



普通高等教育“十二五”规划教材

C 语言程序设计

(第二版)

主 编 任正云 李素若 赖 玲
副主编 琚 辉 胡玉荣 张 牧 肖 衡
主 审 田 原



中国水利水电出版社
www.waterpub.com.cn

普通高等教育“十二五”规划教材

C 语言程序设计（第二版）

主 编 任正云 李素若 赖 玲

副主编 琚 辉 胡玉荣 张 牧 肖 衡

主 审 田 原



中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书遵照 C 语言标准,结合作者多年的教学和科研实践的经验和体会,全面系统、深入浅出地阐述了 C 语言的基本概念、语法和语义,以及用 C 语言进行程序设计的基本方法和技巧。

本次再版以 Visual C++ 6.0 环境为基础进行修订,主要内容包括数据类型和表达式、流程控制、算法分析、函数与程序结构等。内容更丰富,叙述更详细,描述更加准确。每章末都配有习题可供不同层次的读者练习。

本书十分注重知识的应用,重点章节都给出了应用举例。本书的一个亮点是给出了学生成绩管理系统、大奖赛评分系统和万年历的程序,虽然所给出程序代码不一定最优化,但可以引导读者分析,给读者以启发,是学习编程人员一本很好的工具书。

本书是一本准确而又较全面反映标准 C 语言的教材,还配有《C 语言程序设计上机指导与习题集解答》(第二版)一书。既可供高等院校计算机和非计算机专业本、专科或培训班教学使用,也是广大科技工作者和编程爱好者的一本很好的参考书。

本书提供电子教案,读者可以从中国水利水电出版社网站以及万水书苑下载,网址为:
<http://www.waterpub.com.cn/softdown/>或 <http://www.wsbookshow.com>。

图书在版编目(CIP)数据

C语言程序设计 / 任正云,李素若,赖玲主编. — 2
版. — 北京:中国水利水电出版社,2011.10
普通高等教育“十二五”规划教材
ISBN 978-7-5084-9034-2

I. ①C… II. ①任… ②李… ③赖… III. ①
C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(2011)第198598号

策划编辑:杨庆川 责任编辑:杨元泓 加工编辑:韩莹琳 封面设计:李 佳

书 名	普通高等教育“十二五”规划教材 C 语言程序设计(第二版)
作 者	主 编 任正云 李素若 赖 玲 副主编 琚 辉 胡玉荣 张 牧 肖 衡 主 审 田 原
出版发行	中国水利水电出版社 (北京市海淀区玉渊潭南路 1 号 D 座 100038) 网址: www.waterpub.com.cn E-mail: mchannel@263.net (万水) sales@waterpub.com.cn
经 售	电话: (010) 68367658 (发行部)、82562819 (万水) 北京科水图书销售中心(零售) 电话: (010) 88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排 版	北京万水电子信息有限公司
印 刷	三河市鑫金马印装有限公司
规 格	184mm×260mm 16 开本 18.5 印张 462 千字
版 次	2007 年 3 月第 1 版第 1 次印刷 2011 年 11 月第 2 版 2011 年 11 月第 1 次印刷
印 数	0001—4000 册
定 价	32.00 元

凡购买我社图书,如有缺页、倒页、脱页的,本社发行部负责调换
版权所有·侵权必究

第二版前言

本书出版以来,经过多年的使用,得到了广大教师和读者的肯定,特别是2010年被教育部高等学校高职高专计算机类专业教学指导委员会评为优秀教材,根据大多数教师和读者的要求,结合广大读者的宝贵意见,在第一版的基础上进行了如下几个方面的改进:

(1) 将上机环境改为 Visual C++ 6.0,基本数据类型等内容作相应的修改和调整。

(2) 语法讲解更注意通俗易懂,知识的体系结构更符合认知规律。在涉及语法的讲解过程中,除了给出语法知识的讲解外,每个知识点都给出了例题和案例,突出应用特性,把程序设计中 most 基本、最新、最有价值的思想和方法渗透到经典的 C 语言中,其目的就是使读者在学习 C 语言之后,今后无论使用什么语言编程,都能使用这些方法和思想。

(3) 力求暴露程序设计的思维过程,用最简明的语言、最经典的实例,采用类比的方法来挖掘程序设计的精髓,给读者带来全新的学习体验,领悟 C 语言之妙。

(4) 适当增加了一些例题和习题。

(5) 本书的配套教材《C 语言程序设计上机指导与习题解答》(第二版)作了全面的改版,编译环境只介绍了 Visual C++ 6.0,增加了计算机等级考试中的相关内容和上机题例,选编了一些例题和习题,每个例题和习题都给出了详细的解答,尽管不是最佳方法,但给读者学习 C 语言一定会有不小的帮助。

总之,通过这次修订,本书内容更加丰富,叙述更加详细,描述更加准确,更加有利于读者的学习。

本次修订由任正云、李素若、赖玲担任主编,琚辉、胡玉荣、张牧、肖衡担任副主编,全书的第 1、2、3、4、5、6 章由任正云编写,第 7 章由琚辉编写,第 8、9、10 章由赖玲编写,胡玉荣参与了第 4、6 章的修订,李素若参与了第 8 章的修订,张牧参与了第 9 章的修订,肖衡参与了第 10 章的修订,全书由任正云统稿,赖玲和琚辉负责全书的校对,所有的程序由王娅纷、王晓雨、严永松、琚辉和陈万华负责调试,在编写过程中得到了主审田原教授和相关专家的指导,在此一并表示衷心的感谢。

