贝尔(Bell)实验室的肯·汤姆逊(Ken Thompson, UNIX 操作系统的主要研制者)以 BCPL 语言为基础,又发表了一种新的 B 语言,并用汇编语言和 B 语言写成了 UNIX 的初版。B 语言接近机器硬件,但较为简单,而且无常用的数据类型。为了克服 B 语言的局限性, C 语言应运而生。

C 语言是由贝尔实验室的丹尼斯·里奇(Dennis Ritchie)设计的,设计的初衷是为了更好地描述和实现 UNIX 操作系统。1973 年,肯·汤姆逊和丹尼斯·里奇合作,将 UNIX 的 90%以上的代码用 C 语言改写,形成了 UNIX 第 5 版,但此时的 C 语言并没有引起人们太多的注意。随着 C 语言的发展,1977 年出现了不依赖于具体机器的 C 语言编译文本,而 C 语言的真正定义是在 1978 年,在由布莱恩·科尼汉(Brian Kernighan)和丹尼斯·里奇(Dennis Ritchie)所著的 *The C Programming Language* 中被阐述清楚。这是一本影响深远的名著,实际上成了后来的 C 语言标准。

1983 年,美国国家标准化协会(ANSI)整理和扩充了当时的各种 C 语言版本,并制定了一个 C 语言标准,即 83 ANSI C。随后,在 1987 年对其做了重新修订,制定了 87 ANSI 标准 C。1989 年,ANSI 又公布了一个新的 C 语言标准 ANSI X3.159-1989,国际标准化组织 ISO 在 1990 年接受了该标准,使其正式成为国际标准 ISO/IEC 9899: 1990,简称为 C90。随后,ISO 在 1995 年和 1999 年分别又对该标准进行过两次修订,为 C 语言增加了面向对象的特征,并命名为 ISO/IEC 9899: 1999,简称为 C99。不过,由于通常的 C 语言编译器仅支持 C90,故本书的语法介绍也主要以 C90 标准为基础。

近年来的 C 语言发展十分迅速,以面向对象技术和 C 语言语法相结合的 C++语言在软件设计领域占有举足轻重的地位,其应用的广泛性也已逐渐超过了传统的 C 语言,但它更适合开发大型应用系统。在一些涉及硬件开发的系统和嵌入式应用中, C 语言仍具有重要地位并被广泛应用,传统的以汇编为开发语言的应用也基本由 C 语言取代。同时,作为了解程序设计技术的一种基础语言, C 语言也具有独特的优势,因为大量的新兴语言如 Java、C#等都由 C 语言发展而来,与 C 语言有着大量一致的语法现象和规则。因此,学好 C 语言和程序设计方法也将会为以后的进一步学习奠定良好的基础。

1.2.2 C 语言的主要特点

与其他高级语言相比, C 语言有诸多独到之处,主要可以归纳成如下一些特点。

1. 语言简洁、书写自由

C 语言是一种很小的语言,具有精心选择的控制结构和数据类型,摒弃了一切不必要的成分,从而使其规模缩减到最小,代码简洁而高效。C 程序的书写也几乎不受什么限制,可以在一行上书写多

个语句，也可以把一个语句写在多个行上。当然，这种灵活性可能使程序缺少一种可以遵循的标准，因此，学习时应尽量注意程序书写的“规范性”，这一点可以根据本书所提供的示例程序去体会。

2. 运算符丰富

运算符的多少体现了语言对数据的加工能力强弱。C 语言提供了极为丰富的运算符，共 34 种（参见附录 B），这使 C 语言可以直接实现其他语言中很多难以实现的运算，同时也提高了语句的能力。

3. 数据类型丰富

C 语言提供的多种数据类型使程序员可以灵活、方便地构造各种复杂的数据结构和设计适应各种问题的算法。表面上，C 语言支持的数据类型与 PASCAL 语言接近，但 C 语言中的指针类型比 PASCAL 更灵活，也具有更强的操作能力。

4. 结构化良好

C 语言提供了典型的结构化控制语句，是理想的结构化语言，而且 C 语言用函数来组装程序，进而支持程序的模块化。

5. 语法限制不严格

这是 C 语言存在争议的一个特点，这是因为，语法限制不严给程序设计带了灵活性，但也对程序的安全性有“不良”影响。例如，最为典型的数组超界问题就是语法限制不严的产物，不论程序使用了数组的多少元素，是否与定义吻合，C 语言的编译器都不会给出错误通知，因为 C 编译器不做数组边界的检查。如果在设计时不能确保引用的正确性，超界的数组元素有可能导致灾难性的后果，使程序彻底崩溃。

6. 硬件操作能力强

C 语言具有低级语言所具有的特殊能力，体现在对内存地址的直接操作和位运算。通常，这是低级语言所具有的两种能力，以便更直接地操作机器硬件。作为高级语言的 C 也具有这两种能力，或者说具有低级语言的功能特征，因此有人称其为“中级语言”。

此外，从生成代码的效率来看，C 程序明显优于其他高级语言。借助 C 语言的低级语言能力，一些程序员正在用它代替汇编语言来书写系统程序，这不仅使程序的研制周期得以缩短，也使程序具有较高的可移植性。

当然，上述特点只是使用者对 C 语言的一般性评价，更深刻的体会需要在进一步学习和比较之后才能得到。

1.3 C 语言程序的基本结构

这里给出几个简单的 C 语言程序，目的是建立对 C 语言程序的感性认识，并能理解书中使用的程序结构。阅读时应注重代码后的正文解释，不必仔细研究程序代码中有关输入、输出函数的格式等细节，后续章节中会对这些内容做进一步的讨论。

