普通高等教育工程应用型系列规划教材

C语言程序设计教程

王明军　刘培奇
叶娜　钱文珺　张鹏
编著

科学出版社

普通高等教育工程应用型系列规划教材

C 语言程序设计教程

王明军　刘培奇　叶　娜　钱文珺　张　鹏　编著

科学出版社

北　京

内 容 简 介

本书以 C 语言的基本语法、语句和算法为基础，结合编者多年的教学经验和实践，深入浅出、循序渐进地介绍 C 语言的基本思想和程序设计方法。为培养和提高学生分析问题和解决问题的能力，书中增加了案例程序的设计思想和程序注释。为巩固学习内容，在每章后配有习题。本书共 12 章，在对 C 语言进行综合概述的基础上，分别介绍 C 语言程序设计基础、分支程序设计、循环程序设计、数组、函数、字符串处理、指针、结构体和共用体、文件、位运算和编译预处理等内容。

本书在编写过程中力求语言简洁、通俗易懂、系统完整，可作为大学本科、专科院校和独立学院的程序设计教材，也可作为广大科技人员和自学 C 语言人员的参考书。

图书在版编目(CIP)数据

C 语言程序设计教程 / 王明军等编著. —北京：科学出版社，2016.5
普通高等教育工程应用型系列规划教材
ISBN 978-7-03-048106-1

Ⅰ. ①C… Ⅱ. ①王… 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(2016)第 085894 号

责任编辑：张丽花 李 清 / 责任校对：桂伟利
责任印制：霍 兵 / 封面设计：迷底书装

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

新科印刷有限公司 印刷

科学出版社发行 各地新华书店经销

*

2010 年 5 月第 一 版 开本：787×1092 1/16
2016 年 5 月第一次印刷 印张：16 1/2
字数：391 000

定价：39.00 元

(如有印装质量问题，我社负责调换)

前　言

C 语言是计算机的一门重要的编程语言，以简洁、灵活和功能强大而著称，特别是在计算机操作系统和硬件级别上的操作，深受广大编程人员的青睐。目前，“C 语言程序设计”是大学计算机专业及理工类非计算机专业的重要基础课程之一。

编者针对大学本科阶段理工类专业计算机课程设置的特点，根据国家计算机教学指导委员会教育教学改革、教学实践以及人才培养等相关要求，结合多年的教学实践经验，在对 C 语言程序设计的基本内容进一步优化、补充和完善的基础上编写本书。本书主要以计算思维为导向，理论联系实际，引导和启发学生从计算机的角度思考和解决问题，注重培养学生分析问题和解决问题的能力。在本书编写中，内容力求循序渐进、难题简化、重点突出，聚焦学生学习 C 语言程序设计中的基本语法、基本语句和基本算法，以简洁、通俗、易懂的语言风格介绍 C 语言的基本内容，按照各章节的知识点和难点选配例题和习题。

本书共 12 章及附录。在对 C 语言进行综合概述的基础上，分别介绍 C 语言程序设计基础、分支程序设计、循环程序设计、数组、函数、字符串处理、指针、结构体和共用体、文件、位运算和编译预处理等内容。为便于读者自学，与本书配套使用的《C 语言程序设计习题与解析》在总结归纳每章重点内容的同时，配有例题及习题，进一步加深理解程序设计分析和编程方法。

本书由西安建筑科技大学王明军、刘培奇、叶娜、钱文珺和张鹏共同参加编写。其中第 1、3 章由钱文珺编写，第 2、4、5 章由叶娜编写，第 6、9、11 章由张鹏编写，第 7、8 章由刘培奇编写，第 10、12 章由王明军编写。王明军对全书进行了统编。

在本书的编写中，刘家全教授审阅了全书，并提出许多宝贵的修改意见，在此对刘家全教授的辛勤劳动表示衷心感谢。本书的出版得到了科学出版社工科分社领导和员工的鼎力帮助和支持，特别是匡敏社长及李清编辑给予了大力支持。另外，在本书的编写过程中，还查阅了大量的文献资料，在此对出版社领导和文献资料作者的辛勤劳动一并致以衷心的感谢。

由于作者水平有限，虽然在本书的成书过程中进行了反复修改完善，但书中的不足之处在所难免，敬请广大读者批评指正。

编　者

2015 年 12 月于西安

目　录

第 1 章　C 语言概述

计算机是由硬件系统和软件系统两部分组成的，如果只有硬件而没有软件，计算机就不能实现任何功能。而软件的编制离不开程序设计语言(或称编程语言)。在众多程序设计语言中，C 语言既具有高级语言的优点，又具有低级语言的特性，因而备受软件开发者的欢迎。

