



普通高等教育“十二五”重点规划教材 公共课系列
中国科学院教材建设专家委员会“十二五”规划教材

C语言程序设计

张淑华 朱丽莉◎主编



科学出版社

内 容 简 介

本书以教育部高等学校计算机科学与技术教学指导委员会编制的《关于进一步加强高等学校计算机基础教学的意见暨计算机基础课程教学基本要求》中有关计算机程序设计基础(C语言)课程教学基本要求为指导,结合教育部考试中心最新的全国计算机等级考试二级(C语言程序设计)考试大纲要求和作者多年的教学实践经验编写而成。

全书共11章,主要包括计算机语言与结构化程序设计的基本概念、C语言的基本概念、三种基本结构程序设计、数组、结构体、共用体、枚举类型、函数、变量的存储属性、编译预处理、指针、用户自定义类型以及文件等内容。

本书体系完整、语言精炼,叙述深入浅出,内容详略得当、注重实践。针对全国计算机二级考试,对例题与习题都作了精心设计。

本书可作为高等学校本、专科计算机程序设计课程教学用书,也可作为全国计算机二级考试辅导用书和学生自学参考书。

图书在版编目(CIP)数据

C语言程序设计/张淑华,朱丽莉主编. —北京:科学出版社,2012

(普通高等教育“十二五”重点规划教材 公共课系列·中国科学院教材建设专家委员会“十二五”规划教材)

ISBN 978-7-03-033012-3

I. ①C… II. ①张… ②朱… 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(2011)第258535号

责任编辑:戴薇 李瑜 张振华 / 责任校对:耿耘
责任印制:吕春珉 / 封面设计:东方人华平面设计部

科学出版社出版

北京东黄城根北街16号

邮政编码 100717

<http://www.sciencep.com>

骏立印刷厂印刷

科学出版社发行 各地新华书店经销

*

2012年2月第 一 版 开本:787×1092 1/16

2012年3月第二次印刷 印张:22 3/4

字数:522 000

定价:38.00元

(如有印装质量问题,我社负责调换<骏杰>)

销售部电话 010-62140850 编辑部电话 010-62135763-2038

版权所有,侵权必究

举报电话:010-64030229; 010-64034315; 13501151303

前 言

教育部高等学校计算机科学与技术教学指导委员会于 2006 年编制了《关于进一步加强高等学校计算机基础教学的意见暨计算机基础课程教学基本要求》，提出计算机基础教学是面向非计算机专业的计算机教学，所以它不同于计算机专业的计算机教学。计算机基础教学的目标是培养学生掌握一定的计算机基础知识、技术与方法，以及利用计算机解决本专业领域中问题的能力。计算机程序设计基础是大学计算机基础教学系列中的核心课程，主要讲授程序设计语言的基本知识和程序设计的技术与方法。

由于 C 语言具有功能丰富、表达能力强、使用灵活方便、应用面广、目标程序效率高、可移植性好等特点，自 20 世纪 90 年代以来，C 语言迅速在全世界得到普及和推广。现在，大多数高校都把 C 语言作为第一门计算机语言进行计算机程序设计基础教学，学习和掌握 C 语言已经成为许多学生的迫切需要。

本书编者都是长期从事计算机基础教学的高校一线教师，按照计算机基础教学规律精心编排各章节内容，将多年的教学经验和体会融入书中。在各章内容的选择上兼顾教育部考试中心最新的全国计算机等级考试二级（C 语言程序设计）考试大纲要求，精选并设计例题和习题。全书共 11 章，主要包括计算机语言与结构化程序设计的基本概念、C 语言的基本概念、三种基本结构程序设计、数组、结构体、共用体、枚举类型、函数、变量的存储属性、编译预处理、指针、用户自定义类型以及文件等内容。本书体系完整、语言精炼，叙述深入浅出，内容详略得当、注重实践，力求使学生掌握结构化程序设计的方法和技能，具有用 C 语言编程并解决实际问题的能力。书中所有的例题和习题，都是在 Visual C++ 6.0 环境下调试通过的。

本书可作为高等学校本、专科计算机程序设计课程教学用书，也可作为全国计算机二级考试辅导用书和学生自学的参考书。C 语言程序设计是一门实践性很强的课程，读者可以参考、使用与本书配套的《C 语言程序设计实践教程》（韩志明、王立君主编，科学出版社）。

