



21世纪高等学校计算机科学与技术规划教材

© 主编 占跃华 主审 王明文

C YUYAN CHENGXU SHEJI C语言程序设计

第2版



北京邮电大学出版社
www.buptpress.com



21 世纪高等学校计算机科学与技术规划教材

C 语言程序设计

(第 2 版)

主 编 占跃华

副主编 虞 芬 陈忠文

主 审 王明文

北京邮电大学出版社

• 北京 •

内 容 简 介

C语言是目前较好的学习程序设计的语言,“C语言程序设计”课程是程序设计的重要基础课,是培养学生程序设计能力的重要课程之一。因此,学好“C语言程序设计”课程对掌握基本编程方法、培养基本编程素质具有重要意义。

本书是作者多年来在讲授“C语言程序设计”课程的基础上,总结多年的教学经验,对授课内容做了深入细致的研究后整理而成的。本书采用“以用促学”的编写原则,即通过编写应用程序来学习C语言抽象的标准和规则。本书不仅在内容上强调逻辑性,更注重介绍学习方法,使学生能根据例题举一反三。本书结构新颖、实例丰富,强调语言的规范和程序设计的方法与技巧,注重培养学生程序设计的思维方式和提高学生程序开发的能力。本书共12章:第1章C语言概述,第2章C语言的基本知识,第3章运算符和表达式,第4章顺序和选择结构程序设计,第5章循环结构程序设计,第6章数组,第7章函数,第8章指针,第9章结构体与其他数据类型,第10章文件,第11章预处理命令,第12章位运算。全书重点为第4章、第5章和第6章,难点为第7章、第8章和第9章。

本书适合作为高等院校各专业“C语言程序设计”课程的教材,也可用作社会教育培训的教学用书,授课内容、例题和习题可根据实际情况进行选用。

图书在版编目(CIP)数据

C语言程序设计/占跃华主编. —2版. —北京:北京邮电大学出版社,2011.1

ISBN 978-7-5635-2532-4

I. ①C… II. ①占… III. ①C语言—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字(2010)第255177号

书 名	C语言程序设计
主 编	占跃华
责任编辑	沈 辉
出版发行	北京邮电大学出版社
社 址	北京市海淀区西土城路10号(100876)
电话传真	010-82333010 62282185(发行部) 010-82333009 62283578(传真)
电子信箱	ctrd@buptpress.com
经 销	各地新华书店
印 刷	北京市梦宇印务有限公司
开 本	787 mm×1 092 mm 1/16
印 张	17.5
字 数	400千字
版 次	2010年12月第2版 2010年12月第1次印刷

ISBN 978-7-5635-2532-4

定价: 30.00 元

如有质量问题请与发行部联系
版权所有 侵权必究

前 言

C 语言是一种应用广泛的程序设计语言,它的功能强大、使用灵活、可移植性好,既具有高级语言的优点,又具有低级语言的特点,可用于编写系统软件,也可用于编写应用软件。C 语言的语法规则清晰,便于掌握和记忆,是大多数人学习计算机程序设计的入门语言。因此,所有计算机专业和许多理工科专业都开设“C 语言程序设计”课程。

本书在内容组织上根据作者在 C 语言教学过程中遇到的问题和教学经验进行了有效的改进,突出了 C 语言的学习目的,使学习和掌握 C 语言更加容易和快捷。本教材具有以下特点。

1. 改变了以往部分教材过于注重语法的不足。通过对 C 语言的学习,掌握 C 语言的语法规则固然重要,但语法规则毕竟比较枯燥,过多强调语法容易使读者丧失学习兴趣,不能使读者真正有效掌握。本书在讲解语法规则的过程中,始终通过例题的方式,将语法规则融入例题中加以讲解。一方面加强语法规则的理解;另一方面在学习语法的同时掌握了 C 语言的应用,充分调动了学习兴趣。

