HZ BOOKS
华章教育

高等学校计算机专业规划教材

C语言程序设计

第2版

王立柱 主编

The C Programming Language
Second Edition

机械工业出版社
China Machine Press

高等学校计算机专业规划教材

C语言程序设计

第2版

王立柱 主编

欧阳勇 杜江毅 程玉 李昌 副主编

The C Programming Language
Second Edition

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

C 语言程序设计 / 王立柱主编 . —2 版 . —北京：机械工业出版社，2016.2
（高等学校计算机专业规划教材）

ISBN 978-7-111-52927-9

I. C… II. 王… III. C 语言 – 程序设计 – 高等学校 – 教材 IV. TP312

中国版本图书馆 CIP 数据核字（2016）第 028750 号

本书采用辩证逻辑叙述方法，从程序设计中的问题出发学习 C 语言知识。全书共 11 章，内容包括：机器语言模式、C 语言模式、函数、一维数组和指针、字符串、结构体、联合体、枚举、顺序表、链表、文件、二维数组和指针、非线性结构与递归。

本书内容详实连贯，结构清晰严谨，目标具体明确，既可作为本科及大专院校 C 语言程序设计课程的教材，也可作为编程爱好者的自学参考书。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：佘　洁　　责任校对：董纪丽
印　　刷：北京诚信伟业印刷有限公司　　版　　次：2016 年 3 月第 2 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：12.5
书　　号：ISBN 978-7-111-52927-9　　定　　价：35.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

前　言

本书采用辩证逻辑叙述方法。所谓辩证，是指从程序设计的基本矛盾出发，说明概念的必然发展过程。所谓逻辑，是指从机器语言程序出发，逐步分析和引出有关程序设计的概念。程序设计的基本矛盾是数据结构和算法的矛盾。数据结构即一组有组织的数据和对数据的基本操作，一般用程序语言来表示；算法是用基本操作表示的数据处理的有限步骤。算法越来越复杂，同时要求程序易写、易读、易维护且足够高效，这就推动了程序语言的发展，而发展的程序语言又反过来扩展了算法。

本书从程序设计中的问题出发学习各种概念：每一个问题都是程序设计中的问题，每一个问题的解决都产生新的程序以及新的概念。读者从本书中所感受到的，不是一个高高在上的思想者的存在，即他告诉你做什么，而是读者自身的存在，即读者知道自己要做什么，并且可以描述出每一个问题的产生过程和解决方案。

本书内容概要

第 1 章机器语言模式。首先用若干条虚拟的机器指令模拟一台计算机，通过一个简单的求和程序引入存储器、地址、寄存器、程序计数器等概念。然后用一个方程的两种求解程序，说明算法和指令集的相互作用关系，同时引入程序入口地址、子程序调用、断口地址、现场保护、栈等概念。最后，通过分析机器语言的局限性，引入汇编语言、汇编程序等概念。本章的重点不是机器语言程序设计，而是理解有关程序设计的一些基本概念，以及数据结构和算法这一对矛盾最初的表现形式。

第 2 章 C 语言模式。首先分析汇编语言的局限性，由此引出高级程序语言——C 语言。然后通过将一个整数按位逆置的各种算法，引出 C 语言基于基本类型的编程模式，包括主函数框架、标准输入 / 输出函数、表达式、程序流程控制结构等基本知识。最后分节详述这些基本知识。

第 3 章函数。首先将一个整数按位逆置的算法设计为一个函数，然后逐步引入函数的相关知识。核心内容是函数调用的三个步骤：一是主调函数通过实参对

被调函数的形参进行初始化；二是被调函数如果有返回值，系统就根据返回值类型创建一个临时变量，以初始化方式存储这个返回值；三是主调函数如果需要被调函数的返回值，就从这个临时变量取值。

第 4 章一维数组和指针。由一个整数的按位逆置，到一个整数的按位求和，再到一组整数求和，通过算法的逐步深入，自然而然地揭示出对指针的需求。然后，通过“对象即类型化的空间”这个基本概念，结合程序设计的需要，简洁明了地把指针“设计”出来。