由于作者水平有限,书中难免还会存在不足之处,再次恳请广大读者批评指正。

编 者

2011 年 12 月

第一版前言

C 语言是一种结构化的程序设计语言。它功能丰富、表现力强、使用灵活、应用面广、目标程序效率高、可移植性好，既具有高级语言的特点，又具有低级语言的功能，因此它成为当今的主流程序设计语言之一。由于它简单易学，使用方便灵活，所以学习和使用 C 语言的人越来越多，国内高等院校理工科专业都开设了这门课程。同时，C 语言也是全国计算机二级考试的指定考试科目之一。学好 C 语言对进一步学习其他计算机语言具有积极的意义。

C 语言程序设计是一门实践性很强的课程，它包含理论学习、编程方法和程序调试三个方面的内容。由于它的语法现象比较复杂，数据类型转换和表示灵活多变，因此，在学习和掌握 C 语言时，要把实践分为三个层次和两个方面。三个层次是：阅读别人写好的程序（或函数），理解程序所要完成的任务（也就是程序的功能），从中学习编程的方法和技巧；模仿编写功能类似的程序；自己独立设计和编写完成指定任务的程序。两个方面是：在条件有限的情况下，动手在纸上严格按语法规则一丝不苟地写出程序；另一方面，在条件允许的情况下，应该尽量上机练习，调试自己所写的程序。根据当前的形式和教学的需要，从 C 语言教学实际出发，我们编写了这本《C 语言程序设计》，希望本书能为广大读者提供有益的帮助。

针对 C 语言在计算机专业课程以及计算机公共基础课程体系中的地位，本书从培养学生的理解、设计基本算法出发，结合掌握 C 语言的语法规则训练，培养学生基本编程能力。本书在编写过程中，着重体现以下特色：

（1）语言通俗易懂、结构符合教学规律。考虑到学习者的基础，本书在编写过程中从语言的角度尽量做到通俗易懂，避免按 C 语言说明书、操作手册的内容安排和描述，内容及讲解由浅入深，符合 C 语言的特点，符合程序设计语言学习的特点。

（2）理论讲解力求体现“必需、够用为度”。充分考虑学生的特点和 C 语言程序设计在课程体系中的地位，在内容的讲解上，尽量形象地描述算法产生的过程，突出学习重点，理论教学力求体现必需、够用为度，强调实际应用，有意回避一些 C 语言中出现频率很低或与语言实现版本有关的内容，把重点放在语言本身的难点和程序设计的技巧方面，为以后从事软件开发的学生提供良好的参考。

（3）坚持“两个并重”，重视学生技能的形成过程。所谓两个并重，就是程序设计语言和程序设计技巧并重、典型案例和实际编程并重。本书在讲解理论后都有强化理论的例题，每章结束部分都安排有应用实例。这样做的目的就是力求使读者学完 C 语言程序设计之后，不仅能懂 C 语言的语法、语义，更重要的是具备编程解决实际问题的能力，通过给出的一些典型案例，让学生能够从中借鉴、模仿及改写，从而提高学习者的编程能力。

特别值得一提的是：本书在讲解必要的理论知识的同时，十分注重知识的应用，重点章节都给出了应用举例，本书的一个亮点是给出了学生成绩管理系统，大奖赛评分系统，万年历的程序，是学习编程人员的一本很好的工具书。本书还配有《C 语言程序设计上机指导与习题集解答》一书，供不同层次的读者练习。

本书由任正云、李素若任主编，胡玉荣、张牧、肖衡任副主编，具体分工如下：第 1、2、

3、5、7 章由任正云编写，第 4、6 章由胡玉荣编写，第 8 章由李素若编写，第 9 章由张牧编写，第 10 章由肖衡编写，全书由任正云统稿，所有的程序由张牧、胡玉荣负责调试，在编写过程中得到了主审田原副教授和相关专家的指导，在此一并表示衷心的感谢。