2. 本书结构严谨,由浅入深,循序渐进。本书系统全面地讲述了 C 语言的相关知识,在内容安排上由浅入深,循序渐进,使读者不断有收获,不断有提高。在保持学习兴趣的前提下完整掌握 C 语言的相关知识。

3. 突出 C 语言的学习目的。通过 C 语言的学习,重要的是掌握结构化程序设计的基本方法。本书通过大量例题,根据教师的教学经验,将 C 语言的典型应用加以讲述,使读者能真正掌握利用 C 语言编写结构化程序的基本方法,同时提高阅读 C 语言程序的能力。

4. 概念准确,讲解细致,文字通俗易懂,本书概念严谨准确,对 C 语言的内容进行了详细的讲解,文字通俗易懂,对难点问题配有图解,帮助读者加深理解。本书便于教师讲授,也便于学生自学。

5. 与《C 语言程序设计实训教程》(第 2 版)配合使用,能有效提高参加计算机各类考试的读者的应试能力和实际编程能力。程序设计是一门实践性很强的课程,读者应当十分重视自己动手编写程序和上机运行程序。

本书由占跃华、虞芬、陈忠文、裴南平、毕传林、徐敬东、艾迪、代飞等老师共同编写,由占跃华任主编并统稿,由王明文任主审。在此还要特别感谢李圣良主任,他对本书的编写提出了很多建设性意见。

由于编者水平有限,书中难免存在不足,恳请读者批评指正。

编 者

目 录

第 1 章 C 语言概述	(1)
1.1 程序与程序设计语言	(1)
1.1.1 程序	(1)
1.1.2 程序设计语言	(2)
1.2 C 语言发展概述和主要特点	(3)
1.2.1 C 语言的发展历史	(3)
1.2.2 C 语言的主要特点	(3)
1.3 C 程序的基本结构	(4)
1.4 程序的调试	(7)
1.4.1 调试步骤	(7)
1.4.2 Visual C++ 6.0 集成开发环境	(8)
本章小结	(10)
习题 1	(11)
第 2 章 C 语言的基本知识	(12)
2.1 C 语言的数据类型	(12)
2.2 变量与常量	(13)
2.2.1 变量	(13)
2.2.2 常量	(15)
2.3 整型数据	(16)
2.3.1 整型变量	(16)
2.3.2 整型常量	(19)
2.3.3 用 printf() 输出整型数据	(20)
2.3.4 用 scanf() 输入整型数据	(24)
2.4 浮点数	(27)
2.4.1 浮点变量	(27)
2.4.2 浮点常量	(29)
2.4.3 用 printf() 输出浮点数据	(29)
2.4.4 用 scanf() 输入浮点数据	(31)
2.5 字符型数据	(32)
2.5.1 字符变量	(32)
2.5.2 字符常量	(33)
2.5.3 用 printf() 输出字符	(34)
2.5.4 用 scanf() 输入字符	(35)

2.5.5 用 getchar()输入字符和用 putchar()输出字符	(36)
2.5.6 字符串常量	(37)
本章小结	(38)
习题 2	(39)
第 3 章 运算符和表达式	(41)
3.1 表达式	(41)
3.2 算术运算符	(43)
3.3 赋值运算符	(44)
3.3.1 普通赋值运算符	(44)
3.3.2 复合赋值运算符	(45)
3.4 不同数据类型的数据间的混合运算	(46)
3.4.1 自动类型转换	(46)
3.4.2 强制类型转换	(48)
3.4.3 赋值表达式的类型转换	(48)
3.5 自增、自减运算符	(50)
3.6 逗号运算符	(52)
3.7 其他运算符	(53)
本章小结	(54)
习题 3	(55)
第 4 章 顺序和选择结构程序设计	(57)
4.1 C 语句概述	(57)
4.2 算法基本知识	(59)
4.2.1 算法的特性和要素	(59)
4.2.2 算法的描述	(59)
4.3 顺序结构程序设计	(62)
4.4 关系运算与逻辑运算	(63)
4.4.1 关系运算	(63)
4.4.2 逻辑运算	(64)
4.4.3 程序中对条件的描述	(66)
4.5 if 语句	(67)
4.5.1 if 语句的基本形式	(67)
4.5.2 缺省 else 结构的 if 语句	(69)
4.6 if 语句的嵌套	(71)
4.7 条件运算符	(75)
4.8 switch 语句	(76)
4.9 选择结构程序设计举例	(80)
本章小结	(83)
习题 4	(84)
第 5 章 循环结构程序设计	(85)
5.1 while 语句	(85)