本书由张淑华、朱丽莉担任主编，由于雪晶、顾煜新担任副主编。具体编写分工：于雪晶编写第 1~4 章，顾煜新编写第 5~7 章，朱丽莉编写第 8 章，张淑华编写第 9~11 章和附录。全书由朱丽莉教授审稿。在本书的编写和出版过程中，科学出版社给予了大力帮助和支持，在此表示感谢。

由于编者水平有限，书中难免有疏漏之处，恳请读者批评指正。

编 者

2011 年 10 月

目 录

第 1 章 C 语言概述	1
1.1 计算机语言与结构化程序设计	1
1.1.1 计算机语言	1
1.1.2 结构化程序设计	2
1.2 C 语言的发展历史和特点	5
1.2.1 C 语言的发展历史	5
1.2.2 C 语言的特点	6
1.3 C 语言程序的基本结构和执行过程	7
1.3.1 C 语言程序简单实例	7
1.3.2 C 语言程序执行过程	12
1.4 C 语言的字符集和标识符	13
1.4.1 字符集	13
1.4.2 标识符	13
1.4.3 预定义标识符	15
习题 1	15
第 2 章 数据类型及其表达式	17
2.1 C 语言数据类型简介	17
2.2 基本数据类型	18
2.2.1 常量与变量	18
2.2.2 整型数据	18
2.2.3 实型数据	23
2.2.4 字符型数据	26
2.2.5 空类型	30
2.3 运算符与表达式	31
2.3.1 C 运算符简介	31
2.3.2 算术运算符和算术表达式	32
2.3.3 关系运算符和关系表达式	35
2.3.4 逻辑运算符和逻辑表达式	36
2.3.5 条件运算符和条件表达式	38
2.3.6 赋值运算符和赋值表达式	39
2.3.7 逗号运算符和逗号表达式	41
2.3.8 位运算符和位运算表达式	43
2.4 不同类型数据间的转换	46
2.4.1 不同类型数据间的隐式转换	46

2.4.2 不同类型数据间的显式转换	47
习题 2	48
第 3 章 顺序结构的程序设计	52
3.1 C 语句概述	52
3.2 数据输入/输出的概念及在 C 语言中的实现	57
3.3 字符数据的输入和输出	57
3.3.1 字符输入函数 <code>getchar()</code>	58
3.3.2 字符输出函数 <code>putchar()</code>	59
3.4 格式化输入与输出	60
3.4.1 格式化输入函数 <code>scanf()</code>	60
3.4.2 格式化输出函数 <code>printf()</code>	64
习题 3	68
第 4 章 选择结构的程序设计	73
4.1 条件选择结构	73
4.1.1 <code>if</code> 语句的两种形式	73
4.1.2 <code>if</code> 语句的嵌套	77
4.2 开关选择结构	83
4.2.1 <code>switch</code> 语句形式	83
4.2.2 <code>switch</code> 语句的嵌套	88
4.3 选择结构程序设计举例	89
习题 4	91
第 5 章 循环结构的程序设计	96
5.1 穷举与迭代算法	96
5.2 <code>goto</code> 语句	98
5.3 <code>while</code> 语句	99
5.4 <code>do-while</code> 语句	102
5.5 <code>for</code> 语句	104
5.6 循环的嵌套	108
5.7 循环结构中的跳转语句	111
5.7.1 <code>break</code> 语句	111
5.7.2 <code>continue</code> 语句	113
5.8 循环结构程序设计举例	115
习题 5	118
第 6 章 数组	124
6.1 一维数组	125
6.1.1 一维数组的定义	125

6.1.2	一维数组元素的引用	126
6.1.3	一维数组的初始化	128
6.1.4	一维数组应用举例	129
6.2	二维数组	136
6.2.1	二维数组的定义	136
6.2.2	二维数组元素的引用	138
6.2.3	二维数组的初始化	139
6.2.4	二维数组应用举例	141
6.3	字符数组	145
6.3.1	字符数组的定义	145
6.3.2	字符数组的初始化	146
6.3.3	字符数组的引用	148
6.3.4	字符串处理函数	155
6.3.5	字符数组应用举例	159
	习题 6	161
第 7 章	结构体、共用体与枚举	167
7.1	结构体	167
7.1.1	结构体类型的定义	167
7.1.2	结构体变量	169
7.1.3	结构体数组	172
7.2	共用体	175
7.2.1	共用体类型的定义	175
7.2.2	共用体变量	176
7.3	枚举类型	178
7.3.1	枚举类型的定义	178
7.3.2	枚举类型变量	179
7.4	类型的重新命名	180
	习题 7	181
第 8 章	函数	186
8.1	概述	186
8.2	函数的定义	187
8.2.1	无参函数的定义	187
8.2.2	有参函数的定义	188
8.2.3	空函数的定义	189
8.3	函数的参数和函数的返回值	190
8.3.1	形式参数和实际参数	190
8.3.2	函数的返回值	191