本章从计算机与程序的关系出发，介绍程序设计的一般过程，引入算法与结构化程序设计，回忆程序设计语言的发展过程，对 C 语言及其程序设计过程进行简单介绍，最后阐述在 Turbo C 环境下开发 C 程序的过程，并对 C 程序的基本构成进行分析。

1.1　计算机与程序

有人以为计算机是“万能”的，会自动进行所有工作，这是很多初学者的误解。其实计算机的每一个操作都是根据人们事先制定的指令进行的。例如，用一条指令要求计算机进行一次加法运算，用另一条指令要求计算机将某一运算结果输出到显示屏等。为了使计算机执行一系列的操作，必须事先编好一条条指令，输入到计算机中。

计算机能够识别和执行的、按照一定次序事先编制好的、能完成特定功能的指令序列就是程序。程序通常是使用某种程序设计语言编写的，运行于某种目标体系结构上。打个比方，一个程序就像一个用汉语(程序设计语言)写下的红烧肉菜谱(程序)，用于指导懂汉语的人(体系结构)来做这个菜。每一条指令是计算机执行的特定操作，一组特定的指令集(一个程序)用来实现一个功能。只要让计算机执行这个程序，计算机就会“自动地”执行指令，有条不紊地进行工作。为了使计算机系统能实现各种功能，需要成千上万个程序。这些程序大多是由计算机软件设计人员根据需要设计好的，作为计算机软件系统的一部分提供给用户使用。

总之，计算机的一切操作都是由程序控制的，离开程序计算机将一事无成。所以，计算机的本质是程序的机器，程序和指令是计算机系统中最基本的概念，只有懂得程序设计才能真正了解计算机是怎样工作的，才能更深入地使用计算机。

1.2　程序设计的一般过程

要让计算机完成特定功能，首先要设计、编写出一个能够正确指导计算机完成这个功能的程序，这就是程序设计。一般情况下，程序设计包括下面几个阶段。

(1)分析问题，建立模型。要先明确让计算机解决的问题是什么，需要输入什么数据，经过计算机处理后要取得什么结果或效果。针对具体问题，建立合适的数学模型。

(2)设计算法与数据结构。算法是指解决问题的具体方法与步骤，是对解决方案的一种准确而完整的描述。对同一个问题，通常可以采取不同的解决方法和步骤，也就是采用不

同的算法，因此需要从多种算法中选择一个相对较优的。数据结构是计算机存储、组织数据的方式。好的数据结构决定了软件系统实现的难易程度和系统的质量，因此对数据结构的设计与选择是程序设计的一个基本考虑因素。

(3)编写程序(简称编程或编码)。将算法用某种程序设计语言进行描述。

(4)检查并确定最终程序。当编写好程序后，可以采用所用程序设计语言的开发工具对程序进行检查，找到其中的错误并进行修正，直到确定出能解决最初问题的有效程序。

(5)撰写文档。对于大型软件系统，当需要把软件程序交付给用户使用时，需要向用户提供软件使用说明书，说明程序的功能、运行环境等；作为软件程序的开发者，需要对程序设计与开发调试的过程进行记录，如写设计文档、测试文档等，便于日后对软件程序的维护。

【例 1.1】 根据产品数量和单价计算产品总价，当产品数量大于 50 时打 8 折。

根据以上描述设计程序的基本步骤，需要经过下列步骤编写出能够实现题目所要求功能的程序。

(1)分析题目要求可知，需要根据输入的产品数量(假设用 count 表示)和单价(假设用 price 表示)，计算产品总价(假设用 total 表示)，产品总价的计算可依据下面的数学模型：

$$\text{total}=\begin{cases}\text{count}\times\text{price}, & \text{count}\leqslant 50\\ \text{count}\times\text{price}\times 0.8, & \text{count}>50\end{cases}$$

(2)算法有多种描述方式，假设选择使用自然语言来描述，则解决该问题的算法如下。

第一步：输入 count 和 price 的值。

第二步：如果 count≤50，则将 total 置为 count*price，否则将 total 置为 count*price*0.8。

第三步：输出 total 的值。

(3)如果选择 C 语言作为程序设计语言，根据上述数学模型与算法，可以编写出如下程序代码：

```
#include<stdio.h>
void main()
{
    int count;
    float price, total;
    printf("\n 请输入产品数量:");
    scanf("%d",&count);
    printf("\n 请输入产品单价:");
    scanf("%f",&price);
    if (count<=50) total=count*price;
    else total=count*price*0.8;
    printf("产品总价为:%7.2f",total) ;
}
```