例 1.1 在屏幕上输出“Hello world!”的 C 语言程序。

最简单的 C 程序仅由下面的代码实现：

```
void main( )  
{  
      
}
```

这是一个完整的 C 程序，仅由一个模块组成，但它不做任何工作。将其编译成可执行文件并运行时，只相当于做了一次空操作。

这个基本的程序框架也可以写成如下形式：

```
int main( )
{
    return 0;
}
```

尽管此程序还不能输出题目要求的文字，但可以说明 C 语言程序的一些基本特征。

1. 由函数组成程序与 main 函数

C 程序是由函数组装而成的，一个完整的 C 程序中必须且只能有一个名字为 main 的函数，称作“主函数”。若程序简单，可以只由这一个函数构成完整的程序，此例即如此。不管组成一个 C 程序的函数有多少，总是从 main 函数的第一个语句开始执行。

2. 函数的基本结构

通常，每个函数名之前有一个数据类型，用来表示函数的值是什么类型。main 函数的数据类型可以是 void 或 int。一般应用中的 main 函数仅执行一些处理指令，不带回任何计算结果，可将其类型指定为 void。如果采用 int 作为函数的数据类型，可以将“return 0;”置于最后一行，其作用是向操作系统表明程序正常执行结束。为了避免引入过多的成分，本书主要使用第一种形式，即 void 类型的 main 函数。

代码中跟在数据类型之后的是函数名，除了 main 函数的名字为固定写法以外，其他函数都由设计者自己命名。每个函数名之后都要有一对圆括号，这是函数的标志。函数名和圆括号之后的部分称为“函数体”，置于括号{ }内，这是书写函数代码的地方，用于说明函数所要做的全部工作。

与一些简单的语言如 BASIC 等不同，无论构成程序的代码有多少，不能仅由语句构成 C 程序，必须将代码置于函数框架内。或者说，C 语言程序的基本组成单位是函数而非语句。

由于 C 语言对书写的要求并不严格，因此上述程序中的各部分也可以写在一行内，但这并不是一种可取的书写方式，如：

```
void main( ) { } /* 不可取的书写方式 */
```

下述程序在基本框架中增加了一个语句，使其能够按要求输出文字“Hello world!”。

```
#include <stdio.h>
void main( )
{
    printf("Hello world!"); /* display a string. */
}
```

以下是程序中所添加内容的含义。

1. 输出语句

```
printf("Hello world!");
```

这是一次“库函数”调用。C 语言提供了大量预先设计的函数，以完成一些常见的事务性处理工作，如输入和输出等。只要了解它们的功能和使用方式（格式），就可以直接使用而不用自己去编写，这些函数称为“库函数”。此处的 printf 就是一个用于输出数据的库函数，这样的语句也称为“输出语句”。

2. 文件包含指令

```
#include <stdio.h>
```

添加此行代码是因为使用了库函数 `printf`。通常，C 语言要求对使用者声明函数的格式。这里的格式就是接口或界面。对于自己定义的函数，可以在第 5 章中了解说明方法，而对于每一个库函数，其格式都记录在系统提供的一个文件里。例如，函数 `printf` 的格式记录在文件 `stdio.h` 中，或者说 `printf` 函数定义于文件 `stdio.h`。C 语言允许使用命令 `#include <stdio.h>` 来对库函数 `printf` 进行声明，以便让编译器从文件 `stdio.h` 中查找并了解函数 `printf` 的格式。因此，了解每个库函数所属的文件（称为“头文件”）是必要的任务，并注意在程序开头加上如下代码，称为“头文件包含”：

```
#include <头文件名>
```

如果几个库函数定义于同一个头文件，只要包含该文件一次就可以了。

★提示

`stdio.h` 中的 `std`、`i` 和 `o` 分别来自单词 **standard**、**input** 和 **output**，即标准输入输出；字符 `h` 来自单词 **head**，故称为头文件或头部文件。

3. 注释

```
/*...*/
```

在这一对符号中间的内容称为“注释”。注释是对程序中的代码、变量等所做的说明，在程序编译时，编译器会舍弃注释而不做任何处理，执行时自然也不会引起任何机器操作。对程序的注释有助于提高程序的可读性，是帮助阅读者理解源程序代码的非常必要的部分。

C 语言程序的注释非常灵活，可以在任何允许插入空格的地方插入注释，但正确的做法通常是用单独的行添加必要的注释，不严格要求时也可以加在行末。限于篇幅，本书的注释以中文形式在行末给出，用于解释所在行代码的功能、含义和应注意的事项。注意，应尽量以某种方式对齐注释，以便于阅读。

注释不可嵌套。

★工程

实际项目中通常这样对程序进行注释：

- (1) 在程序（文件）头，添加有关程序名、功能、作者、目的、用途、修改历史等的注释；
- (2) 在函数前，添加有关函数的功能、参数、返回值和副作用等的注释；
- (3) 在变量定义前（或后），说明变量的含义；
- (4) 在函数体开头，解释算法思路。

工程项目中一般对注释的方法有严格的要求，甚至对注释的数量也非常重视，因为一旦缺少注释，代码将很难理解和维护。另外，应该在编写程序的同时为代码添加适当的注释，而不是在项目开发结束后一次性地添加注释。总之，注释是软件的重要组成部分，而正确添加注释也是软件项目开发的重要工作之一。

通常，一个完整的程序总会包含数据输入、处理以及数据输出等几个部分。

例 1.2 从键盘输入两个整数，输出它们的和。

```
#include <stdio.h>
void main( )
```