第 5 章字符串。字符串是标准库提供的一种数据结构。字符数组加串结束符是它的存储结构，串长度、串复制、串连接等是它的基本操作。该章不仅实现了字符串的基本操作代码，而且补充了一些基本操作，并在此基础上实现了模式匹配。

第 6 章结构体、联合体和枚举。这是基本类型的扩展。

第 7 章顺序表。首先分析了数组的局限性，然后建立顺序表克服这些局限性。

第 8 章链表。首先分析了顺序表这种连续存储结构的局限性，然后建立链表克服这种局限性。

第 9 章文件。文件主要是指磁盘文件，是对标准输入 / 输出文件的扩展。该章通过对比引出磁盘文件的读写操作。

第 10 章二维数组和指针。该章是第 4 章一维数组和指针的自然扩展，对数组和指针的一系列概念进行了严谨而简洁的综述。

第 11 章非线性结构与递归。首先通过英语的树形结构引入递归概念，然后将递归算法与非线性结构联系起来进行讲解。

教辅资源

本书配有多媒体教学课件，大部分程序在课件中都有结构、算法、代码、运行及结果的跟踪演示，而且允许自由输入数据。

用书教师可访问华章网站（www.hzbook.com）下载。

致读者

读者如有问题，可以直接与作者联系：tjwanglizhu@163.com。

教学建议

教学章节	教学要求	课时 + 实验
第 1 章 机器语言模式	熟悉例 1.1 和例 1.2 了解存储器、地址、寄存器、程序计数器、程序入口地址、子程序调用指令、断口地址、现场保护、栈、汇编语言、汇编程序的概念	2
第 2 章 C 语言模式	熟悉程序 2.1 ~ 程序 2.11，以及函数 Josephus 掌握 if-else 语句、循环结构语句 熟悉 switch-case 语句	4+4
第 3 章 函数	掌握函数定义和调用过程、函数原型、变量的存储类别、函数应用设计举例 了解模块化程序设计、编译预处理等	6+6
第 4 章 一维数组和指针	掌握一维数组和指针的定义、赋值、应用举例。可根据具体情况，选择性地掌握相应算法	4+4
第 5 章 字符串	掌握字符串的定义、赋值和基本操作函数 熟悉字符串基本操作函数的实现 了解取子串、子串插入、子串删除、字符查找、模式匹配等算法	4+4
第 6 章 结构体、联合体和枚举	掌握结构体的定义，结构体变量的定义、赋值，结构体数组的定义、赋值，应用举例中的 Date 结构 熟悉联合体与枚举 了解应用设计举例中的其他算法	2+2
第 7 章 顺序表	通过 purge_array 算法了解数组的局限性，掌握顺序表的声明和实现 了解数据抽象与封装	4+4
第 8 章 链表	掌握链表的声明 了解链表基本函数的实现	2+2
第 9 章 文件	掌握文件的打开与关闭操作 熟悉文件的各种读写操作	2+2
* 第 10 章 二维数组和指针	（本章为选讲内容） 掌握二维数组与二维指针的定义、赋值，掌握二维数组与一维指针的关系 了解指针数组的应用	2+2
* 第 11 章 非线性结构与递归	（本章为选讲内容） 熟悉汉诺塔、快速排序和八皇后算法 掌握递归概念和归纳证明方法	4+4
总课时		70

目　录

第 1 章　机器语言模式

指令集结构是对实际处理硬件的抽象。使用这个抽象，机器代码程序表现得就像它是运行在一个一次只执行一条指令的处理器上。

——Randal E Bryant

机器语言（machine language）是一个机器指令集，称为**机器码**（machine code），它们直接联系着计算机硬件，每一条机器指令都是计算机可以直接执行的。学习机器语言程序设计，需要深入了解计算机体系结构。但是本章的重点不是机器语言程序设计，而是数据结构和算法这对矛盾最初是如何表现的，因此基本上不涉及计算机体系结构，只用若干条虚拟的机器指令集来模拟计算机。