(4)编写好程序后，可以在 C 语言集成开发环境中对程序进行编译、链接、执行，检测程序是否实现了所要求的功能。如果程序有错误，可对程序进行调试，修改错误，直到能够正确实现所要求功能。

1.3　算　　法

1976 年，N.Wirth 出版的名为 *Algorithms + Data Structure = Programs* 的著作中，明确提出“数据结构”和“算法”是程序的两个要素，即程序设计主要包括两方面的内容：结构特性设计和行为特性设计。结构特性设计是指在问题求解的过程中，计算机所处理的数据及数据之间联系的表示方法。行为特性设计是指完整地描述问题求解的全过程，并精确地定义每个解题步骤的过程，也称为算法设计。

算法就是指求解问题的具体步骤与方法，可以理解为由基本运算及规定的运算顺序所构成的完整的解题步骤，它是对解题方案的准确而完整的描述。例如，在例 1.1 中，为了计算产品总价，我们分析了第一步应该干什么、第二步应该干什么，这就是算法。

1.3.1　算法的描述

算法的描述方法有多种，常用的有自然语言、流程图、N-S 图、伪代码、程序设计语言等。下面依次进行说明。

1. 自然语言表示法

自然语言是指人们日常生活中使用的语言，如汉语、英语等。可以使用自然语言来描述算法，例 1.1 中就使用了自然语言来表达如何根据产品数量及单价来计算产品总价。下面再举一个例子进行说明。

【例 1.2】　计算 1+2+3+…+50 的值。

解决该问题的算法有多种，下面介绍其中一种。设置一个变量 sum 来表示累加和，再设置一个变量 i 来表示加数。

第一步：置 sum 为 0，置 i 为 1。

第二步：把 sum 的值增加 i。

第三步：把 i 的值增加 1。

第四步：判断 i 是否小于等于 50，如果是，则转到第二步，重复执行；如果不是，接着执行第五步。

第五步：输出 sum 的值。

使用自然语言来描述算法比较简单、易懂，但容易出现二义性。例如，“小明说小强把他的作业落在家里了”，这句话中“他的”是指小明的还是小强的，就无法判断。此外，对于一些逻辑比较复杂的算法，用自然语言不能够清楚地表达算法过程。因此，一般情况下，自然语言表示法适用于描述一些简单问题的算法。

2. 流程图表示法

流程图是对过程、算法、流程的一种图形化表达方法，它由一组标准的、约定好的符号组成。图 1-1 给出了常用的一些流程图符号。

图 1-1 中的圆角矩形表示流程的开始或结束。矩形框表示某个流程或处理。菱形框表

示进行条件判断，它通常有一个入口和两个出口，其中包括一个真出口（条件为真时的转向）和一个假出口（条件为假时的转向）。流程线代表流程的进展方向。平行四边形表示输入或输出的数据。圆圈表示连接点，当一个流程图很大，一页画不完需要转到下一页时，可以用连接点将位于不同页上的流程图连接起来。