本书存在一些不足之处，竭诚希望广大读者批评指正。

编 者

2006 年 12 月

目 录

第二版前言

第一版前言

第1章 C语言程序设计基础	1	3.1.3 自顶向下、逐步细化的设计过程	44
1.1 程序设计及程序设计语言	1	3.2 C语句概述	46
1.1.1 程序设计语言的发展	1	3.2.1 控制语句	46
1.1.2 C语言的发展过程	2	3.2.2 表达式语句	46
1.2 C语言的特点	3	3.2.3 复合语句	47
1.3 C程序结构	4	3.3 数据的输入和输出	48
1.3.1 简单的C程序介绍	4	3.3.1 数据的输出函数	48
1.3.2 C程序结构	6	3.3.2 scanf函数	53
1.4 源程序的编辑、编译、连接与运行	7	3.4 getchar函数与 putchar函数	57
习题	8	3.4.1 字符输出函数 putchar()	57
第2章 C语言的数据类型与基本操作	10	3.4.2 字符输入函数 getchar()	57
2.1 常量与变量	10	习题	58
2.1.1 常量	10	第4章 选择结构程序设计	63
2.1.2 变量	13	4.1 if语句	63
2.2 标识符和关键字	15	4.1.1 if语句的形式	63
2.3 整型数在计算机中的存储方式	16	4.1.2 if语句的嵌套	67
2.4 有符号的数据类型和无符号的数据类型	18	4.2 switch语句	69
2.5 运算符和表达式	19	4.3 程序举例	71
2.5.1 赋值运算符和赋值表达式	21	习题	75
2.5.2 算术运算符和算术表达式	22	第5章 循环结构程序设计	79
2.5.3 长度测试运算符 sizeof	23	5.1 while语句	79
2.5.4 关系运算符和关系表达式	24	5.2 do-while语句	84
2.5.5 逻辑运算符与逻辑表达式	25	5.3 for语句	86
2.5.6 条件运算符与条件运算表达式	27	5.4 三种循环语句的比较	90
2.5.7 逗号运算符与逗号表达式	29	5.5 循环的嵌套	90
2.5.8 位运算	29	5.6 break语句和 continue语句	95
2.6 不同类型数据间的转换	32	5.6.1 break语句	95
习题	34	5.6.2 continue语句	96
第3章 顺序结构程序设计	40	5.7 综合实例	96
3.1 算法	40	5.7.1 列举算法	96
3.1.1 算法的组成要素	40	5.7.2 试探算法	98
3.1.2 算法的表示方法	41	5.7.3 密码问题	99

习题	100	7.2.4 二维数组的经典实例	160
第6章 函数	106	7.3 字符数组	163
6.1 函数概述	106	7.3.1 字符数组的定义	163
6.1.1 模块化程序设计方法	106	7.3.2 字符数组的输入输出	164
6.1.2 函数的概念	106	7.3.3 常用字符串函数	165
6.2 函数的定义与声明	108	7.3.4 字符数组的使用	167
6.2.1 函数定义	109	7.4 数组应用实例	168
6.2.2 函数的参数和返回值	110	7.4.1 排序	168
6.2.3 函数的声明	110	7.4.2 二分查找	172
6.3 函数的调用	111	习题	173
6.3.1 调用函数的一般形式	111	第8章 指针	178
6.3.2 调用函数时数据的传递	111	8.1 地址和指针的概念	178
6.3.3 函数的嵌套调用	113	8.2 指向变量的指针变量	179
6.3.4 函数的递归调用	114	8.2.1 指向变量的指针变量的定义	179
6.4 局部变量和全局变量	117	8.2.2 指针运算符	179
6.4.1 局部变量	117	8.2.3 指针变量的初始化	181
6.4.2 全局变量	118	8.2.4 指针变量的运算	181
6.5 变量的存储属性	121	8.2.5 指针变量作为函数参数	183
6.5.1 自动变量(auto)	122	8.2.6 程序实例	184
6.5.2 寄存器变量(register)	124	8.3 指针与数组	186
6.5.3 静态变量(static)	124	8.3.1 一维数组和指针	186
6.5.4 外部变量	125	8.3.2 二维数组和指针	190
6.6 编译预处理	126	8.3.3 程序实例	192
6.6.1 宏定义	127	8.4 字符串的指针和指向字符串的指针变量	193
6.6.2 文件包含	132	8.4.1 字符串的指针	193
6.6.3 条件编译	133	8.4.2 字符串作为函数参数	195
6.7 应用举例	135	8.5 指针数组	196
习题	142	8.6 指向指针的指针	197
第7章 数组	150	8.6.1 指向指针的指针的概念	197
7.1 一维数组	150	8.6.2 指向指针的指针的应用	198
7.1.1 一维数组的定义	150	8.7 函数的指针	200
7.1.2 一维数组的初始化	151	8.7.1 函数的指针和指向函数的指针变量	200
7.1.3 一维数组元素的引用	151	8.7.2 函数的指针作为函数参数	202
7.1.4 一维数组的使用	151	8.7.3 函数指针的应用	202
7.2 二维数组和多维数组	156	8.8 返回指针的函数	204
7.2.1 二维数组和多维数组的概念及其定义	156	8.9 main()函数的返回值和参数	206
7.2.2 二维数组的初始化	158	8.9.1 main()函数的返回值	206
7.2.3 二维数组和多维数组的引用	159		