5.2 do...while 语句	(89)
5.3 for 语句	(91)
5.4 break 语句与 continue 语句	(94)
5.4.1 break 语句	(94)
5.4.2 continue 语句	(96)
5.5 循环的嵌套	(97)
5.6 用 if 和 goto 语句构成的循环结构	(101)
5.7 程序举例	(102)
本章小结	(105)
习题 5	(106)
第 6 章 数组	(107)
6.1 一维数组	(107)
6.1.1 一维数组的说明	(107)
6.1.2 一维数组的引用	(108)
6.1.3 一维数组的初始化	(110)
6.1.4 一维数组的程序举例	(111)
6.2 二维数组	(116)
6.2.1 二维数组的说明	(116)
6.2.2 二维数组的引用	(117)
6.2.3 二维数组的初始化	(118)
6.2.4 二维数组的程序举例	(119)
6.3 字符数组与字符串	(122)
6.3.1 字符数组	(122)
6.3.2 用字符数组存储字符串	(124)
6.3.3 字符串输入输出	(125)
6.3.4 字符串处理函数	(127)
6.3.5 字符串应用举例	(130)
本章小结	(133)
习题 6	(134)
第 7 章 函数	(136)
7.1 函数概述	(136)
7.2 函数的定义和调用	(137)
7.2.1 函数定义	(137)
7.2.2 函数调用	(139)
7.2.3 形参和实参	(140)
7.2.4 函数的值	(143)
7.3 函数的嵌套调用	(144)
7.4 函数的递归调用	(147)
7.5 局部变量和全局变量	(148)
7.5.1 局部变量	(149)

7.5.2 全局变量	(149)
7.6 数据的存储类别	(151)
7.6.1 动态存储与静态存储	(151)
7.6.2 auto 变量	(152)
7.6.3 register 变量	(152)
7.6.4 static 变量	(152)
7.6.5 extern 变量	(153)
7.7 内部函数与外部函数	(155)
本章小结	(156)
习题 7	(157)
第 8 章 指针	(159)
8.1 地址与指针	(159)
8.2 指针变量	(161)
8.2.1 指针的定义	(161)
8.2.2 指针变量的引用	(161)
8.2.3 指针变量的使用	(162)
8.3 指针与数组	(165)
8.3.1 通过指针访问一维数组	(165)
8.3.2 通过指针访问二维数组	(167)
8.3.3 指针的基本运算	(170)
8.4 指针与字符串	(172)
8.4.1 字符数组与字符指针	(172)
8.4.2 字符指针举例	(173)
8.5 指针与函数	(175)
8.5.1 指针作函数的参数	(175)
8.5.2 数组名作函数的参数	(178)
8.5.3 函数返回值是指针	(183)
8.5.4 指向函数的指针	(184)
8.6 指针数组	(185)
8.6.1 指针数组	(185)
8.6.2 main()函数的参数	(186)
8.7 多级指针	(187)
本章小结	(189)
习题 8	(189)
第 9 章 结构体与其他数据类型	(192)
9.1 结构体类型	(192)
9.1.1 声明结构体类型	(193)
9.1.2 结构体变量的定义	(194)
9.1.3 结构体变量的引用	(196)
9.1.4 结构体变量的初始化	(198)

