

普通高等院校计算机基础教育系列教材

程序设计基础— C 语言(第二版)

主 编 龙昭华 王玉柱
副主编 纪 钢 于显平
编 者 (以姓氏笔画为序)
丁明勇 王朝霞 张 仿
何 丽 杨芳明 高 论

重庆大学出版社

内 容 简 介

本书以模块和结构化为主线,由浅入深地介绍 C 语言程序设计的必要知识内容,并突出程序设计的模块化思想,适当淡化 C 语言的语法成分。全书共分 8 章,不同层次的读者依章节顺序阅读,能收到一定的学习效果,能较好地建立模块化程序设计思想,培养用软件解决实际问题的能力。

本书首先介绍了函数的概念,然后在后续章节的章节中贯穿和加深了函数的应用。为了突出 C 语言的实用性,第 7 章、第 8 章分别介绍了常用数据结构的编程实现和高级编程技术。

本书适用于理工类各层次学生的学习,也可作为软件开发人员的参考书。

图书在版编目(CIP)数据
程序设计基础:C 语言 /龙昭华, 王玉柱主编. —2 版.
重庆:重庆大学出版社,2006.2
(普通高等院校计算机基础教育系列教材)
ISBN 7-5624-2993-6
.程... . 龙... 王... . C 语言—程序设
计—高等学校—教材 . TP312
中国版本图书馆 CIP 数据核字(2006)第 003142 号

普通高等院校计算机基础教育系列教材
程序设计基础——C 语言

(第二版)

主 编 龙昭华 王玉柱
副主编 纪 钢 于显平
责任编辑:王海琼 吴庆佳 版式设计:范欣渝
责任校对:邹 忌 责任印制:秦 梅

*

重庆大学出版社出版发行
出版人:张鸽盛
社址:重庆市沙坪坝正街 174 号重庆大学(A 区)内
邮编:400030
电话:(023) 65102378 65105781
传真:(023) 65103686 65105565
网址: <http://www.cqup.com.cn>
邮箱: fxk@cqup.com.cn (市场营销部)
全国新华书店经销
重庆华林天美彩色报刊印务有限公司印刷

*

开本:787×1092 1/16 印张:21.5 字数:537 千
2004 年 2 月第 1 版 2006 年 2 月第 2 版 2006 年 2 月第 3 次印刷
印数:10 001—13 000
ISBN 7-5624-2993-6 定价:28.00 元

本书如有印刷、装订等质量问题,本社负责调换
版权所有,请勿擅自翻印和用本书
制作各类出版物及配套用书,违者必究

编委会

顾问	吴中福	邱玉辉	
主任	陈流汀		
副主任	杨天怡	严欣平	张鸽盛
委员	(以姓氏笔画为序)		
	王玉柱	甘玲	杨国才
	李建平	肖贵元	应宏
	邹显春	陈维	周建丽
	孟民果	洪汝渝	莫堃
	高占国	郭松涛	黄勤
	曾一		

序 言

计算机技术的飞速发展,加快了人类进入信息社会的步伐,改变了世界,改变了人们的工作、学习和生活,对社会发展产生了广泛而深远的影响。计算机技术在其他各学科中的应用,极大地促进了各学科的发展。不掌握计算机技术,就无法掌握最先进、最有效的研究开发手段,将影响到其所从事学科的发展。因此,计算机技术基础是 21 世纪高校非计算机专业大学生必须掌握的、最重要的基础之一。

1997 年教育部颁发“加强非计算机专业计算机基础教学工作的几点意见”教高[1997]155 号文件,明确了计算机基础教学在大学教育中的地位,提出了计算机基础教学三个层次的课程体系(即计算机文化基础、计算机技术基础和计算机应用基础),并提出了课程建设与改革思路,对促进和规范高校非计算机专业计算机基础教学、高校非计算机专业计算机知识和能力培养起到了重要作用。

进入 21 世纪,针对信息化社会中计算机应用领域不断扩大和高校学生计算机知识的起点不断提高等特点,教育部高校计算机课程教学指导委员会对高校非计算机专业计算机基础教学的目标、课程设置和主要课程教学内容进行了新的规划,将 1997 年提出的三层次教学调整为四个领域、三个层次和六个核心课程,即“大学计算机