8.9.2	main()函数的参数	206	10.2	格式化数据文件操作	258
习题		207	10.2.1	打开文件函数 fopen()	258
第9章	结构体和共用体	211	10.2.2	关闭文件函数 fclose()	260
9.1	概述	211	10.2.3	从文件中格式化输入数据 fscanf()	260
9.2	结构体与结构体类型变量	211	10.2.4	向文件中格式化输出函数 fprintf()	260
9.2.1	结构体类型的定义	211	10.2.5	文件建立	261
9.2.2	结构体类型变量的定义	212	10.2.6	文件访问	262
9.2.3	结构体变量的引用	214	10.2.7	文件修改	262
9.2.4	结构体变量的初始化	215	10.3	字符数据文件操作	264
9.3	结构体数组	216	10.3.1	从指定文件中读取一个字符	264
9.4	指向结构体类型数据的指针	219	10.3.2	putc()函数——指向文件输出一个字符	265
9.4.1	指向结构体变量的指针	219	10.3.3	fgets()函数——从文件中读一个字符串	265
9.4.2	指向结构体数组的指针	220	10.3.4	fputs()函数——向指定文件输出一个字符串	265
9.4.3	结构体指针变量作为函数的参数	221	10.3.5	feof()函数——文件操作是否到文件尾的检测函数	266
9.4.4	结构体与函数的类型	223	10.4	fread()与 fwrite()函数	267
9.5	动态存储分配与链表	225	10.5	文件位置定位与错误处理函数	268
9.5.1	链表的概念	225	10.5.1	rewind()函数——将文件读写指针定位于文件头	268
9.5.2	动态存储分配	226	10.5.2	fseek()函数——重新定位文件位置指针	269
9.5.3	建立和输出链表	227	10.5.3	ftell()函数——测试当前读写指针位置	269
9.5.4	链表的基本操作	228	10.5.4	ferror()函数——检测文件流操作中的错误	269
9.6	共用体	235	10.5.5	clearerr()函数——清除出错标志	269
9.6.1	共用体的概念和定义	235	10.6	应用实例——大奖赛评分系统	269
9.6.2	共用体变量的引用	237	习题		274
9.7	枚举类型	239	附录A	ASCII 字符编码一览表	278
9.7.1	枚举的定义与说明	239	附录B	C 语言库函数	279
9.7.2	枚举类型变量的赋值与引用	240	参考文献		285
9.8	用 Typedef 定义类型	241			
9.9	应用实例——学生成绩管理系统	242			
习题		252			
第10章	文件	256			
10.1	文件的概念	256			
10.1.1	文件的类型	256			
10.1.2	C 语言对文件的处理方法	257			
10.1.3	文件的访问方式	257			

第 1 章 C 语言程序设计基础

1.1 程序设计及程序设计语言

程序是能被机器识别并能被执行的一系列的指令代码，这些指令代码是用程序设计语言来描述的。程序设计语言是人与计算机对话的工具。程序设计需要在一定程序设计语言环境下进行。

1.1.1 程序设计语言的发展

从计算机诞生到现在，程序设计语言也伴随着计算机技术不断地升级换代。程序设计语言是计算机软件系统的重要组成部分，程序设计语言一般分为机器语言、汇编语言和高级语言三大类。

1. 低级语言

低级语言一般包括机器语言和汇编语言。

机器语言用二进制代码表示机器指令和数据。下面是某 CPU 指令系统中的两条指令：

10000000（进行一次加法运算）

10010000（进行一次减法运算）

用机器语言编写程序，就是从所使用的 CPU 指令系统中挑选出合适指令，组成一个指令系列。利用机器语言编写的程序能够直接被机器理解和执行，因而程序效率高，但编程烦琐，而且不便于记忆和阅读，程序维护困难。

为了减轻编写程序的劳动强度，20 世纪 50 年代中期人们开始用一些“助记符”来代替 0、1 码进行编程。如前面的两条机器指令可以写成

$A+B \Rightarrow A$ 或 ADD A, B

$A-B \Rightarrow A$ 或 SUB A, B

这种用助记符号描述的指令系统称为符号语言或汇编语言。

用汇编语言编写的程序比机器语言易于理解与记忆，但在 CPU 执行之前需要先转换为机器能够识别的二进制代码。完成这种转换任务的系统软件称为汇编程序，这种转换的过程称为汇编。低级语言是面向机器的，依 CPU 的不同而不同。用低级语言写的程序效率高，但可移植性差，程序员不仅要考虑解题的思路，还要熟悉机器的内部结构，并且还要“手工”地进行存储器分配，这种编程劳动强度很大，给计算机的普及与推广造成了很大的障碍。

2. 高级语言

高级语言是相对于低级语言而言的，它以人类易于接受的文字来表达解决问题的全过程。用户不必费时费力去记忆难以理解的 0、1 代码，而是可以把主要精力集中于解决问题的算法上。高级语言分为面向过程的程序设计语言和面向对象的程序设计语言，如 C 语言和 Pascal 语言、FoxBase、Visual C++、Visual Basic 等。

（1）面向过程的程序设计语言。汇编语言和机器语言都是面向机器的，随机器而异，1954 年出现的 FORTRAN 语言以及随后相继出现的其他计算机语言，使人们开始摆脱进行程序设计前必须先熟悉机器的桎梏，把精力集中在解题的思路和方法上，使程序设计语言开始与解题

方法相结合。其中一种就是把解题过程看作数据被加工的过程。基于这种方法的程序设计语言称为面向过程的设计语言, C 语言就是一种面向过程的设计语言。下面是一个计算长方形面积的 C 语言程序设计的片段:

```
main()                /*告诉编译器 C 程序由此开始执行*/
{                    /*这一段程序开始*/
    float a,b;        /*定义两个实型变量长 a 和宽 b */
    float area;       /*定义面积变量 area*/
    scanf("%f,%f",&a,&b) /*从键盘输入 a、b 的值*/
    area=a*b;         /*把计算的面积赋给 area*/
    printf("%f ", area); /*输出面积 area 的值*/
}                    /*程序结束*/
```