9.2 结构体数组	(199)
9.3 指向结构体类型数据的指针	(200)
9.4 链表	(202)
9.4.1 链表概述	(203)
9.4.2 链表的基本操作	(205)
9.4.3 链表操作应用	(210)
9.5 共用体	(214)
9.5.1 共用体概念	(214)
9.5.2 共用体类型的应用	(216)
9.6 枚举类型	(217)
9.7 用 typedef 定义类型	(220)
本章小结	(221)
习题 9	(222)
第 10 章 文件	(223)
10.1 文件概述	(223)
10.2 文件指针	(224)
10.3 文件的打开与关闭	(224)
10.3.1 文件的打开	(224)
10.3.2 文件的关闭	(226)
10.4 文件的读写	(226)
10.4.1 fputc()函数和 fgetc()函数	(226)
10.4.2 fputs()函数和 fgets()函数	(228)
10.4.3 fprintf()函数和 fscanf()函数	(229)
10.4.4 fread()函数和 fwrite()函数	(230)
10.5 文件的定位	(231)
本章小结	(233)
习题 10	(234)
第 11 章 预处理命令	(235)
11.1 宏定义	(235)
11.1.1 无参数的宏定义	(235)
11.1.2 带参数的宏定义	(237)
11.2 文件包含	(238)
11.3 条件编译	(240)
本章小结	(242)
习题 11	(243)
第 12 章 位运算	(244)
12.1 位运算符和位运算	(244)
12.1.1 按位与运算符	(244)
12.1.2 按位或运算符	(246)
12.1.3 异或运算符	(247)

12.1.4 取反运算符.....	(248)
12.1.5 左移运算符.....	(248)
12.1.6 右移运算符.....	(249)
12.1.7 位运算赋值运算符.....	(250)
12.1.8 不同长度的数据进行位运算.....	(250)
12.2 位运算举例.....	(250)
12.3 位段.....	(252)
本章小结.....	(253)
习题 12	(253)
附录 A ASCII 代码与字符对照表	(254)
附录 B 运算符的优先级和结合性	(255)
附录 C C 语言库函数	(256)
附录 D C 语言中的关键字	(261)
附录 E 学生成绩管理系统参考程序	(262)
参考文献.....	(270)

第1章 C语言概述

【本章重点】 C语言程序的基本结构;在 Visual C++ 6.0 集成开发环境下编辑、编译、调试和运行 C 程序的基本方法和步骤。

【学习目标及方法】 了解 C 语言的发展历史及特性,掌握在 Visual C++ 6.0 集成开发环境下编辑、编译、调试和运行 C 程序的基本方法和步骤。主要通过上机调试和运行简单程序来分析理解 C 程序的基本结构,达到本章的学习目标。建议本章理论 2 学时,实验 2 学时。

【本章概论】 人们利用计算机解决实际问题,一般都要编写程序。本章将就程序设计的一般步骤、程序设计语言的种类、C 语言的主要特点及 C 程序调试的基本方法进行阐述,旨在帮助读者快速地了解 C 语言程序设计的主要内容。

1.1 程序与程序设计语言

1.1.1 程序

计算机程序是指导计算机执行某个功能或功能组合的一套指令。要使指令得到执行,计算机必须执行程序。也就是说,计算机要读取程序,然后按准确的顺序实施程序中编码的步骤,直至程序结束。一个程序可多次执行,而且如果用户每次输入给计算机的选项和数据不同,就有可能得到不同的结果。

程序设计语言是规定如何生成可被计算机处理和执行的指令的一系列语法规则。程序设计是编程者根据程序设计语言的语法规则,编写指令以指示计算机完成某些工作的过程。根据程序设计语言编写得到的指令称做代码,编程者编写的代码的集合称为源代码(或源程序)。

计算机程序设计的过程一般由 4 个步骤组成。

(1) 分析问题

在着手解决问题之前,应该通过分析充分理解问题,明确原始数据、解题要求、需要输出的数据及形式等。

(2) 设计算法

算法是一步步的解题过程。首先集中精力于算法的总体规划,然后逐层降低问题的抽象性,逐步充实细节,直到最终把抽象的问题具体化成可用程序语句表达的算法。这是一个自上而下、逐步细化的过程。