基础”、“计算机程序设计基础”、“计算机硬件技术基础”、“数据库技术与应用”、“多媒体技术与应用”、“网络技术与应用”。

为了适应新的要求,我们组织一批长期从事计算机技术教学和科研的教师,编写了这套计算机基础教育系列教材。本系列教材有如下特点:

1. 适合于计算机技术的发展和应用程序领域的扩大,以及高校学生计算机知识起点的提高。内容主要涉及“计算机系统与平台”、“计算机程序设计基础”、“数据分析与信息处理”和应用系统开发领域,使学生掌握计算机应用基本知识和技能,为今后的学习和工作打下坚实基础。

2. 强调应用和实用。非计算机专业的计算机基础教学以应用为目的,因此,本系列教材在编写上特别注意应用需要,强调实用性。主要课程教材都配有实验教程,基本知识理论讲深讲透,使用技术主要通过学生上机实验来掌握。

3. 便于自学。为了充分调动学生的学习主动性和能动性,本系列教材在写法上,既注意概念的严谨与清晰,又特别注意用易读、易懂的方法阐述问题,应用举例丰富,便于自学。

总而言之,本系列教材的编写指导思想是:内容要新,要体现计算机技术的新发展和适应教学改革的要求;

概念要清晰、通俗易懂,便于学生自学;应用性、实用性要强,切实在培养学生应用能力上下功夫;层次配套,可选择性强,适用面宽,既是普通高校非计算机专业本专科学学生教材,亦可作为高等教育自学教材和工程技术人员参考书。

限于编者水平,系列教材的内容及体系难免有缺点错误,诚恳希望读者和专家给予指正。

编委会
2005年8月

前言

C 语言作为国际上广泛流行的通用程序设计语言,具有简洁、表达能力强、可移植性好、产生的目标代码效率高等优点,在计算机研究和应用中已展现出其强大的生命力。近年来,高等院校非计算机专业也陆续开设了这门程序设计基础课程。

但对于程序设计的初学者来说,C 语言规则较多,使用较灵活,不易掌握,是一门被公认为很难理解、学习的计算机语言。另外,随着 C 语言的应用范围越来越广,所涉及的知识也在不断地增加,学习起来有一定的困难。

为了更好地满足课堂教学和读者自学的要求,培养读者的实际应用能力,达到学以致用目的,编者在本书的编写过程中,打破按 C 语言的语法成分组织教学内容的常规思路,以 C 语言程序的模块与结构为主线,由浅入深地介绍 C 语言程序设计的必要内容,并突出程序设计的模块化思想,适当淡化语言的语法成分。这样,使读者在学习用 C 语言解决实际问题的过程中,逐步理解 C 语言语法成分,增加学习目的性。在有限的学时下开展教学,长期以来,一直是教学难点。编者认为:有效地进行 C 语言“低级语言”

与“高级语言”的融合,是着重解决的关键问题之一,尽力达到“容易读懂、实际应用”的效果。

全书共 8 章,第 1 章介绍了 C 语言的基础知识、算法的基本概念和软件工程的基础思想;第 2 章从介绍 C 语言中的模块—函数入手,阐述函数的必要知识,包括基本数据类型、表达式、函数的相互调用及变量的特性等;第 3 章介绍了程序的 3 种基本结构及语句、各种不同类型的数组等;第 4 章介绍了指针的概念、指针与数组、指针与函数关系等,详细地介绍了指针的特点和灵活性;第 5 章介绍了结构体、共用体、枚举类型、位段等数据类型的定义和使用;第 6 章介绍了文件的操作方法和相关的系统函数;第 7 章介绍了线性表、链表、栈和队列、二叉树等基本数据结构的概念和常用算法;第 8 章介绍了 C 语言的高级编程技术,如递归程序设计、图形的制作、音乐的演奏方法、Windows、Linux 操作系统上编程等。通过一些饶有趣味的实用案例,使读者在轻松、愉快的气氛中理解和探索用 C 语言进行程序设计的奥妙。全书各章均配备有适量习题。为便于学生参加计算机等级考试,本书附录