8.4 函数的调用	193
8.4.1 函数调用的一般形式	194
8.4.2 函数调用的方式	194
8.4.3 函数的声明	195
8.5 函数的嵌套调用	198
8.6 函数的递归调用	200
8.7 构造类型数据作为函数的参数	205
8.7.1 结构体变量作为函数的参数	205
8.7.2 数组作为函数的参数	207
习题 8	212
第 9 章 变量、函数的属性和编译预处理	217
9.1 变量的存储类型	217
9.1.1 变量的存储类型含义	217
9.1.2 变量的存储类型分类	217
9.2 变量的作用域	219
9.2.1 局部变量及其作用域	219
9.2.2 全局变量及其作用域	223
9.3 内部函数和外部函数	228
9.3.1 内部函数	228
9.3.2 外部函数	228
9.4 编译预处理	230
9.4.1 宏定义	230
9.4.2 文件包含	236
9.4.3 条件编译	240
习题 9	242
第 10 章 指针	245
10.1 指针与指针变量	245
10.1.1 指针的概念	245
10.1.2 指针变量	247
10.1.3 指针运算	252
10.2 指针变量与数组	255
10.2.1 一维数组的指针和指向一维数组的指针变量	255
10.2.2 二维数组的指针和指向二维数组的指针变量	261
10.2.3 字符串的指针和指向字符串的指针变量	268
10.3 指针变量与结构体	273
10.3.1 结构体变量的指针和指向结构体变量的指针变量	273
10.3.2 结构体数组的指针和指向结构体数组的指针变量	276

10.4 指针变量与函数	278
10.4.1 函数的指针和指向函数的指针变量	278
10.4.2 指针变量作函数参数	280
10.4.3 返回指针值的函数	288
10.5 指针数组	290
10.5.1 指针数组的定义	290
10.5.2 指针数组元素的引用	290
10.5.3 指针数组的初始化	292
10.5.4 指针数组应用举例	293
10.5.5 指针数组作 main() 函数的形参	294
10.6 指向指针的指针变量	297
10.6.1 指向指针的指针变量的定义	298
10.6.2 指向指针的指针变量的引用	298
10.6.3 指向指针的指针变量的初始化	298
10.6.4 指向指针的指针变量的应用举例	299
10.7 用指针处理链表	300
10.7.1 链表概述	301
10.7.2 单链表	301
10.7.3 动态内存分配函数	303
10.7.4 建立链表	306
10.7.5 输出链表	308
10.7.6 对链表的删除操作	309
10.7.7 对链表的插入操作	311
习题 10	314
第 11 章 文件	319
11.1 C 文件概述	319
11.2 文件类型指针	321
11.3 文件的打开与关闭	322
11.3.1 文件的打开 (fopen() 函数)	322
11.3.2 文件的关闭 (fclose() 函数)	324
11.4 文件的读写	325
11.4.1 字符读写函数	325
11.4.2 字符串读写函数	328
11.4.3 格式化读写函数	330
11.4.4 数据块读写函数	332
11.5 文件的定位	335
11.5.1 rewind() 函数	335
11.5.2 文件的随机读写	336

11.5.3 ftell()函数	339
11.6 出错的检测	340
11.6.1 ferror()函数	340
11.6.2 clearerr()函数	340
习题 11	340
附录 A 常用字符与 ASCII 代码对照表	344
附录 B 运算符的优先级和结合性	345
附录 C 常用的 C 语言库函数	346
附录 D 调试程序时常见的出错提示信息	350
参考文献	352

C 语言概述

C 语言是当今国际上最流行的计算机程序设计语言之一，无论是开发系统软件，还是设计应用软件，都可以看到 C 语言的身影。人们之所以选择 C 语言，是源于它具有一定的独特优点。所以，了解它的特点很重要。本章主要介绍计算机语言、C 语言的历史及 C 语言的特点，并对 C 语言程序的组成和结构特点、对结构化程序设计做了简明阐述，解释 C 语言的执行过程，最后介绍 C 语言的基本符号。