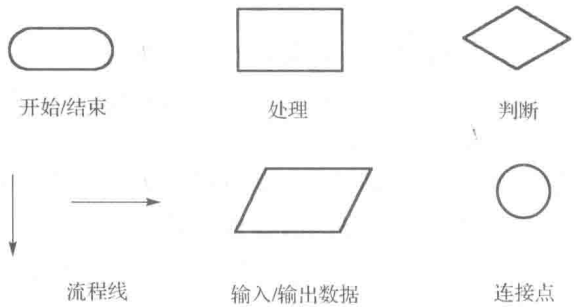


图 1-1　流程图常用符号

【例 1.3】　将例 1.1 中的算法用流程图描述，结果如图 1-2 所示。

【例 1.4】　将例 1.2 中的算法用流程图描述，结果如图 1-3 所示。

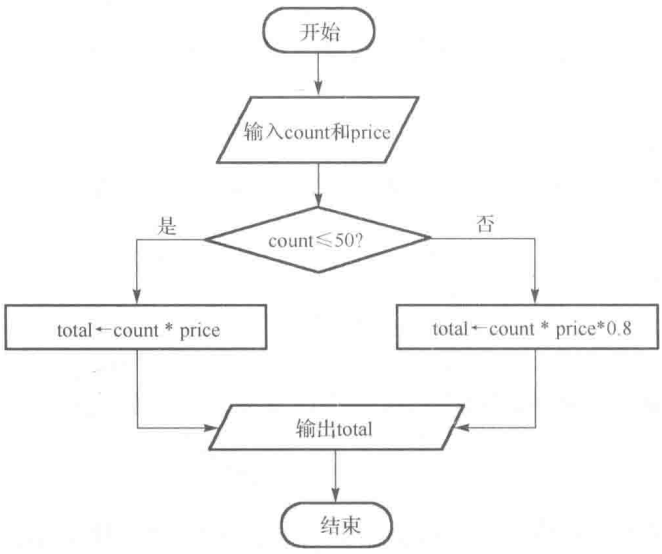


图 1-2　计算产品总价的流程图

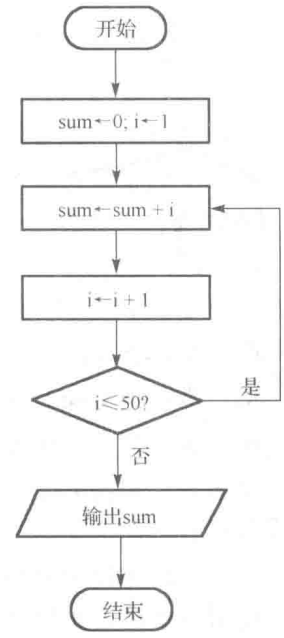


图 1-3　描述例 1.2 算法的流程图

3. N-S 图表示法

1973 年，美国学者 I.Nassi 和 B.Shneiderman 提出了一种新的流程图形式，在这种流程图中去掉了带箭头的流程线，将全部过程写在一个矩形框内，该矩形框内还可以嵌套矩形框，这种流程图称为 N-S 图。

【例 1.5】　将例 1.1 中的算法用 N-S 图描述，结果如图 1-4 所示。

【例 1.6】　将例 1.2 中的算法用 N-S 图描述，结果如图 1-5 所示。

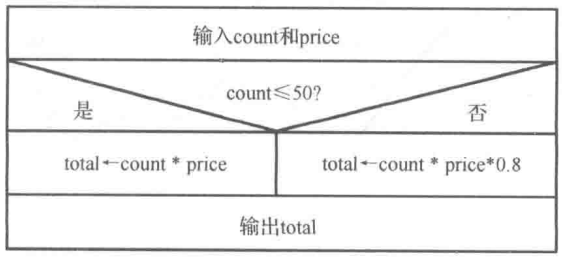


图 1-4　计算产品总价的 N-S 图

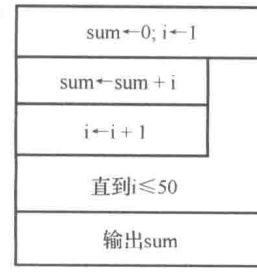


图 1-5　描述例 1.2 算法的 N-S 图

4. 伪代码表示法

伪代码(Pseudo Code)，又称为虚拟代码，是高层次描述算法的一种方法。它是用介于自然语言和计算机语言之间的文字和符号(包括数学符号)来描述算法。使用伪代码的目的是让算法可以容易地被任何一种编程语言(Pascal、C、Java 等)实现。使用伪代码不拘泥于具体实现，格式比较自由，用户可以用自己所熟悉的任何一种语言，只要把程序的意思表达出来就可以。

【例 1.7】　将例 1.1 中的算法用伪代码描述如下。

```
begin
    输入 count 和 price;
    if count<=50 then total=count*price;
    else total=count*price*0.8;
    输出 total;
end
```

【例 1.8】　将例 1.2 中的算法用伪代码描述如下。

```
begin
    sum←0;
    i←1;
    do
    begin
        sum←sum+i;
        i←i+1;
    end
    while(i<=50);
    输出 sum;
end
```

5. 程序设计语言表示法

前面所述的四种算法描述方法可以表达出算法的设计思想，但要实现算法，即让计算机去处理、理解并执行算法，需要使用某种程序设计语言来描述算法。因此，程序设计语言表示法描述的算法是能够被计算机处理与执行的。程序设计语言表示法要求描述算法时必须遵守所使用的程序设计语言的语法规则与要求。