这个程序是很好理解的, 其中计算面积的语句与我们习惯的数学式子没有什么根本的区别 (“/*” 与 “*/” 之间的内容称为注释, 其目的是为了阅读的方便, 让程序更容易被理解)。显然使用高级语言编程可以较大地降低编程过程中的劳动强度, 提高编程效率, 高级语言的诞生是计算机发展史上的一个里程碑。它使人们能摆脱具体机器指令系统的束缚, 用接近人们习惯的语言来构思解题过程, 从而大大提高了编程效率, 使人们能够编制出规模越来越大的程序, 以满足日益广泛而深入的应用需求。

(2) 面向对象的程序设计语言。面向对象的程序设计是一种结构模拟方法, 它把现实世界看成是由许多对象 (Object) 组成, 对象之间通过相互发送和接收消息进行联系。消息的发送是对象本身的运动, 形成对象状态的变化。从程序结构的角度, 每个对象都是一个数据和方法的封装体——抽象数据类型。

从分类学的观点来看, 客观世界中的对象都可以分类。也就是说, 所有的对象都属于特定的类 (Class), 或者说每个对象都是类的一个实例。因而面向对象的程序设计的一个关键是定义 “类”, 并由 “类” 生成 “对象”。

面向对象的程序比面向过程的程序更清晰易懂, 适宜编写更大规模的程序, 成为了当代程序设计的主流。面向对象的程序设计语言有 Java、Visual Basic、Visual Basic.NET 等, 由 C 派生出的 C++ 语言也属于面向对象的程序设计语言, 它是一种多范型程序设计语言, 不仅可以利用它编写面向对象的程序设计语言, 还可以用它编写面向过程的程序。

1.1.2 C 语言的发展过程

C 语言是目前世界上流行最广泛的高级程序设计语言。C 语言的发展过程可粗略地分为三个阶段: 1970~1973 年为诞生阶段, 1973~1988 年为发展阶段, 1988 年以后为成熟阶段。

1. C 语言的诞生

C 语言是为写 UNIX 操作系统而诞生的。1970 年美国 AT&T 公司贝尔实验室 Ken Thompson 为实现 UNIX 操作系统而提出一种仅供自己使用的工作语言, 由于该工作语言是基于 1967 年英国剑桥大学的 Martin Richards 提出的 BCPL 语言设计的, 因而被作者命名为 B 语言, B 取自 BCPL 的第一个字母。B 语言被用于在 PDP-7 计算机上实现第一个 UNIX 操作系统。1972 年贝尔实验室的 Dennis M.Ritchie 又在 B 语言基础上系统地引入了各种数据类型, 从而使 B 语言的数据结构类型化。1973 年 K.Tompson 和 D.M.Ritchie 用 C 语言重写了 UNIX 操作系统, 推出 UNIX v5, 1975 年又推出 UNIX v6。此时的 C 语言是附属于 UNIX 操作系统的。

2. C 语言的发展

为了使 UNIX 操作系统能够在别的机器上得到推广, 1977 年 C 语言的作者发表了不依赖于具体机器系统的 C 语言编译文本《可移植 C 语言编译程序》, 从而推动了 UNIX 操作系统在

各种机器上的实现以及 UNIX 操作系统的不断发展。1978 年以后相继推出了 UNIX v7, UNIX systemV。UNIX 操作系统的巨大成功和广泛使用,使人们普遍注意到 C 语言的突出优点,从而又促进了 C 语言的迅速推广。同时, C 语言也伴随着 UNIX 操作系统的发展而不断发展。1978 年 Brian W.Kernighan 和 D.M.Ritchie 以 UNIX v7 中的编译程序为基础写了影响深远的名著《The C Programming Language》,这本书介绍的 C 语言是以后各种 C 语言版本的基础,被称为传统 C 语言。1978 年以后, C 语言先后移植到各种大型机、中型机、小型机及微型机上。目前, C 语言成为世界上使用最广泛的高级程序设计语言,且不依赖于 UNIX 操作系统而独立存在。

3. C 语言的成熟

1978 年以后, C 语言不断发展产生了各种 C 语言版本,不同的 C 语言版本对传统 C 语言都有所扩充和发展。1983 年,美国国家标准协会(ANSI)综合了各版本对 C 的扩充和发展,制定了新标准,称为 ANSI C。Kernighan 和 D.M.Ritchie 按 ANSI C 标准重写了他们的经典著作,于 1990 年正式发表了国际标准化组织(ISO)公布的 C 语言标准。C 语言标准的制定标志着 C 语言的成熟,1988 年以后推出的各种 C 语言版本与 ANSI C 是相容的。

1.2 C 语言的特点

一种语言之所以能存在和发展,并具有旺盛的生命力,总是有其不同于(或优于)其他语言的特点。C 语言的主要特点如下:

(1) 语言简洁、紧凑,使用方便、灵活。C 语言共有 32 个关键字,9 种控制语句,程序书写形式自由,主要用小写字母表示,压缩了一切不必要的成分。下面将 C 与 Pascal 语言作一比较,如表 1-1 所示。