1.1 计算机语言与结构化程序设计

1.1.1 计算机语言

计算机是一种具有内部存储能力的自动、高效的电子设备，它最本质的使命就是执行指令所规定的操作。需要计算机完成什么工作，只要将其步骤用诸条指令的形式描述出来，并把这些指令存放在计算机的内部存储器中，需要结果时就向计算机发出一个简单的命令，计算机就会自动逐条顺序执行操作，全部指令执行完就得到了预期的结果。这种可以被连续执行的一条条指令的集合称为计算机的程序。也就是说，程序是计算机指令的序列，编制程序的工作就是为计算机安排指令序列。但是，指令是二进制编码，用它编制程序既难记忆，又难掌握，所以，计算机研究人员就研制出了各种计算机能够懂得、人们又方便使用的计算机语言。这样程序就可以用计算机语言来编写了。

计算机程序设计语言的发展，经历了从机器语言、汇编语言到高级语言的历程。

1. 机器语言

计算机所使用的是由“0”和“1”组成的二进制数，二进制是计算机语言的基础。计算机程序是直接计算机能够识别的二进制代码指令进行书写的，这种程序设计语言就是机器语言。机器语言是第一代计算机语言，它通常由指示计算机一次完成一个最基本操作数值串组成。机器语言是直接对计算机硬件产生作用的，所以不同型号的计算机采用的机器语言是不一样的，这种情况使得机器语言很难被人们掌握和推广。但由于使用的是针对特定型号的计算机语言，故而运算效率是所有语言中最高的。

2. 汇编语言

一个复杂程序里的指令可能有成百万、成千万条或者更多，程序中的执行流程错综复杂，在二进制机器指令的层面上理解复杂程序到底做了什么，很容易变成人力所根本不能及

的事情。为缓解这一问题，人们研发了符号形式的、使用相对容易些的汇编语言。用汇编语言写的程序需要用专门软件（汇编系统）加工、翻译成二进制机器指令后才能在计算机上使用。比如，用“ADD”代表加法，“MOV”代表数据传递等，这样一来，人们很容易读懂并理解程序在干什么，纠错及维护都变得方便了。这种程序设计语言就称为汇编语言，即第二代计算机语言。然而计算机是不认识这些符号的，这就需要一个专门的程序，负责将这些符号翻译成二进制数的机器语言，这种翻译程序被称为汇编程序。

汇编语言同样十分依赖于机器硬件，移植性不好，但效率仍然较高。针对计算机特定硬件而编制的汇编语言程序，能准确发挥计算机硬件的功能和特长，程序精炼、质量高，所以，至今仍是一种常用而强有力的软件开发工具。

3. 高级语言

不论是机器语言还是汇编语言都是面向硬件的具体操作，语言对机器的过分依赖，要求使用者必须对硬件结构及其工作原理都十分熟悉，这点非计算机专业人员是难以做到的，对于计算机的推广应用也不利。为了计算机事业的发展，促使人们去寻求一些与人类自然语言相接近且能为计算机所接受的语意确定、规则明确、自然直观和通用易学的计算机语言。这种与自然语言相近并为计算机所接受和执行的计算机语言称为高级语言。经过努力，1954年，第一个完全脱离机器硬件的高级语言——FORTRAN问世了，40多年来，共有几百种高级语言出现，目前被广泛使用的高级语言有PASCAL、C、FORTRAN、VC、VB、C++、Java等。

当然，计算机也不能直接执行高级语言描述的程序。人们在定义好一个语言之后，还需要开发出一套实现这一语言的软件，这种软件被称做高级语言系统，也常被说成是这一高级语言的实现。在研究和开发高级语言的工作过程中，人们也研究了各种实现技术。高级语言的基本实现技术是编译和解释，下面是对编译和解释两种方式的简单介绍。

1) 采用编译方式实现高级语言。人们首先针对具体语言（例如C语言）开发出一个翻译软件，其功能就是将采用这种高级语言书写的程序翻译为所用计算机的机器语言的等价程序。用这种高级语言写出程序后，只要将它送给翻译程序，就能得到与之对应的机器语言程序。此后，只要命令计算机执行这个机器语言程序，计算机就能完成我们所需要的工作。

2) 采用解释方式实现高级语言。人们首先针对具体高级语言开发一个解释软件，这个软件的功能就是读入这种高级语言的程序，并能一步步地按照程序要求工作，完成程序所描述的计算。有了这种解释软件，只要直接将写好的程序送给运行这个软件的计算机，就可以完成该程序所描述的工作。