【例 1.9】　将例 1.2 中的算法用 C 语言描述，程序设计如下。

```
#include<stdio.h>
void main()
{
    int sum=0, i=1;
    do
        {
            sum=sum+i;
            i=i+1;
        }
    while(i<=50);
    printf("1 至 50 的和为:%d", sum);
}
```

1.3.2　算法的特征

一个算法应该具有下列特征。

1. 有穷性(Finiteness)

算法的有穷性是指算法必须能在执行有限个步骤后终止。

2. 确切性(Definiteness)

算法中的每一个步骤都应该十分明确，不应当具有二义性。

3. 输入项(Input)

所谓输入是指在执行算法时需要从外界取得的必要信息。一个算法应该有零个或多个输入，以刻画运算对象的初始情况，所谓零个输入是指算法本身定义了初始条件。

4. 输出项(Output)

一个算法应该有一个或多个输出，以反映对输入数据加工后的结果。算法的目的是求解，没有输出的算法是没有意义的。

5. 有效性(Effectiveness)

算法中的任何计算步骤都可以被分解为基本的、可执行的操作步骤，即每个计算步骤都可以在有限时间内完成。

1.3.3　算法的评价

针对同一个问题，通常有不同的解决方法和步骤，也就是可采用不同的算法，因此需要从多种算法中选择一个相对较优的。通常，我们可以从两方面来对一个算法的质量进行评价，即时间复杂度和空间复杂度。

1. 时间复杂度

时间复杂度定义为执行算法所需要的计算工作量，是对算法执行速度的一个度量。假设用 n 来表示问题的规模，对于排序算法来说，要进行排序的数字个数为 n，由于一个算法花费

的时间与算法中语句的执行次数成正比，哪个算法中语句执行次数多，它花费的时间就多，因此，算法中基本操作重复执行的次数通常是问题规模 n 的某个函数，用 T(n) 来表示，也称为语句频度或时间频度。若有某个辅助函数 f(n)，使得当 n 趋近于无穷大时，T(n)/f(n) 的极限值为不等于零的常数，则称 f(n) 是 T(n) 的同数量级函数。记为 T(n)=O(f(n))，称 O(f(n)) 为算法的渐进时间复杂度，简称时间复杂度。若算法中语句执行次数为一个常数，则算法的时间复杂度为 O(1)。若算法的时间频度 $T(n)=n^2+2n+1$，则算法的时间复杂度为 $O(n^2)$。

2. 空间复杂度

空间复杂度是对一个算法在运行过程中所需要的存储空间大小的度量。一个算法的空间复杂度定义为该算法执行时所耗费的计算机存储空间，它也是问题规模 n 的函数，记为 S(n)=O(f(n))。一个算法运行时所占用的存储空间主要包括三个方面，即输入/输出数据占用的存储空间、存储算法本身所占用的存储空间和算法运行时产生的临时数据占用的存储空间。其中，输入/输出数据占用的空间大小不会随着算法不同而不同，而存储算法所需的空间与算法书写长度相关，临时数据占用的空间也会随算法不同而不同，因此，算法质量的优劣对其所占据的存储空间大小有着密切影响。

1.4　结构化程序设计

结构化程序设计的概念最早由 E.W.Dijikstra 于 1965 年提出，它是软件发展的一个里程碑。结构化程序设计强调程序设计风格与程序构造的规范化，其基本思想如下。

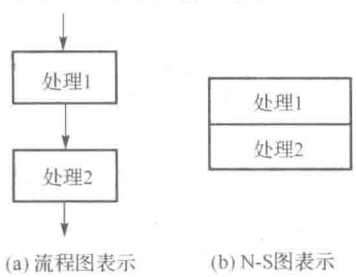
(1) 自顶向下，逐步求精，模块化：将一个复杂任务按照功能进行拆分，并逐层细化到便于理解和描述的程度，最终形成由若干独立模块组成的树状层次结构。

(2) 使用三种基本控制结构构造程序：任何程序都可由顺序、选择、循环三种基本控制结构构造。

下面对三种基本控制结构进行详细说明。

1.4.1　顺序结构

顺序结构表示程序中的各操作是按照它们出现的先后顺序执行的，其流程如图 1-6 所示。按照书写顺序，处理 2 将在处理 1 执行完后才执行。



(a) 流程图表示　　(b) N-S图表示

图 1-6　顺序结构

1.4.2　选择结构

选择结构又称为分支结构，它表示程序的处理步骤出现了分支，需要根据某一特定的