提供了考试大纲、考试模拟题及参考答案。

本教材适用于理工科类各层次学生的学习,各院校教师在授课中可适当进行内容取舍。本书也可作为软件开发人员的参考书。

本书第 1 章由张仿、龙昭华编写;第 2 章由龙昭华、张仿、于显平编写;第 3 章由高论编写;第 4 章由纪钢编写;第 5 章由王朝霞编写;第 6 章由丁明勇编写;第 7 章由王玉柱编写;第 8 章由杨芳明编写;附录由于显平编写;何丽负责全书阅校。

本书编者都是长期从事大学计算机教学工作的教师(重庆工商大学、西

南大学、重庆邮电大学、解放军后勤工程学院、重庆交通学院、重庆工学院、重庆科技学院、重庆师范大学),由于教学和科研任务繁重,本书编写时间仓促,疏漏和错误之处在所难免,敬请谅解,并诚恳期待读者批评指正。

本书配有电子教案和课件,可在重庆大学出版社的资源网站(www.cqup.com.cn,用户名和密码:cqup)下载。

编 者

2005 年 12 月

目 录

1	C 语言程序设计概述	1
1.1	计算机语言发展简史	1
1.2	计算机中数据的表示	3
1.2.1	原码、反码、补码	3
1.2.2	ASCII 码	3
1.3	C 语言的特点	4
1.3.1	C 语言简介	4
1.3.2	C 语言特点	5
1.4	C 语言模块	6
1.4.1	计算机程序	6
1.4.2	模块与 C 语言结构	6
1.5	算法	11
1.5.1	算法的概念	11
1.5.2	结构化表示算法	11
1.6	软件工程概念	14
1.6.1	软件危机	14
1.6.2	软件工程	14
1.6.3	瀑布模型	15
小结 1		16
习题 1		16
2	函数	17
2.1	函数的概念	17
2.1.1	函数引入	17
2.1.2	函数分类	18
2.2	函数定义	18
2.3	基本数据类型	20
2.3.1	常量概念、3 种基本数据类型	20
2.3.2	变量概念、关键字、标识符	22
2.4	使用 printf()、scanf() 函数	23
2.5	运算符及表达式	27
2.5.1	运算符概念	27

2.5.2	表达式概念	28
2.5.3	算术运算符和算术表达式	28
2.5.4	自动类型转换和强制类型转换	29
2.5.5	前缀、后缀自增(减)运算符	30
2.5.6	赋值运算符与赋值表达式	32
2.5.7	逗号运算符和逗号表达式	33
2.5.8	关系运算符和关系表达式	33
2.5.9	逻辑运算符和逻辑表达式	34
2.5.10	取地址运算符和间接访问运算符	35
2.5.11	类型长度运算符	36
2.5.12	位运算符	36
2.5.13	条件运算符	38
2.5.14	C 语言语句分类	38
2.6	计算机存储器地址概念	40
2.6.1	地址概念	40
2.6.2	静态存储区概念	40
2.7	自定义函数设计举例	41
2.8	局部变量、全局变量、静态变量及其值传递	43
2.8.1	从不同角度对函数分类	43
2.8.2	局部变量	44
2.8.3	主调函数与被调函数之间的值传递	45
2.8.4	全局变量	47
2.8.5	静态变量	49
2.9	编译预处理	50
2.9.1	宏定义	50
2.9.2	文件包含	58
2.9.3	条件编译	59
小结 2		61
习题 2		61

3 程序控制结构语句与数组..... 65

3.1	顺序结构程序设计	65
3.2	选择结构程序设计	66
3.2.1	if 语句	66
3.2.2	switch 语句	73
3.2.3	选择结构程序设计举例	75
3.3	循环结构程序设计	79
3.3.1	4 种循环结构	79

3.3.2	break 与 continue 语句	89
3.3.3	循环结构程序设计举例	90
3.4	数组	92
3.4.1	一维数组	92
3.4.2	二维数组	96
3.4.3	字符数组	100
小结 3		104
习题 3		105