高级语言的发展也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。

高级语言的下一个发展目标是面向应用，也就是说，只需要告诉程序你要干什么，程序就能自动生成算法，自动进行处理，这就是非过程化的程序语言。

1.1.2 结构化程序设计

目前常见的程序设计方法有面向过程的设计方法和面向对象的程序设计方法。面向

过程的程序设计方法又称为结构化程序设计方法, 本节将介绍结构化程序设计方法及其基本思想。

1. 结构化程序设计

结构化程序设计 (structured programming, SP) 的概念最早是由荷兰学者 E.W.Dijkstra 提出来的, 目的是提高程序设计的质量。

目前尚没有一个严格的、为所有人普遍接受的定义。一个比较流行的定义是: 结构化程序设计是一种进行程序设计的原则和方法, 按照这种原则和方法设计出的程序的特点是, 结构清晰、容易阅读、容易修改、容易验证。

按照结构化程序设计方法的要求, 结构化程序由三种基本控制结构组成: 顺序结构、选择结构和循环结构。结构化程序的主体结构是顺序结构。

2. 结构化程序设计特征

结构化程序设计的特征主要有以下几点。

1) 以三种基本结构 (顺序、选择、循环) 的组合来描述程序。

结构化程序设计的三种基本控制结构的流程图表示如图 1-1 所示。

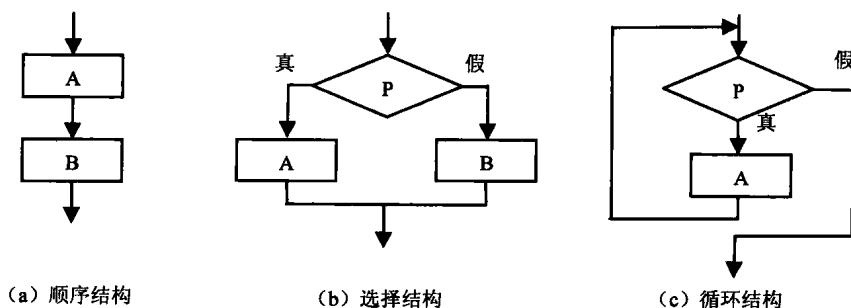


图 1-1 三种基本控制结构流程图

2) 程序设计采用“自顶向下, 逐步求精, 模块化设计, 结构化编码”的方法。程序设计时, 首先考虑程序的总体结构, 按程序实现的功能再细分为若干个子问题, 如果子问题还包含子问题, 再细化, 直到每一个细节均可以用高级语言清楚表达为止。这个过程就是“自顶向下, 逐步求精”。每个子问题都作为子程序, 在 C 语言中用函数表示, 又称为模块。这种层次分明、思路清楚、有条不紊的一步一步地进行设计的方法, 既严谨又方便。

3) 有限制地使用 goto 转移语句, 在非用不可的情况下, 也要十分谨慎, 并且只限于在一个结构内部跳转, 不允许从一个结构跳到另一个结构。这样可缩小程序的静态结构与动态执行过程之间的差异, 使人们能正确理解程序的功能。