表 1-1 C 与 Pascal 语言的比较

C 语言	Pascal 语言	含义
{ }	BEGIN...END	复合语句
if(e) S	IF(e) THEN S	条件语句
int i;	VAR i:INTEGER	定义 i 为整型变量
int a[10];	VAR a:ARRAY[1..10]OF INTEGER	定义 a 为整型一维数组
int f();	FUNCTION f():INTEGER	定义 f 为返回整型值的函数
int *p;	VAR p: ↑ INTEGER	定义 p 为指向整型变量的指针变量
i+=2;	i:=i+2	赋值语句,使 $i+2 \Rightarrow i$
i++,++i	i:=i+1	i 自增值 1, $i+1 \Rightarrow i$

学过 Pascal 的读者可以看到, C 程序比 Pascal 简练,源程序短,因此输入程序时工作量少。

(2) 运算符丰富。C 语言的运算符包含的范围很广泛,共有 34 种运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理,从而使其运算类型极其丰富,表达式类型多样化。灵活使用各种运算符可以实现在其他高级语言中难以实现的运算。

(3) 数据结构丰富,具有现代化语言的各种数据结构。C 语言的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等。能用来实现各种复杂的数

据结构（如链表、树、栈等）的运算，尤其是指针类型数据，使用起来比 Pascal 语言更为灵活、多样。

（4）具有结构化的控制语句（如 if...else 语句、while 语句、do...while 语句、switch 语句、for 语句）。用函数作为程序的模块单位，便于实现程序的模块化。C 语言是良好的结构化语言，符合现代编程风格的要求。

（5）语法限制不太严格，程序设计自由度大。例如对数组下标越界不做检查，由程序编写者自己保证程序的正确。对变量的类型使用比较灵活，例如整型数据与字符型数据可以通用。一般的高级语言语法检查比较严，能检查出几乎所有的语法错误。而 C 语言允许程序编写者有较大的自由度，放宽了语法检查。程序员应当仔细检查程序，保证其正确性，不要过分依赖 C 编译程序去查错。“限制”与“灵活”是一对矛盾。限制严格，就失去灵活性；而强调灵活，就必然放松限制。一个不熟练的编程人员，编写一个正确的 C 程序可能会比编写其他高级语言的程序难一些。也就是说，对用 C 语言编程的人，要求对程序设计更熟练一些。

（6）C 语言能进行位（bit）操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作。因此，C 语言既具有高级语言的特点，又具有低级语言的许多功能，可用来编写系统软件。C 语言的这种双重性，使它既是成功的系统描述语言，又是通用的程序设计语言。

（7）生成目标代码质量高，程序执行效率高。一般只比汇编程序生成的目标代码效率低 10%~20%。

（8）程序可移植性好（与汇编语言比）。基本上不做修改就能用于各种型号的计算机和各种操作系统。

上面只介绍了 C 语言最容易理解的一般特点，至于 C 语言应用内部的其他特点将结合以后各章的内容作介绍。由于 C 语言的这些优点，使 C 语言应用面很广。许多大的软件都用 C 语言编写，都是基于 C 语言的可移植性好和硬件控制能力高、表达和运算能力强的特点。许多以前只能用汇编语言处理的问题，现在可以改用 C 语言来处理了。C 语言对程序员的要求较高，程序员使用 C 语言编写程序会感到限制少，灵活性大，功能强，可以编写出任何类型的程序。现在，C 语言已不仅用来编写系统软件，也用来编写应用软件。

1.3 C 程序结构

1.3.1 简单的 C 程序介绍

【例 1.1】C 程序输出。

```
main()
{
    printf("This is a C program.\n");
}
```

运行结果为：

This is a C program.

说明：

（1）main 表示“主函数”。每个 C 语言程序都必须有一个 main 函数，它是每一个 C 程序的执行起始点（入口点）。main()表示“主函数”main 的函数头。

（2）用 {} 括起来的是“主函数”main 的函数体。main 函数中的所有操作（或语句）都在这一对 {} 之间，也就是说 main 函数的所有操作都在 main 函数体中。

(3) “主函数”main 中只有一条语句,它调用了 C 语言的库函数(见附录 B),功能是用程序的输出(显示在屏幕上),本例中将一个字符串“This is a C program.\n”的内容除格式控制符和转义字符外的其他字符原样输出,即在屏幕上显示:

```
This is a C program.
```

\n 的意思是回车换行。

(4) 注意: 每条语句用“;”号结束。

【例 1.2】 设计一个程序,计算两数之和并输出结果。

```
main()
{
    int a,b,sum;           /* 定义三个整型变量 a,b,sum*/
    a=123; b=456;          /* 给 a、b 赋初值 */
    sum=a+b;               /* 计算 a、b 之和赋给 sum */
    printf("sum=%d\n",sum); /* 输出 sum 的值 */
}
```

运行结果为:

```
sum=579
```

说明:

(1) 程序包含一个 main 函数作为程序执行的起点。{} 之间为 main 函数的函数体,main 函数的所有操作均在 main 函数体中。

(2) “/*”和“*/”括起来的部分是一段注释,注释只是为了改善程序的可读性,在编译、运行时不起作用(事实上编译时会跳过注释,目标代码中不会包含注释)。注释可以放在程序任何位置,并允许占用多行,需要注意“/*”、“*/”匹配,一般不要嵌套注释。