4 指针

4.1	指针与指针变量	111
4.1.1	指针的概念	111
4.1.2	指针变量	111
4.2	指针变量与数组	117
4.2.1	指针变量与数组	117
4.2.2	指针变量在一维数组中的应用	119
4.2.3	指针变量在多维数组中的应用	121
4.2.4	指针变量在字符数组中的应用	127
4.3	指针数组与多级指针	130
4.3.1	指针数组的定义	130
4.3.2	指针数组的使用	130
4.3.3	多级指针	132
4.4	指针变量与函数	134
4.4.1	函数的操作方式与指针变量	134
4.4.2	指针型函数的定义与使用	134
4.4.3	函数指针的定义与使用	135
4.4.4	与指针有关的函数参数传递方式	137
4.4.5	带参数的 main 函数和命令行参数	143
小结 4		144
习题 4		146

5 结构体、共用体、枚举类型

5.1	结构体类型变量的定义与引用	153
5.1.1	结构体类型的定义	153
5.1.2	结构体变量的定义	154
5.1.3	结构体变量的引用	157
5.1.4	结构体变量的初始化	158

5.1.5	位段	159
5.2	结构体数组	161
5.2.1	结构体数组的定义	161
5.2.2	结构体数组的初始化	163
5.2.3	结构体数组的引用	163
5.2.4	结构体数组应用举例	164
5.3	指针与结构体	167
5.3.1	指向结构体变量的指针	167
5.3.2	指向结构体数组的指针	170
5.3.3	用结构体变量和指向结构体的指针作函数参数	171
5.4	动态存储管理	174
5.4.1	需要动态存储管理的原因	174
5.4.2	C 语言的动态存储管理机制	175
5.5	结构体应用举例	178
5.6	共用体	181
5.6.1	共用体及共用体变量的定义	181
5.6.2	共用体变量的引用方式	184
5.6.3	共用体类型数据的特点	184
5.7	枚举类型	186
5.7.1	枚举类型的定义及变量的定义	187
5.7.2	枚举类型应用举例	189
5.8	复合数据类型及 typedef	190
小结 5		191
习题 5		192

6 文件

6.1	文件概述	198
6.1.1	文件的概念	198
6.1.2	文件系统	198
6.1.3	文件分类	199
6.2	文件类型指针	200
6.3	文件相关预定义函数	201
6.3.1	文件的打开与关闭	201
6.3.2	文件的读/写	202
6.3.3	文件的定位	211
6.4	应用举例	213
小结 6		215
习题 6		216

7	常用数据结构及其程序设计	219
7.1	线性表	219
7.1.1	线性表的定义和运算	219
7.1.2	顺序表	220
7.1.3	链表	222
7.2	栈和队列	227
7.2.1	栈的定义	227
7.2.2	栈的基本操作	228
7.2.3	队列的定义	229
7.2.4	顺序队列的基本操作	230
7.3	二叉树	231
7.3.1	树的定义	231
7.3.2	二叉树的定义	232
7.3.3	二叉树的链式存储	233
7.3.4	二叉树的遍历	235
小结 7		236
习题 7		237
8	C 语言高级编程技术	238
8.1	递归程序设计	238
8.1.1	递归与递归程序设计	238
8.1.2	递归程序执行过程分析	239
8.1.3	递归算法的优缺点	241
8.1.4	递归程序设计的应用实例	241
8.2	文本的屏幕输出和键盘输入	245
8.2.1	文本的屏幕输出	245
8.2.2	键盘输入	251
8.3	图形程序设计	253
8.3.1	图形模式的初始化	253
8.3.2	图形模式下的坐标系	254
8.3.3	屏幕图形的色彩与相关操作	254
8.3.4	基本绘图函数	256
8.3.5	图形设计举例	259
8.4	发声技术	263
8.4.1	声音函数	263
8.4.2	计算机乐谱	264