1.1　模拟机器指令集与程序设计举例

在机器语言程序设计中，程序员亲自安排数据的存储，然后调用机器指令处理数据。

一条机器指令包含操作码和操作数两部分，前者是操作内容，后者是数据所在的存储单元地址，有时直接就是数据。例如，“01H 3000H”是一条机器指令，其中 01H 是操作码，3000H 是操作数地址，具体意思是：取出地址为 3000H 存储单元中的数据，存入 CPU 的累加器 A 中。其中 H 表示操作码 01 和地址 3000 都是用十六进制表示的二进制数。一位十六进制和二进制的对应关系如表 1-1 所示。

表 1-1　1 位十六进制数和 4 位二进制数的对应表

十六进制	二进制	十六进制	二进制	十六进	二进制	十六进制	二进制
0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

计算机存储器是一系列连续的字节，每一字节有 8 位。指令“01H 3000H”需要 3 字节。

表 1-2 是一台模拟计算机的机器指令集。

表 1-2 一台模型计算机的机器指令集

指令名称	机器指令		指令说明
	操作码	操作数	
取数	01H	N	A ← (N) 从地址为 N 的单元读取数据，存入累加器 A
直接取	10H	C	A ← C 把数据 C 存入累加器 A
存数	02H	N	(N) ← A 从累加器 A 中读取数据，存入地址为 N 的单元
加法	03H	N	A ← A+(N) 将地址为 N 的单元中的数据经过某一个寄存器和累加器 A 中的数据相加，结果存入 A
直接加	30H	C	A ← A+C 将数据 C 经过某一个寄存器和累加器 A 中的数据相加，结果存入 A
乘法	04H	N	A ← A × (N) 将地址为 N 的单元中的数据经过某一个寄存器和累加器 A 中的数据相乘，结果存入 A
转移	06H	N	PC ← N 用地址 N 更新程序计数器 PC 的值
停机	00H		停机

例 1.1 编程计算 $z=x+y$。

算法步骤：

1）从 x 中读数。

2）从 y 中读数，计算 $x+y$。

3）把求和结果存到 z。

首先把 x、y 和 z 安排在存储器数据区，地址依次为 3000H、3001H 和 3002H，如图 1-1a 所示。

算法的步骤可以用表 1-2 中的机器指令表示，其对应关系如表 1-3 所示。

表 1-3 求和算法和程序

算法步骤	机器指令		指令说明
从 x 中读数	01H	3000H	取出地址为 3000H 单元中的数据，存入寄存器 A
从 y 中读数，计算 $x+y$	03H	3001H	将地址为 3001H 单元中的数据与寄存器 A 中的数据相加，结果留在 A
把求和结果存到 z	02H	3002H	将寄存器 A 中的数据存入地址为 3002H 的单元
	00H		停机

机器指令位于存储器的程序区，除停机指令之外，其他指令都占 3 字节，机器指令首字节地址称为指令地址。与算法步骤对应的 4 条机器指令的地址依次是 2000H、2003H、2006H 和 2009H，如图 1-1 所示。CPU 中的程序计数器 PC 存储机器指令地址，CPU 不断从 PC 中提取指令来执行，直到停机为止，每执行一条指令，PC 都会指向下一条指令。表 1-3 的程序执行过程如图 1-1 所示。