(3) “int a,b,sum;”是变量声明,声明了三个具有整数类型的变量 a, b, sum。C 语言的变量必须先声明再使用。

(4) “a=123;b=456;”是两条赋值语句。将整数 123 赋给整型变量 a, 将整数 456 赋给整型变量 b。a, b 两个变量的初始值分别为 123, 456。注意这是两条赋值语句,每条语句均用“;”结束。也可以将两条语句写成两行,即:

```
a=123;
b=456;
```

由此可见, C 语言程序的书写可以相当随意,但是为了保证容易阅读也要遵循一定的规范。

(5) “sum=a+b;”是将 a, b 两变量中存放的内容取出来做加法,然后将结果赋值给整型变量 sum。此时 sum 的内容为 579。

(6) “printf(“sum=%d\n”,sum);”是调用库函数输出 sum 的结果。%d 为格式控制,表示 sum 的值以十进制整数形式输出。

程序运行后,输出(显示): sum=579。

【例 1.3】 输入两个整数,计算两者中较大的数,并输出。

```
main()                               /* 主函数 */
{                                     /* main 函数体开始 */
    int a,b,c;                       /* 定义整型变量 a,b,c */
    int max(int x,int y);            /* 声明 max() 函数 */
    scanf("%d,%d",&a,&b);             /* 调用库函数 scanf() 给 a,b 赋值 */
    c=max(a,b);                     /* 调用 max, 将调用结果赋给 c */
    printf("max=%d",c);
```

```

}                                     /*main 函数体结束 */
int max(int x,int y)                 /*定义函数 max*/
{                                   /*max 函数体开始 */
    int z;                          /*定义变量 z */
    if(x>y)
        z=x;
    else
        z=y;
    return z;                       /*将 z 值返回,通过 max 带回调用处 */
}                                   /*max 函数体结束*/

```

说明:

(1) 本程序包括两个函数。其中主函数 main 仍然是整个程序执行的起点。函数 max 计算两数中较大的数。

(2) 主函数 main 调用 scanf 函数从键盘输入两个整数,赋给两个变量 a, b, 然后调用函数 max 获得两个数字中较大的值,并赋给变量 c。最后输出变量 c 的值(结果)。

(3) “int max(int x,int y);” 是函数 max 的声明,函数 max 的声明中可以看到此函数获得两个整数,返回一个整数。

(4) 函数 max 同样也用 {} 将函数体括起来。max 的函数体是函数 max 的具体实现。从参数表获得数据,处理后得到结果 z,然后将 z 返回调用函数 main。

(5) 本例还表明函数除了调用库函数外,还可以调用用户自己定义、编制的函数。

1.3.2 C 程序结构

综合上述三个例子,我们对 C 程序的基本组成和形式(程序结构)有了一个初步了解。

(1) C 程序由函数构成,C 语言是函数式的语言,函数是 C 程序的基本单位(如例 1.3)。

1) 一个 C 源程序至少包含一个 main 函数,也可以包含一个 main 函数和若干个其他函数。函数是 C 程序的基本单位。

2) 被调用的函数可以是系统提供的库函数,也可以是用户根据需要自己编写设计的函数。C 语言是函数式的语言,程序的全部工作都是由各个函数完成。编写 C 程序就是编写一个个函数。

3) C 函数库非常丰富,ANSI C 提供 100 多个库函数,Turbo C 提供 300 多个库函数。

(2) main 函数(主函数)是每个程序执行的起始点(如例 1.3)。一个 C 程序总是从 main 函数开始执行,而不论 main 函数在程序中的位置。

(3) 一个函数由函数首部 and 函数体两部分组成(如例 1.3 的 max 函数)。

1) 函数首部:一个函数的第一行。

返回值类型 函数名([函数参数类型 1 函数参数名 1][, ..., 函数参数类型 2, 函数参数名 2])
int max(int x,int y)

注意: 函数可以没有参数,但是后面的一对括号不能省略。

2) 函数体:函数首部下用一对 {} 括起来的部分。如果函数体内有多个 {}, 最外层是函数体的范围。函数体一般包括声明、执行两部分。

```

{
    [声明部分]: 定义本函数所使用的变量。
    [执行部分]: 由若干条语句组成命令序列(可以在其中调用其他函数)。
}

```

(4) C 程序书写格式自由。

1) 一行可以写几个语句，一个语句也可以写在多行上。

2) C 程序没有行号，也没有 FO、COBOL 那样严格规定书写格式（语句必须从某一列开始）。

3) 每条语句的最后必须有一个分号“;”表示语句的结束。

(5) 可以使用“/*”、“*/”对 C 程序中的任何部分作注释。注释可以提高程序可读性，使用注释是编程人员的良好习惯。

1) 编写好的程序往往需要修改、完善，事实上没有一个应用系统是不需要修改、完善的。很多人会发现自己编写的程序在经历了一段时间以后，由于缺乏必要的文档、必要的注释，最后连自己都很难再读懂。需要花费大量时间重新思考、理解原来的程序，这浪费了大量的时间。如果一开始编程就对程序进行注释，虽然刚开始麻烦一些，但日后维护可以节省大量的时间。