8.5	Linux 操作系统 C 语言编程基础	268
8.5.1	Linux 操作系统基础	268
8.5.2	GCC 的使用	268
8.5.3	利用 GCC 开发 C 语言程序	271
8.5.4	调试技巧	274
8.6	Windows 操作系统 C 语言编程基础	276
8.6.1	Windows 编程的特点	276
8.6.2	重要的 Windows 术语	278
8.6.3	编写基本的 Windows 应用程序	279
8.7	综合应用开发实例	286
8.7.1	问题定义	286
8.7.2	概要设计	286
8.7.3	详细设计	287
8.7.4	编码	288
小结 8		299
习题 8		300

附录		301
----	-------	-----

附录 1	ASCII 码表	301
附录 2	运算符表	302
附录 3	常用函数表	303
附录 4	常用术语中英文对照表	313
附录 5	计算机等级考试大纲及样题	315

参考文献		328
------	-------	-----

1 C 语言程序设计概述

在学习 C 语言程序设计之前,有必要知道“什么是计算机语言”、“什么是程序”、“一个 C 语言程序究竟像什么样子”、“如何运行一个简单的 C 语言程序”。本章首先讨论这方面的问题,以帮助读者建立对这些概念的感性认识,为后面的学习打下基础。

1.1 计算机语言发展简史

计算机工作的基本原理为存储程序,程序是它的核心,而编写程序的工具是计算机语言,因此了解程序设计语言具有至关重要的作用。计算机的历史有多久,计算机语言的发展就有多久。总的来说,计算机语言发展的规律为:其描述问题的方法越来越贴近人类的思维习惯,功能也越来越完善,且表达形式却越来越简单。

1) 机器语言

第 1 代语言又称“机器语言”,是计算机诞生和发展初期使用的语言,表现为二进制编码形式,是由 CPU 可以识别的 0、1 序列构成的指令码。用机器语言编写程序极为复杂,要求使用者熟悉计算机的所有细节,程序的质量很大程度上取决于个人的硬件知识水平。随着计算机的硬件结构复杂化,指令系统变得非常庞大,一般的工程技术人员难以掌握。

2) 汇编语言

第 2 代语言是“汇编语言”,它开始于 20 世纪 50 年代初,用助记符来表示每一条机器指令,比机器语言进了一步,但其大部分指令还是和机器语言一一对应的,用汇编语言编写的程序要翻译成机器语言后才能执行。

3) 高级语言

第 3 代语言即“高级语言”,它起源于 20 世纪 50 年代中期,与人们的自然语言和数学语言更接近,可读性强,编程方便。用一种高级语言编辑的源程序,可以在具有与该种语言编译系统的不同计算机上使用,但必须编译或解释成机器语言才能执行。Basic、Fortran、Cobol、Pascal、C 等都属于第 3 代语言。

前面 3 代语言都强调“过程”,所以可称为面向过程的语言。

第 4 代语言称为“非过程化语言”。顾名思义,用户使用这种语言,不必关心问题的解法和处理过程的细节描述,只要说明所要完成的加工和条件,指明输入数据和输出形式,就能得到需要的结果,其他工作由系统来完成。第 3 代语言告诉计算机怎么做,第 4 代语言告诉计算机做什么。第 4 代语言又称为“面向对象的语言”。

第 5 代语言,堪称智能性语言,除具有第 4 代语言的许多特点外,还具备许多新的功能,

尤其是具有一定的智能处理功能。Prolog 语言是第 5 代语言的代表, 广泛运用于抽象问题求解、数据逻辑、公式处理、自然语言理解、专家系统和人工智能等许多领域。

在 20 世纪 60 年代这 10 年里, 出现了上百种高级语言。20 世纪 70 年代, 随着结构化程序设计思想的日益深入, 这段时间里问世的几种程序设计语言(如 Pascal、Ada、C 等) 的控制结构大大简化。20 世纪 80 年代提出的“面向对象”的概念是相对于“面向过程”的一次革命。C++ 便是面向对象的程序设计语言, 面向对象技术在系统程序设计、数据库及多媒体运用等诸多领域得到广泛运用。面向对象程序设计语言与人们的自然语言比较接近, 大大提高了程序设计的效率, 便于人们进行交流。