(3) 编码

利用程序设计语言表示算法的过程称为编码。程序是一个用程序设计语言通过编码实

现的算法。

(4) 调试程序

调试程序包括编译和链接等操作。编译程序对编程者编写的源程序进行语法检查,编程者根据编译过程中的错误提示信息,查找并改正源程序的错误后再重新编译,直到没有语法错误为止。编译程序将源程序转换为目标程序。大多数程序设计语言往往还要使用链接程序把目标程序与系统提供的库文件进行链接以得到最终的可执行文件。在链接过程中,若程序使用了错误的内部函数名,将会引起链接错误。对于经过成功编译和链接并最终顺利运行结束的程序,编程者还要对程序运行的结果进行分析,只有得到正确结果的程序才是正确的程序。

1.1.2 程序设计语言

为了让计算机解决一个实际问题,必须使用计算机的程序设计语言编写程序。程序设计语言使得人们能够与计算机进行交流。

程序设计语言种类繁多,一般可以划分为 4 类:机器语言、汇编语言、高级语言和面向对象的语言。

机器语言又称面向机器的语言,它的所有操作与数据都用二进制数表示。机器语言是特定的计算机硬件系统所固有的语言,是 CPU 唯一能够真正不经过翻译而直接识别和执行的语言。机器语言难于记忆和理解,因此,用机器语言编写程序繁琐且易出错。目前,除了编写计算机的核心程序外,一般不使用机器语言。

相比而言,其他任何语言编写的程序都必须最终转换成机器语言以后才能在 CPU 上执行。

汇编语言也称符号语言,它使用助记符来表示指令的操作码,用符号来表示地址和变量。由于引入了助记符,与机器语言相比,采用汇编语言编写程序要容易很多,可读性也更强,具有优越性。但因为汇编语言具有一个本质上与机器语言一一对应的指令系统,因此,与机器指令一样,汇编指令也直接针对计算机硬件进行操作,要求编程者具有深厚的计算机专业知识,源程序一般也比较冗长、复杂,且容易出错。

汇编语言和机器语言一样,依赖于具体的计算机指令系统,是面向机器的语言,因此,称它们为低级语言。

高级语言的出现使得编写程序变得方便。高级语言更接近人类的自然语言,易学易用,而且编程者不必了解计算机的硬件及指令系统,从而可以把主要精力放在算法描述上。高级语言编写的程序通用性好,可移植性强,不依赖具体的硬件设备。从 1954 年第一个完全脱离机器硬件的高级语言 FORTRAN 问世至今,共有几百种高级语言出现。影响较大、使用较普遍的有 FORTRAN、ALGOL、COBOL、BASIC、LISP、SNOBOL、PL/1、Pascal、C、PROLOG、Ada 等。高级语言的发展经历了从早期语言到结构化程序设计语言,从面向过程到非过程化程序语言的过程。相应地,软件的开发也由最初的个体手工作坊式的封闭式生产,发展为产业化、流水线式的工业化生产。

20 世纪 80 年代初,产生了面向对象的程序设计思想。在此之前的高级语言几乎都是面向过程的,程序的执行是流水线似的,无法动态地改变程序的执行方向。这和人们日常处理事物的方式是不一致的,人们希望不是面向过程,而是面向具体的应用功能。面向对象程序设计的方法就是设计一些通用的、封装紧密的功能模块(称为软件集成块),它与具体应用

无关,但能相互组合,完成具体的应用功能,同时又能重复使用。对使用者来说,只关心它的接口及能实现的功能,至于如何实现的,那是它内部的事,使用者完全不用关心。C++、Object Pascal、Java 等就是典型代表。

1.2 C语言发展概述和主要特点

1.2.1 C语言的发展历史

C语言功能强大、使用灵活、可移植性好,既可用于编写系统软件,也可用于编写应用软件。再者,C语言的语法规则清晰,便于掌握和记忆,因此,目前C语言已经成为绝大多数人学习计算机程序的入门基础课程。