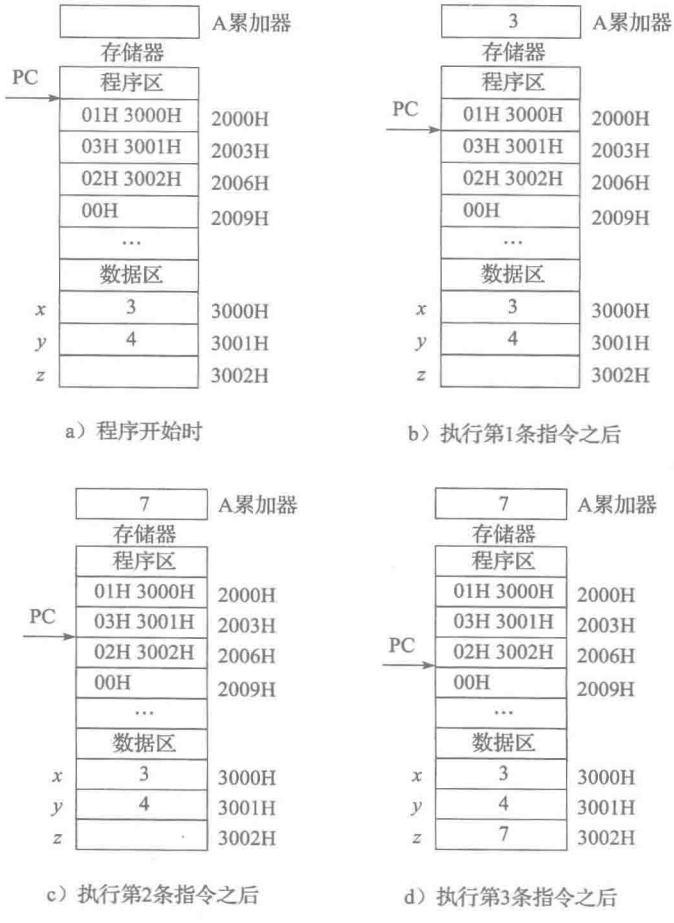

图 1-1 求和程序 $z=x+y$ 的执行过程示意图

例 1.2 编程计算 $y=2x^2+3x+4$。

将原方程分解为 $y=(2x+3)x+4$，据此得到算法步骤如下：

1）取 2。

2）从 x 中读数，计算 $2\times x$。

3）取 3，计算 $2x+3$。

4）从 x 中读数，计算 $(2x+3)x$。

5）取 4，计算 $(2x+3)x+4$。

6）将计算结果存入 y。

将 x 和 y 安排在存储器的数据区，地址分别为 3003H 和 3004H。

算法步骤可以用表 1-2 中的机器指令来表示，其对应如表 1-4 所示。

表 1-4 例 1.2 的算法和程序

算法步骤	机器指令		指令说明
取 2	10H	2	A ← 2
从 x 中读数，计算 $2x$	04H	3003H	A ← A*(3003H)
取 3，计算 $2x+3$	30H	3	A ← A+3
从 x 中读数，计算 $(2x+3)x$	04H	3003H	A ← A × (3003H)
取 4，计算 $(2x+3)x+4$	30H	4	A ← A+4
将计算结果存入 y	02H	3004H	(3004H) ← A
	00H		停机

例 1.2 的算法还可以用如下步骤描述：

1）取 2。

2）从 x 中读数，计算 $2x$。

3）从 x 中读数，计算 $2x^2$。

4）取 3。

5）从 x 中读数，计算 $3x$。

6）计算 $2x^2+3x$。

7）取 4，计算 $2x^2+3x+4$。

但是，利用表 1-2 的指令集，步骤 4 是无法直接实现的。因为在执行步骤 4 之前，累加器 A 存储着 $2x^2$ 的计算结果，这个结果以后还有用，而步骤 4 需要累加器来存储“3”，这与累加器产生了冲突。解决的办法是，设计一个新的变量，例如 z，在执行步骤 4 之前，用操作码为 02H 的指令把累加器中 $2x^2$ 的计算结果保存到 z，然后执行步骤 4 和 5，接下来，在步骤 6 中，利用操作码为 03H 的指令，读取 z 中的值与累加器 A 相加。

从算法的实现来讲，这涉及计算的中间结果保护和 CPU 的资源共享问题，这是一个复杂的问题，是一个程序员无法驾驭的；而从算法的设计来讲，这又是一个化复杂为简单的普遍方法。于是产生了一个尖锐的矛盾：算法越来越复杂，而机器指令限制了算法设计的正常发展。解决这个矛盾的方法是增加相关的机器指令，如表 1-5 所示。

表 1-5 与子程序调用有关的指令

指令名称	机器指令		指令说明
	操作码	操作数	
子程序调用	07H	N	断口地址进栈，用地址 N 更新程序计数器 PC 的值
返回主程序	08H		断口地址出栈，用来更新程序计数器 PC 的值
压栈	09H	T	寄存器 T 中的数据进栈保存
出栈	0aH	T	出栈，出栈数据进入寄存器 T