4) 解释程序和编译程序

计算机的 CPU 只能够执行机器语言, 高级语言不能在 CPU 中直接运行。这意味着: 计算机系统必须有一个程序将高级语言“书写”的语句序列“翻译”成为机器语言。担负这一任务的程序称为“语言处理程序”, 它可以分为两大类: 解释程序和编译程序。

(1) 解释程序

解释程序是把编译和运行过程合为一体。这种方式并不形成目标程序, 而是按目标程序的动态顺序逐句解释并立即执行程序。解释程序输入的是源程序和原始数据, 输出为目标程序的计算结果而非目标程序。

解释程序的实际工作原理很复杂, 通常先对源程序进行词法、语法检查, 源程序换码变成便于处理的内部形式, 建立一些表格, 如符号表、标号表等。进行错误检查后, 下一阶段的任务是用前一阶段提供的表格, 对内部形式的源程序解释并执行。

(2) 编译程序

用高级语言编写的源程序可通过编译器翻译为语意上等价的目标程序, 由机器语言、汇编语言和某些中间语言表示。如目标程序是用汇编语言编成的, 要先用汇编语言程序编译成机器语言程序再执行; 如果是用某些中间语言编程, 则应先由解释器解释执行或再经编译后执行。

高级语言程序要运行需要执行 3 个步骤: 编译阶段、连接装配阶段、运行阶段。编译阶段由编译程序扫描源程序并翻译为目标程序。编译程序输出是待装配的目标程序模块, 需要先经过连接装配, 把目标程序及必须的运行子程序连接形成可执行的机器代码。

在编译和解释这两种方式中, 解释器比编译器灵活, 解释器执行源程序时有对机器的控制权, 能在发现错误时停止执行源程序, 并且会指示错误的行; 而编译程序发现目标程序错误时, 程序只能简单退出或打印一些运行结果产生的错误信息。调试时, 解释器可以按要求执行。目前一些高级程序开发环境的编译器有解释器的特性, 可编译一些程序, 使其在环境的控制下执行。如果有错误, 系统会给出提示, 并接管控制器来改正错误, 然后选择重新执行还是再次编译后执行。虽然解释器比编译器灵活, 但是编译器的速度快于解释器, 两者各有优点, 因此在实际应用中要根据需求灵活掌握。

1.2 计算机中数据的表示

1.2.1 原码、反码、补码

计算机中所有的信息都是用二进制表示的,而数据的类型很多,表达的信息也是千变万化,怎样通过 0 和 1 这二个数字来实现呢?计算机有它自己的一套模式。如对于正整数(大于等于零,在 C 语言中称为无符号整型)数据,表示较为简单,这里就不再赘述。对于有符号的数据,计算机中采用固定一位为符号位的方法,即首位用来表示符号(“1”代表负、“0”代表正),其他位则表示数据。计算机中表示的带符号的数称为机器数,有原码、反码、补码 3 种表示方法。

(1) 原码

在原码中规定正数符号位为 0,负数符号位为 1,后面的数据位则表示该数的绝对值。例如:

[+15] 原 = 0000000000001111B

[-15] 原 = 1000000000001111B

利用原码运算时,要首先判定符号,在运算中较为复杂。为了解决这个困难,出现了反码和补码。

(2) 反码

规定正数的反码与原码相同;负数的反码由该负数的原码进行计算:原码符号位不变,数据位按位取反(即 0 变为 1,1 变为 0)。例如:

[+15] 反 = 0000000000001111B

[-15] 反 = 1111111111110000B

(3) 补码

正数的补码等于原码;负数的补码等于该负数的反码加 1,即按位取反后加 1。例如:

[+15] 补 = [+15] 原 = 0000000000001111B

[-15] 补 = 1111111111110001B

1.2.2 ASCII 码

计算机需要处理的数据不仅包括数值数据,还有大量用于表达文字信息的符号数据。而在计算机内部,所有信息都是用“0”和“1”两个符号来表示的,包括汉字字符信息也一样。汉字信息编码方式的思想来自于西文信息编码方式,在程序设计中一般都是采用西文编码方式,此处我们着重学习西文编码方式中 ASCII 码的编码方式。