4) 以控制结构为单位, 每个结构只有一个入口, 一个出口, 各单位之间接口简单, 逻辑清晰。

5) 采用结构化程序设计语言书写程序, 并采用一定的书写格式使程序结构清晰, 易于阅读。

6) 注意程序设计风格。采用顺序、选择和循环三种基本结构作为程序设计的基本单元, 避免无限制地使用 goto 语句而使流程任意转向。

3. 设计程序的过程

下面举例说明用结构化程序设计方法设计程序的过程。

【例 1.1】 求 3 个数中的最大数。

1) 给出程序的总体设计算法。

s1: 给定或输入 3 个数 a、b、c。

s2: 在 a、b、c 中找出大数赋给 max。

s3: 输出 max。

s1、s2、s3 表示第 1 步、第 2 步、第 3 步。

2) 对 s2 需进一步细化, 即求出最大数的设计算法。

s21: 从 a、b 中取大数赋给 max。

s22: 再用 max 与 c 进行比较, 取两者大的值赋给 max。

1)、2) 的流程描述如图 1-2 所示。

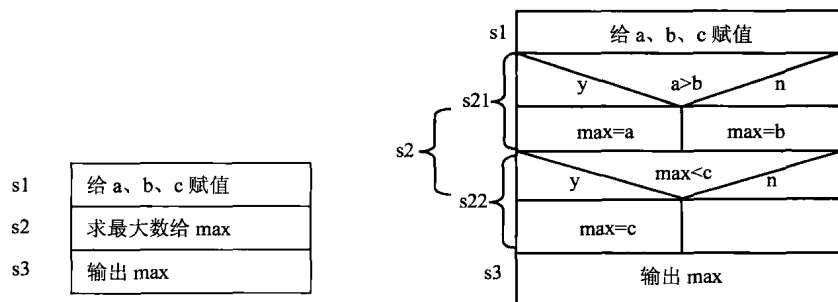


图 1-2 1)、2) 的流程图

3) 用计算机语言实现算法。

C 语言程序如下:

```
#include <stdio.h>
void main()
{
    int a,b,c,max;
    a=3;b=7;c=5;           /*s1,也可以使用 scanf() 对 a、b、c 赋值*/
    if(a>b)                 /*s21*/
        max=a;
    else
        max=b;
    if(max<c)               /*s22*/
        max=c;
    printf("max=%d\n",max); /*s3*/
}
```

结构化程序设计方法是公认的面向过程编程应遵循的基本方法和原则。结构化程序设计方法主要包括以下几点。

1) 只采用三种基本的程序控制结构来编制程序, 从而使程序具有良好的结构。

2) 程序设计自顶而下。

3) 用结构化程序设计流程图表示算法。有关结构化程序设计及方法有一整套不断发展和完善的理论和技术,对于初学者来说,完全掌握是比较困难的。但在学习的起步阶段就了解结构化程序设计的方法,学习好的程序设计思想,对今后的实际编程很有帮助。

1.2 C 语言的发展历史和特点

本书是介绍以 C 语言作为程序设计语言进行的程序设计,在开始对程序设计和 C 语言的系统学习之前,读者必须首先了解 C 语言的发展历史和 C 语言的基本特点。

1.2.1 C 语言的发展历史

C 语言是一种得到广泛重视并普遍应用的计算机程序设计语言,也是国际公认的最重要的几种通用程序设计语言之一,它既可用于编写系统软件,也可用于编写应用软件。

C 语言是由贝尔实验室的 Dennis Ritchie 和 Brian Kernighan 在 1972 年根据 Thompson 的 B 语言设计的,而 B 语言是由一种早期的编程语言 BCPL (basic combined programming language) 发展演变而来的。BCPL 的根源可以追溯到 1960 年的 ALGOL 60 (algol programming language),ALGOL 60 是一种面向问题的高级语言,离硬件较远。1963 年,英国剑桥大学推出 CPL (combined programming language) 语言,CPL 修改了 ALGOL 60,使其能够直接作较低层次的操作。1967 年英国剑桥大学的 Martin Richards 对 CPL 做了改进,推出了 BCPL 语言。

最初的 C 语言是为描述和实现 UNIX 操作系统提供的一种工具语言,但 C 语言并没有被束缚在任何特定的硬件或操作系统上,它具有良好的可移植性。1977 年出现了不依赖于具体机器的 C 语言编译文本——《可移植的 C 语言编译程序》,用该程序编写的 UNIX 系统迅速在各种机器上实现,UNIX 系统支持的 C 语言也被移植到相应的计算机上。C 语言和 UNIX 系统在发展过程中相辅相成,得到了广泛应用,使 C 语言先后被移植到各种大、中、小、微型计算机上。

以 1978 年发表的第 7 版 UNIX 系统中的 C 语言编译程序为基础,B.W.Kernighan 和 D.M.Ritchie 合著了《The C Programming Language》。这本书中介绍的 C 语言成为后来广泛使用的 C 语言版本的基础,被称为标准 C 语言。1983 年,美国国家标准化协会(ANSI)根据 C 语言问世以来的各种版本对 C 语言的发展和扩充制定了新的标准,称为 ANSI C。1990 年,C 语言成为国际标准化组织(ISO)通过的标准语言,并接受了 C90 作为国际标准 ISO/IEC9899: 1990。

1995 年,ISO 对 C90 做了一些修订,即“1995 基准增补 1 (ISO/IEC9899/AMD1: 1995)”。1999 年,ISO 又对 C 语言标准进行修订,在基本保留原来的 C 语言特征的基础上,针对应用的需要,增加了一些功能,尤其是 C++中的一些功能,命名为 ISO/IEC 9899: 1999。2001 年和 2004 年先后进行了两次技术修正,即 2001 年的 TC1 和 2004 年的 TC2。ISO/IEC 9899: 1999 及其技术修正被称为 C99,C99 是 C89 (及 1995 基准增补 1) 的扩充。本书的叙述以 C99 标准为依据,为了与 C89 作比较,在本书的叙述中对 C99 一部分新增加的功能作了特别的说明。