下面结合表 1-5，介绍一些概念：

入口地址。程序是一组指令，第一条指令的地址称为程序的入口地址。例如，例 1.1 中的 2000H。

中断和断口地址。一个程序在执行过程中去执行另一个程序，我们称之为中断，前者称为主程序，后者称为子程序。主程序中断时的下一条指令的地址称为主程序的断口地址。

现场保护和现场恢复。在主程序中断时，如果有些寄存器还存放着主程序的中间结果，那么就要将中间结果放入异地暂存，把寄存器“让给”子程序，这个过程是现场保护。待子程序执行完毕，再将主程序的中间结果放回寄存器，这个过程是现场恢复。

子程序调用过程。当主程序转去执行子程序时，首先将断口地址压入栈保存，再将子程序的入口地址送入 PC。进入子程序后，首先保护主程序现场，然后执行子程序，子程序执行完毕后，恢复主程序现场，然后将断口地址从栈顶送回 PC，过程如图 1-2 所示。

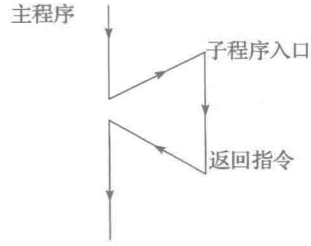

图 1-2　子程序调用过程

例 1.3　在增加了子程序调用指令的条件下，重新编程计算 $y=2x^2+3x+4$。

首先将方程分解为两个简单的方程：

1）$y=2x^2+z+4$

2）$z=3x$

把求解第一个方程的程序称为**主程序**，求解第二个方程的程序就是**子程序**。算法设计和程序清单见表 1-6。

表 1-6　例 1.3 算法和程序清单

算法	存储器	指令地址	指令说明
求解 $y=ax^2+z+c$ 的算法	主程序区		
取 2	10H　2	2000H	A ← 2
计算 $2x$	04H　3003H	2003H	A ← A × (3003H)
计算 $2x^2$	04H　3003H	2006H	A ← A × (3003H)
调用子程序计算 $z=3x$	07H　2020H	2009H	PC ← 2020H，栈← 200c H

（续）

算法	存储器		指令地址	指令说明
求解 $y=ax^2+z+c$ 的算法		主程序区		
计算 $2x^2+z$		03H 3005H	200cH	A ← A+(3005H)
计算 $2x^2+z+4$		30H 4	200fH	A ← A+4
计算结果存入 y		02H 3004H	2012H	(3004H) ← A
算法结束		00H	2015H	停机
求解 $z=bx$ 的算法		子程序区		
寄存器 A 中的数据进栈（现场保护）		09H A	2020H	栈← A
取 3		10H 3	2023H	A ← 3
计算 $3x$		04H 3003H	2026H	A ← A × (3003H)
计算结果存入 z		02H 3005H	2029H	(3005H) ← A
栈中的数据进入寄存器 A（恢复现场）		0aH A	202cH	A ←栈
返回主程序		08H	202fH	PC ←栈
		数据区	数据地址	
	x	5	3003H	
	y		3004H	
	z		3005H	

在计算 $y=2x^2+3x+4$ 的算法设计中，例 1.2 将其分解为 $y=(2x+3)x+4$，这是因为机器指令集没有子程序调用指令，这是数据结构对算法的限制。例 1.3 是化复杂为简单的典型方法，但要求补充子程序调用指令，这是算法对数据结构的规定。因为任何限制或规定都是一种否定，所以这是数据结构和算法在机器语言层面上的斗争。

由于计算机的硬件成本不断下降，软件开发成本不断提高，人们开始扩充机器指令系统，将使用频率高、执行时间长的指令序列，即算法，用一条新的复杂机器指令来代替，使算法设计步骤更容易实现，由此产生了**复杂指令系统计算机**（Complex Instruction Set Computer，CISC）。相反，把使用频率在 80% 以上、能在一个时钟周期内执行完毕、在指令系统中仅占 20% 的简单机器指令保留下来，把其他 80% 的复杂机器指令改由子程序来实现，即将指令转为算法，以减少指令和寻址方式的种类、固定指令格式、提高性能价格比。由此产生了**精简指令系统计算机**（Reduced Instruction Set Computer，RISC）。这是数据结构和算法在机器语言层面上的相互转化。