C语言是由B语言发展而来的,而B语言又是由A语言发展而来。

A语言是指高级语言ALGOL 60。1960出现的ALGOL 60是一种面向问题的过程式高级语言。以前的操作系统等系统软件主要是用汇编语言编写的,但汇编语言的可读性和可移植性比较差,因此想改用高级语言。而ALGOL 60离硬件较远,不适宜用来编写系统软件。1963年英国的剑桥大学推出了CPL(Combined Programming Language)语言。CPL相比ALGOL 60更接近硬件,但规模较大,难于实现。1967年英国剑桥大学的Martin Richards对CPL语言进行了简化,即BCPL(Basic Combined Programming Language)语言。1970年美国贝尔实验室的K. Thompson在BCPL语言的基础上再次简化,设计出简单且很接近硬件的B语言(取BCPL的第一个字母),并用B语言编写了UNIX操作系统初版。但B语言过于简单,功能有限。1972年贝尔实验室的D. M. Ritchie在B语言的基础上设计出C语言(取BCPL的第二个字母),它保留了B语言的精练和接近硬件的优点,又克服了过于简单、无数据类型等缺点。1973年,K. Thompson和D. M. Ritchie合作用C语言改写了UNIX操作系统。到1977年,UNIX得到日益广泛的使用,同时C语言也迅速地得到推广。至今,C语言已经风靡世界,得到越来越多人的赞誉。

随着微型计算机的日益普及,出现了许多C语言版本。由于没有统一的标准,使得这些C语言之间出现了一些不一致的地方。为了改变这种情况,美国国家标准化协会(American National Standards Institute, ANSI)为C语言制定了一套ANSI标准,成为现行的C语言标准。

1.2.2 C语言的主要特点

C语言具有下列特点。

(1) C语言是中级语言

C语言通常被称为中级语言,是因为它把高级语言的成分同汇编语言的功能结合起来了。一方面,C语言具有高级语言面向用户、容易理解、便于阅读和书写的优点;另一方面,C语言不仅具有与内存地址对应的指针处理方式,可以直接处理内存中的各种类型的数据,而且能够进行位运算,可以实现汇编语言的大部分功能,可以直接对硬件进行操作,具有接近汇编语言程序执行的高效率。

(2) C 语言是结构化语言

C 语言具有表达 3 种基本结构(顺序、选择和循环)的流程控制语句和实现模块化的函数结构、函数调用,符合现代编程风格的要求。结构化语言程序清晰,更易于程序设计和维护。

(3) C 语言功能齐全

C 语言表达能力强大,具有丰富的运算符,应用范围广,表达式类型多样化,可以实现其他高级语言难以实现的运算。C 语言的数据类型丰富,能用来实现各种复杂的数据类型。尤其是引入了指针的概念,使得 C 语言更为灵活多样。

(4) C 语言简洁、紧凑,使用方便、灵活

C 语言一共只有 32 个关键字、9 种控制语句,程序书写形式较自由,压缩了不必要的成分。C 语言的语法限制不太严格,程序设计自由度大,非常灵活。

(5) C 语言具有较高的可移植性

C 语言程序本身并不依赖于计算机硬件系统和操作系统,因此在不同的硬件结构和运行不同操作系统的计算机间基本上不作修改就可实现程序的移植。C 语言提供的预处理命令可以提高软件开发效率,并为程序的组织和编译提供了便利,也提高了程序的可移植性。

上述所列 C 语言的特点都是它的优点,但某些特点若运用得不好则适得其反,这决定于编程者本身的编程能力。C 语言对编程者要求较高,如能熟练掌握,编程者使用 C 语言编写程序时会感到限制少、灵活性大、功能强,可以编写出任何类型的程序。

但对于初学者,由于 C 语言的自由、灵活,较之其他高级语言在学习上要困难一些。关于以上特点,初学者可以先了解一下,随着学习的深入,运用 C 语言的熟练程度提高,就能深刻体会到。