目前,由不同软件公司所提供的一些 C 语言编译系统并未完全实现 C99 建议的功能,

它们多以 C89 为基础开发,读者应了解自己所使用的 C 语言编译系统的特点。初学者所用到的初步编程知识基本上在 C89 的范围内,因此,使用目前的 C 编译系统仍然可以满足对初学者的教学需要,在今后进行实际软件开发工作时,应注意使用能在更大程度上实现 C99 功能的编译系统。本书中所举的示例程序都是在目前流行的编译系统 Visual C++ 6.0 上编译和运行。

1.2.2 C 语言的特点

C 语言经历了不断的发展和完善,成为当今计算机界公认的一种优秀程序设计语言,C 语言之所以能具有如此强大的生命力而存在和发展,是源于其有不可替代的特点。C 语言的特点可归纳为以下几点。

1. 数据类型丰富,具有现代语言的各种数据类型

C 语言的基本数据类型有整型、浮点型、字符型。在此基础上按层次可产生多种构造类型,如数组、结构体、共用体等,同时还提供了用户自定义数据类型。C99 又扩充了复浮点类型、超长类型和布尔类型,用这些数据类型可以实现复杂的数据结构,如栈、链表、树等。因此,C 语言具有较强的数据处理能力。

2. 结构化的程序设计语言简洁、紧凑

C 语言的主要组成部分是函数,函数允许一个程序中的各任务分别定义和编码,符合模块化原理。C 语言还提供了 3 种结构化的控制语句,如用于判定的 if-else、switch 语句,用于循环的 for、while、do-while 语句等,十分便于采用自顶向下、逐步细化的结构化程序设计方法。C 语言只有 37 个关键字,而一些高级语言,关键字多达 100 个,因此,用 C 语言编写的程序更加简单、容易理解和便于维护。

3. 具有丰富的运算符

在 C 语言中共有 34 个运算符,除了一般高级语言使用的算术运算、关系运算及逻辑运算等功能外,还具有独特的以二进制位(bit)为单位的位与、位或、位非以及移位操作等运算。并且 C 语言具有如 a++、b--等单目运算和+=、-=等复合运算功能。

4. 高效率的目标代码

C 语言允许直接访问物理地址,并能进行位(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功能,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。有人把 C 语言称为“高级语言中的低级语言”,也有人称它为“中级语言”,意为兼有高级和低级语言的特点。

5. 具有预处理能力

在 C 语言中提供了#include 和#define 两个预处理命令,实现对字符串的宏定义以及对外部文件的包含,同时还具有#if-#else-#endif 等条件编译预处理命令。这些功能的使用提高了软件开发的工作效率并为程序的组织和编译提供了便利。

6. 可移植性好

目前, C 语言在计算机上实现大部分是通过 C 语言编译系统移植得到的。C 编译程序的可移植性使 C 语言程序便于移植。用 C 语言编写的程序, 基本上可以不加修改地用于各种计算机和操作系统上。

7. 语法限制不严格, 程序设计自由度比较大

例如, 对数组下标越界不进行检查, 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, 整型量与字符型数据以及逻辑型数据可以通用。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, 因此放宽了语法检查。程序员应当仔细检查程序, 保证其正确, 而不要过分依赖 C 语言编译程序查错。“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。对于不熟练的人员, 编一个正确的 C 语言程序可能会比编一个其他高级语言程序难一些。也就是说, 对于用 C 语言的人, 要求更高些。

总之, C 语言已成为国内外广泛使用的一种编程语言。

1.3 C 语言程序的基本结构和执行过程

如果要很好地使用 C 语言去编写程序, 就要非常熟练地掌握 C 语言基本知识, 本节将通过几个简单的程序来帮读者加深了解如何阅读 C 语言程序和使用 C 语言程序, 为以后进行 C 语言编程打下坚实的基础。