1.2 机器语言的局限性

机器语言的局限性很明显：机器指令是用二进制编码的，不容易记忆和阅读。为了克服这种局限性，引入一种替代方法：用助记符来代替操作码，用符号代替地址。助记符是缩写的英文字符，与操作码的功能相对应；表示地址的符号即符

号地址，由用户根据需要来确定。这种由助记符和符号组成的指令集合称为**汇编语言**。汇编语言程序必须经过翻译，转变为机器语言程序，才能被计算机执行。完成这一翻译任务的程序叫做**汇编程序**，它是系统软件，一般是与计算机设备一起配置的。利用汇编程序将汇编语言程序翻译为机器语言程序的过程称为**汇编**。汇编语言程序称**源代码**，翻译后的机器语言程序称为**目标代码**。如图 1-3 所示。

图 1-3 汇编示意图

机器语言的另一个局限性是对计算机系统结构的依赖，在一种机器上开发的机器语言程序在另一种机器上一般是不能执行的。汇编语言克服了这个局限性，因为**汇编程序**是与计算机设备一起配置的系统软件，它可以把汇编语言程序翻译为相应的机器语言程序。

机器语言还有一个局限性：机器指令与硬件结构相联，它对数据的操作很简单，等同于最简单的机械动作，这使得对一个像求和一样的简单计算，都需要多条机器指令才能完成，随着算法越来越复杂，用这种简单的机器指令来描述算法肯定越来越困难，而且阅读、修改和维护也越来越不容易。这个矛盾在汇编语言中依然存在，因为汇编指令不过是符号化的机器指令而已。要有效地解决这种矛盾，就要超越计算机硬件结构，建立面向算法处理对象的程序语言。这便是高级语言。

问题与练习

分别用顺序结构和子程序调用结构两种方法编程计算 $y=7x^3+6x^2+5x+3$。

第2章　C语言模式

机器语言是二进制代码语言，汇编语言是符号化的机器语言，高级语言是算法语言。

运算符表达式，例如例1.1中的$x+y$，在机器语言中是算法，而在C语言中是指令，是基本操作。用运算符指令来描述的算法，是C语言程序的基本形式。运算符表达式的内容和意义按类型划分，这种基于类型的语言称为高级语言。

C语言中的运算符指令是计算机无法识别的，但是每一条运算符指令都可以转换为一组对应的机器语言指令，就像例1.1中的求和程序那样，因此，用运算符指令描述的C语言程序也可以转换为机器语言程序。这项工作由**编译程序**（也称**编译器**）来完成。高级语言程序（源代码）转换为机器语言程序（目标代码）的过程称为**编译**。如图2-1所示。

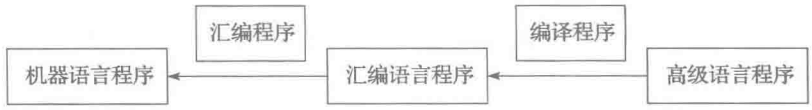

图2-1　编译示意图

低级语言的局限性在高级语言中被克服了，但是数据结构和算法的矛盾并没有消失，只是表现为不同的形式，因为算法是不断发展的。下面在高级语言学习中继续考察这个矛盾。

2.1　基于基本类型的编程模式

计算机内存是一个物理上连续的字节序列，每一个字节都有一个用无符号整数表示的地址。相邻的字节，地址相差1。在32位操作系统中，字节数是2^{32}（4GB）。如果用十六进制表示地址，那么取值范围是0x00000000 ~ 0xffffffff。

数据在C语言中的存在方式可以用（地址，类型）来概括。从一个地址所标识的字节开始，由连续多少个字节构成一种数据的存储空间、以什么格式来存储这种数据以及对这种数据可以实施哪些基本操作，都由类型来决定。