1.3 C 程序的基本结构

在使用 C 语言编写程序时必须按其规定的格式和提供的语句进行编写。下面通过几个简单的 C 语言程序,来介绍 C 程序的基本结构。

例 1.1 编写一个简单的 C 程序,用于输出指定信息。

```
#include <stdio.h>
main()          /* 定义主函数 */
{
    printf("Hello, world! \n");
}
```

运行结果为:

```
Hello, world!
```

该程序的第 1 行 main() 表示主函数,每一个 C 程序都必须有一个 main() 函数。主函数名 main 后跟了一对圆括号, C 程序中的函数,其函数名后必须跟一对圆括号。程序的第 2~4 行是函数体部分,函数体是用一对大括号括住的若干条语句。程序的第 3 行为一个函

数调用语句,调用 printf()函数来输出数据,该函数的使用将在第2章中详细介绍。程序中的分号是C语言的语句结束标志。/* */表示注释部分,注释是对语句的说明,对程序的运行不起任何作用。

下面程序主函数的函数体由多条语句组成。

例 1.2 求半径为 2.5 的圆的面积。

```
#include <stdio.h>
main()
{
    float r,s;                /* 定义实数类型变量 r,s */
    r=2.5;                    /* 圆的半径值为 2.5 */
    s=3.14159 * r * r;        /* 求圆的面积值 s */
    printf("area is:%f\n",s); /* 输出圆的面积值 */
}
```

运行结果为:

```
area is:19.634937
```

本程序的第4行为定义变量部分,这里定义了两个实数类型(float)变量 r 和 s。第5行是使变量 r 的值为 2.5 的赋值语句。第6行仍然为赋值语句,根据圆面积的计算公式将该圆的面积值计算出来赋给 s。第7行调用 printf()函数来输出变量 s 的值,“area is:”是等待输出的提示信息,“%f”指定 s 值的输出格式为小数形式的实数,“\n”是输出回车换行符。

再来看一个较为复杂的C程序,它由多个函数组成。

例 1.3 求圆的周长和面积,圆的半径值由键盘输入。

```
#include <stdio.h>
/* 定义 circumference()函数,求圆的周长 */
float circumference(float r) /* 函数首部,circumference 为函数名,函数值为实型,形式参数 r 为实型 */
{
    float c;                /* 函数中的声明部分,定义本函数中用到的变量 c 为实型 */
    c=2 * 3.14159 * r;
    return(c);              /* 将 c 的值传回到调用处 */
}
/* 定义 area()函数,求圆的面积 */
float area(float r)
{
    float s;
    s=3.14159 * r * r;
    return(s);
}
/* main()函数实现半径值的输入,然后调用上述两个函数求出圆的周长及面积并输出 */
main()
{
```

```
float radius,cir,s;  
scanf("%f",&radius);  
cir=circumference(radius);          /* 调用 circumference()函数求圆的周长 */  
s=area(radius);                     /* 调用 area()函数求圆的面积 */  
printf("circumference is:%f,area is:%f\n",cir,s);  
}
```

运行结果为:

```
3 ✓  
circumference is:18.849541,area is:28.274309
```

本程序由 `circumference()`、`area()` 和 `main()` 3 个函数构成,它们各实现一定的功能。

`circumference()` 函数的作用是求出半径为 r 的圆周长的值并将其赋值给 c 。`return` 语句将 c 的值返回到主调函数 `main()` 中。返回的值是通过函数名 `circumference` 带回到 `main()` 函数的调用处。同样,`area()` 函数的作用是求半径为 r 的圆面积 s 值并返回。

主函数中调用 `scanf()` 函数来输入变量 `radius` 的值。`&radius` 的“&”是取地址运算符,将键盘输入的实型数据的值存入变量 `radius` 的地址所标志的内存存储单元中。`main()` 函数中第 5 行调用 `circumference()` 函数,在调用时,将实际参数 `radius` 的值传递给 `circumference()` 函数的形式参数 r 。执行该函数得到一个由 `return(c)` 传回的返回值,该值赋值给 `cir`。主函数 `main()` 无论在什么位置上,程序总是从它开始执行的。