1.3.1 C 语言程序简单实例

C 语言是函数式语言, 函数是构成 C 语言程序的基本单位。下面通过一个程序来分析 C 语言程序的组成和结构。

【例 1.2】 求两个数的最大值。

程序如下:

```
#include <stdio.h>           /*文件包含*/
int max(int m,int n);        /*函数说明*/
void main()                  /*主函数*/
{
    int x,y,z;               /*变量说明*/
    printf("input two numbers:\n");
    scanf("%d%d",&x,&y);      /*输入 x、y 值*/
    z=max(x,y);              /*调用 max() 函数*/
    printf("maximum=%d",z);   /*输出*/
}
int max(int m,int n)         /*定义 max() 函数*/
{
    if(m>n)
        return m;
    else
        return n;           /*把结果返回主调函数*/
}
```

```
    }
```

程序运行情况如下（本书中所有输入数据均由下划线标识）：

```
input two numbers:
 2 3
maximum=3
```

程序分析：本程序包括两个函数定义体，主函数 `main()` 和被调用函数 `max()`。

程序第1行是预编译命令 `#include <stdio.h>`，表示包含程序中使用的标准输入输出库函数文件（后面章节将详细介绍）。

程序第2行是对被调用函数 `max()` 的声明。为什么要作这个函数声明呢？因为在主函数中要调用 `max()` 函数（主函数中语句“`z=max(x,y);`”），而 `max()` 函数的定义部分却在 `main()` 函数定义之后，计算机对程序的编译是自上而下进行的，在对程序中调用语句进行编译时，编译系统无法知道 `max` 是什么，因而无法把它作为函数调用处理。为了使编译系统能识别 `max()` 函数，就要在调用 `max()` 函数之前用语句“`int max(int m,int n);`”对 `max()` 函数进行“声明”，所谓声明，通俗地说就是告诉编译系统 `max` 是什么，以及它的有关信息（有关函数的声明将在第8章详细介绍）。

`max()` 函数的作用是将 `x` 和 `y` 中较大者返回到主函数中。`max()` 下面用花括号括起来的是 `max()` 函数的函数体。在函数体中，通过 `if-else` 语句判断，如果判定条件成立，由第14行的 `return` 语句将 `m` 的值作为 `max()` 的函数值返回给调用 `max()` 函数的函数（即主函数 `main()`），如果判定条件不成立，则由第16行 `return` 语句将 `n` 的值返回。返回值是通过函数名 `max()` 带回到 `main()` 函数中去的（带回到语句“`z=max(x,y);`”，`main()` 函数调用 `max()` 函数处）。

程序第3行中 `main` 是主函数的名称，`main` 前面的 `void` 表示函数 `main()` 没有返回值，每个程序都有一个主函数，函数体由花括号括起来，第4~10行是主函数的函数体。

程序第5行是对变量的声明，表示定义了3个整型变量，变量名称分别为 `x`、`y`、`z`。

程序第6行 `printf` 是输出函数的名称（`scanf()` 和 `printf()` 都是C语言的标准输入输出函数）。本例中 `printf()` 函数的作用是原样输出字符串“`input two numbers:`”，也就是在屏幕上显示该字符串，其中“`\n`”是回车换行的含义，使得光标自动移到下一行行首。

程序第7行 `scanf` 是输入函数的名称。`scanf()` 函数的作用是输入变量 `a` 和 `b` 的值。`scanf()` 函数后面圆括号中包括两部分内容：一是双引号中的内容，它指定输入的数据按什么格式输入，“`%d`”的含义是十进制整数形式；二是输入的数据准备放到哪里，即赋给哪个变量。现在，`scanf()` 函数中指定的是 `a` 和 `b`，在 `a` 和 `b` 的前面各有一个“`&`”，在C语言中“`&`”是取地址符，`&a` 的含义是“变量 `a` 的地址”，`&b` 是“变量 `b` 的地址”。执行 `scanf()` 函数，从键盘读入两个整数，送到变量 `a` 和 `b` 的地址处，然后把这两个整数分别赋值给变量 `a` 和 `b`。

程序第8行利用 `max(a,b)` 调用 `max()` 函数。在调用时，将 `x` 和 `y` 作为 `max()` 函数的参数（称为实际参数）的值分别传送给 `max()` 函数中的参数 `m` 和 `n`（称为形式参数），然后执行 `max()` 函数的函数体（程序第12~17行），使 `max()` 函数返回一个值（即 `m` 和 `n` 中大者的值），`return` 语句的作用是把函数值带回到程序的主函数调用 `max()` 函数的位置，取代 `max(a,b)`，然后把这个值赋给变量 `z`。

第9行输出结果。在执行 `printf()` 函数时，对双引号括起来的“`maxmun=%d`”是这样处