2) 一个实际的系统往往是多人合作开发，程序文档、注释是其中重要的交流工具。

(6) C 语言本身不提供输入/输出语句，输入/输出的操作是通过调用库函数（scanf, printf）完成。输入/输出操作涉及具体计算机硬件，把输入/输出操作放在函数中处理，可以简化 C 语言和 C 的编译系统，便于 C 语言在各种计算机上实现。不同的计算机系统需要对函数库中的函数作不同的处理，以便实现同样或类似的功能。

不同的计算机系统除了提供函数库中的标准函数外，还按照硬件的情况提供一些专门的函数。因此不同计算机系统提供的函数数量、功能会有一定差异。

1.4 源程序的编辑、编译、连接与运行

用高级语言编写的程序称为“源程序”，通常简称为程序。高级程序设计语言也必须被转换为机器语言程序才能被机器理解和执行，完成这种转换任务的系统软件称为编译程序。相应的转换过程通常称为编译。

高级语言是面向解题过程的，语言本身与具体的机器系统无关，因而用高级语言编写的应用程序可移植性好。编译程序是一种语言的具体实现，编译程序与具体机器系统有关，通常称为语言的一个版本。同一语言的不同版本不完全相同，在使用一种具体的高级语言及其编译程序开发软件时，必须参考与编译程序配套的有关资料。本书阐述的内容遵从 ANSI C 标准，对于大多数 C 编译程序具有通用性。为了使上机环境尽量简单，书中所有的例题均在 Visual C++ 6.0 上编译通过。

C 语言采用编译方式将源程序转换为二进制的目标代码。编写好一个 C 程序到完成运行一般经过以下几个步骤。

1. 编辑

所谓编辑，包括以下内容：①将源程序逐个字符输入到计算机内存；②修改源程序；③将修改好的源程序保存在磁盘文件中。编辑的对象是源程序，它是以 ASCII 代码的形式输入和存储的，不能被计算机执行。目前使用较多的编辑软件有：UNIX 下的编辑程序 ed、vi 等，MS-DOS 下的 EDLIN、Wordstar，Windows 下的 Write、Word 等字处理软件。关于编辑软件的使用方法请参阅《C 语言程序设计上机指导与习题集解答》一书或其他有关手册。

2. 编译

编译就是将已编辑好的源程序（已存储在磁盘文件中）翻译成二进制的目标代码。在编

译时,还要对源程序进行语法检查,如发现有错,则在屏幕上显示出错信息,此时应重新进入编辑状态,对源程序进行修改后再重新编译,直到通过编译为止。编译后得到的二进制代码在 UNIX 下是后缀为.o 的文件,在 MS-DOS 下是后缀为.obj 的文件。应当指出,经编译后得到的二进制代码还不能直接执行,因为第一个模块往往是单独编译的,必须把经过编译的各个模块的目标代码与系统提供的标准模块(如 C 语言中的标准函数库)连接后才能运行。

3. 连接

将各模块的二进制目标代码与系统标准模块经连接处理后,得到具有绝对地址的可执行文件,它是计算机能直接执行的文件。在 UNIX 下它以.out 为后缀(例如, f.out),在 MS-DOS 下以.exe 为后缀(例如, f.exe)。

4. 执行

执行一个经过编译和连接的可执行的目标文件。只有在操作系统的支持和管理下才能执行它。图 1-1 用以表示编辑、编译、连接、运行的过程。

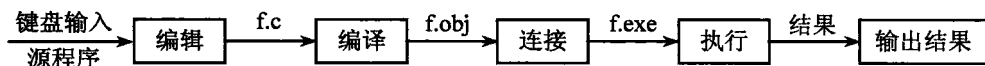


图 1-1 程序执行步骤

其中文件名为 f, 后缀按 MS-DOS 的规则表示。

近来“集成化”的工具环境已将编辑、编译、连接、调试工具集于一身(例如 Turbo C 和 Visual C++ 6.0 等),用户可以方便地在窗口状态下进行编辑、编译、连接、调试、运行的全过程。

习 题

一、单选题

1. 以下叙述中正确的是()。
 - A. C 语言的源程序不必经过编译就可以直接运行
 - B. C 语言的每条可执行语句最终都将被转换成二进制的机器指令
 - C. C 语言的源程序经编译形成的二进制目标文件可以直接运行
 - D. C 语言的函数可以单独进行编译
2. 对于一个正常运行的 C 程序,以下叙述中正确的是()。
 - A. 程序的执行总是从 main 函数开始,在 main 函数中结束
 - B. 程序的执行总是从第一个函数开始,在 main 函数中结束
 - C. 程序的执行总是从 main 函数开始,在最后一个函数中结束
 - D. 程序的执行总是从第一个函数开始,在最后一个函数中结束
3. 以下叙述中正确的是()。
 - A. C 程序的基本组成单位是语句
 - B. C 程序书写格式严格,要求一行内只能写一个语句
 - C. C 程序书写格式自由,一个语句可以写在很多行上
 - D. C 语言的语句必须在一行内写完