本程序中提到了函数调用、形式参数和实际参数等概念,读者不必深究,在后续章节中会学习到相关知识,本例只是简要介绍 C 程序的组成形式。

上面列举了 3 个 C 程序,相信大家已经对 C 程序有了一个初步的了解,可以看到:

① C 程序是由函数构成的。每个 C 程序必须有一个并且只能有一个称为主函数的 `main()` 函数,除主函数外,可以没有其他函数(如例 1.1、例 1.2),也可以有一个或多个其他函数(如例 1.3)。因此函数是构成 C 程序的基本单位。编写 C 程序就是编写一个个函数。

② 函数的定义分为两部分:函数首部和函数体。以例 1.3 中的函数 `area()` 为例:

<pre>float area(float r) { float s; s=3.14159 * r * r; return(s); }</pre>	<pre>函数首部 函数体的开始 函数体的声明部分 函数体的执行部分 函数体的结束</pre>
---	---

函数首部是定义一个函数的开始,包括函数类型、函数名、函数形式参数表。函数名后面必须跟上一对圆括号,函数可以没有参数。最简单的函数首部的形式如 `main()`。

函数体是函数首部下面用一对大括号括起的部分,是函数功能的具体实现,包括声明部分和执行部分。声明部分中定义在本函数内部使用到的变量,此部分还可能对所调用的函数进行声明;执行部分由 C 语句构成,用来完成函数所要实现的功能。

因此,一般来说,函数定义的格式如下:

```
类型名 函数名(形式参数表)  
{  
    声明部分
```

执行部分

```
}
```

但根据实际情况,声明部分是可以没有的,甚至连执行部分也可以没有。例如:

```
blankfun()
```

```
{
```

```
}
```

这是一个空函数,什么也不做,没有意义,但是合法。

注意:每个函数的定义必须是相互独立的,不允许在函数体内部定义其他函数,即函数不允许嵌套定义。

③ C程序的执行总是从主函数开始,并在主函数中结束。主函数定义的位置是任意的,可以在程序的开头,也可以在某两个函数定义之间或程序的结尾。

④ C语句的最后必须以分号(;)结束(复合语句除外)。C程序书写格式自由,一条语句可以分多行写,一行上也可以写多条语句。除非是程序流程控制语句(如 while 语句)等较复杂的语句,较简单的语句建议不要分行写。

⑤ C语言严格区分大小写。C语言习惯上用小写字母书写,只有符号常量、宏定义等习惯用大写,所有的关键字必须小写,如 if、else、int 等只能小写。

注意:If 不是关键字,Sum 和 sum 不是同一个变量名。

⑥ /* */表示注释。一个好的源程序应当加上必要的注释,以增强程序的可读性。注释通常放在一段程序的开始,用以说明该段程序的功能,或者放在某个语句的后面,对该语句进行说明。编译源程序时,注解部分将不被编译。

初学者不需要死记上述这么多的格式要求,因为随着对C语言的深入学习,C语言的书写格式在无形中会被领悟和掌握的。

1.4 程序的调试

在编写完一个C程序后,还需要检查程序中的语法错误,运行程序并查看运行结果是否正确,如果存在错误,反复修改错误直至得到正确的运行结果,这个过程称为程序的调试。

1.4.1 调试步骤

从编写完程序到上机运行,一般要经过以下几个步骤。

上机输入、编辑源程序→对源程序进行编译→与库函数链接→运行目标程序

此过程如图 1-1 所示。

(1) 编辑源程序

首先将编写的C语言源程序输入到计算机中以文件的形式保存起来,C语言源程序的扩展名为“c”或“cpp”,如 file.c 或 file.cpp。C语言源程序为文本文件,可以用文本编辑器(如记事本)编辑,也可以用C编译系统集成的编辑器编辑。

(2) 编译

C语言源程序编辑好后,应该进行编译的工作,对编写的源程序进行语法检查。如果源