ASCII 码是 ANSI(美国国家标准协会)制定的“美国标准信息交换代码”的简称,是目前国际上最流行的字符信息编码方案,早已经被国际标准化组织(ISO)采纳。它包括 0~9 十个阿拉伯数字、大小写英文字母、控制符(如回车、换行等)、专用符号以及几十种可打印字符等。

在计算机中, 用一个正整型数据来代表一个字符。全部字符与其对应的正整型数据排列在表格中。由于一个字节有 8 个二进制位, 当只用低 7 位存放无符号整型数据时, 可存放 0 ~ 127 中的任一值; 当 8 个位全部用于存放无符号整型数据时, 可存放 0 ~ 255 中的任一值。前者称为基本 ASCII 码表, 有 128 个字符; 后者称为扩展 ASCII 码表, 有 256 个字符。习惯上, 将代表这些字符的正整数值称为 ASCII 码值。

例如: 字母 A ~ Z 的 ASCII 码值为 65 ~ 90, 其小写形式的 ASCII 码值为 97 ~ 122。在 C 语言中, 有时程序员使用十六进制、八进制、二进制表示 ASCII 码值, 是为了换算方便、理解直观的原因。为了方便查阅, 本书附录列出了 ASCII 表。

在计算机里, 存储和传送的单位通常使用 Byte(1 Byte = 8 bits), 所以 7 位的 ASCII 码, 也用一个字节来表示。最高 1 位没有使用, 通常填 0; 也可以把它用作校验位或者用来扩展字符集。

1.3 C 语言的特点

1.3.1 C 语言简介

1978 年美国电话电报公司(AT&T) 贝尔实验室正式发表了 C 语言。同时 B. W. Kernighan 和 D. M. Ritchie 合著了著名的“ The C Programming Language ”一书, 通常简称为《K&R》。

C 语言的初衷是用于编写操作系统和系统程序, 初期用在 PDP-11 计算机上写 UNIX 操作系统。因为系统软件要与底层硬件打交道, C 语言出现前系统软件都是利用汇编语言来编写的, 但汇编语言的开发效率低, 程序可读性和可移植性都很差, 而一般的高级语言离硬件太远, 实现起来无能为力, 所以需要一种既能兼顾低级语言特点又具有高级语言特点的计算机语言, 这就是 C 语言出现的主要原因。

20 世纪 70 年代后 C 语言成为 UNIX 的标准开发语言, 并随着 UNIX 系统的流行而得到越来越广泛的接受和应用。20 世纪 80 年代后 C 语言被搬到包括大型机、工作站等的许多系统上, 逐渐成为开发系统程序和复杂软件的一种通用语言。随着微机的蓬勃发展、处理能力的提高和应用的日益广泛, 越来越多的人参与微机应用系统的开发工作, 需要适合开发系统软件和应用软件的语言。C 语言能较好满足人们的需要, 在微机软件开发中也得到日益广泛的应用, 逐渐成为最常用的系统开发语言之一, 被用于开发微型机上的各种程序, 直至非常复杂的软件系统。

在设计 C 语言时, 设计者主要把它作为汇编语言的替代品, 作为自己写操作系统的工具, 因此更多强调的是灵活性和方便性, 其规定很不严格, 可以用许多不“ 规矩 ”的方式写程序, 因此也留下了许多不安全因素。这就要求编程者自己注意可能的问题, 程序的正确性主要靠人来保证, 语言的处理系统(编译程序) 不能提供多少帮助。随着应用范围的扩大, 使用 C 语言的人越来越多(显然其中大部分人对语言的理解远不如设计者), C 语言的缺点日益突出, 用 C 语言开发的复杂程序里常带有隐藏很深的错误, 难以发现和改正。

人们更强烈地希望 C 语言能成为一种更安全可靠、不依赖于具体计算机和操作系统(如 UNIX) 的标准程序设计语言。美国国家标准局(ANSI) 在 20 世纪 80 年代建立了专门小组研