C 语言程序设计基础

主　编　鲍广华

副主编　钦明皖

编　著　胡　勇　王　虎　王文兵　郑珺露　张　磊

安徽大学出版社

内 容 简 介

本教材是根据教育部高等学校非计算机专业计算机基础课程教学指导分委员会《关于进一步加强高校计算机基础教学的意见》中有关计算机程序设计课程要求，并兼顾计算机等级考试需要而编写，其内容涵盖了二级 C 语言程序设计（考试）大纲的要求。全面系统地介绍了 C 语言的语法知识以及程序设计的基本思想、基本方法与基本技能，内容丰富、深入浅出，便于理解。尤其是对编程题的分析与阐释，有利于培养读者的实际编程能力。

本教材可作为高等院校非计算机专业理工科学生的教材，也可作为计算机等级（水平）考试的考生参考教材，还可以作为各类成人教育计算机程序设计课程的教材。同时也可作为计算机程序设计自学者的参考书。

图书在版编目(CIP)数据

C 语言程序设计基础/鲍广华主编．—合肥：安徽大学出版社，2008

ISBN 978－7－81110－326－7

Ⅰ．C… Ⅱ．鲍… Ⅲ．C 语言—程序设计—水平考试—教材

Ⅳ．TP312

中国版本图书馆 CIP 数据核字(2008)第 017465 号

C 语言程序设计基础

鲍广华 主 编 钦明皖 副主编

出版发行	**安徽大学出版社**	**经 销**	**新华书店**
	(合肥市肥西路 3 号 邮编 230039)	**印 刷**	**合肥中德印刷培训中心印刷厂**
联系电话	**发行部** 0551－5107784	**开 本**	787×1092 1/16
E-mail	ahdxchps@mail.hf.ah.cn	**印 张**	17.125
网 址	www.ahupress.com.cn	**字 数**	417 千
责任编辑	**李镜平**	**版 次**	2008 年 2 月第 1 版
特约编辑	**罗 罹 罗季重**	**印 次**	2008 年 2 月第 1 次印刷
封面设计	**孟献辉**		

ISBN 978－7－81110－326－7 **定价** 26.00 **元**

前　言

“计算机程序设计”是高等学校计算机基础教学中的核心课程。众多计算机程序设计语言中，C 语言是目前国内外最广泛使用的程序设计语言之一。C 语言不仅具有高级语言完备的特性，而且具有汇编语言的功能，可以直接实现对系统硬件及外部设备接口的控制，具有较强的系统处理能力。正是由于 C 语言程序的简洁高效、清晰的模块化结构、良好的可移植性等特点，使得 C 语言成为诸多高校计算机程序设计语言的首选。

根据教育部高等学校非计算机专业计算机基础课程教学指导分委员会《关于进一步加强高校计算机基础教学的意见》中有关计算机程序设计课程的要求，兼顾计算机等级考试需要编写了本书，其内容涵盖了二级 C 语言程序设计（考试）大纲的要求。可作为高等院校非计算机专业理工科学生的教材，也可作为计算机程序设计自学者、计算机等级（水平）考试的考生参考教材，还可以作为各类成人教育计算机程序设计课程的教材。

本书以 C 语言程序设计为基础，注重讲解程序设计的概念、方法和思路，培养读者的基本编程能力、逻辑思维和抽象能力。主要内容包括算法、基本数据类型、程序控制结构、复合数据类型、指针、结构化程序设计的基本概念与方法、函数与模块设计等。

编者希望读者通过对本书的学习，初步掌握程序设计的基本概念与基础知识、以及程序设计的基本思想和基本方法；了解从问题分析、算法设计、编制程序、调试程序到算法分析的基本过程，了解在程序开发时起重要作用的思想与技术，掌握一些典型问题的算法；能用 C 语言进行一些实际程序的设计与编写，并尝试通过编程解决一些示例性的应用问题，以便将来更好地利用编程技术解决读者自身专业领域中的实际问题。

教学的宗旨是培养读者实际动手编程能力。在编写过程中，自始至终注重培养读者的编程能力；注重例题与习题的实用性，以激发读者的学习兴趣；例题由浅入深、层层深入，以培养读者深入全面地观察、分析与解决问题的能力；编程题中程序代码前给出分析、程序代码中给出注释，使读者的学习更容易；书中舍去词法语法中的繁琐细节，以突出重点，有利读者的掌握。

建议读者，学习过程中强化实践，重视上机调试；提倡“自立、自信、自尊、自强”，知难而进，尽可能不向他人求助，独立完成；做到：多看、多想、多写、多上机。

本书的第 1、9 章由鲍广华编写，第 2、3 章由郑珺露编写，第 4 章及附录由张磊编写，第 5 章由钦明皖编写，第 6、10、11 章由胡勇编写，第 7、8 章由王文兵编写，第 12、13 章由王虎编写。本书由鲍广华、钦明皖策划与审定，全书由鲍广华统稿，王虎参加了本书的校对工作。

由于编写时间仓促，加之编者水平有限，书中缺点、错误在所难免，恳请读者批评指正。

鲍广华

2008 年 1 月

目 录

第 1 章　算法与程序

学习目标

了解程序、程序设计的概念
了解算法的概念、特征
用伪代码法、流程图法、N－S 图法表示算法
了解程序的基本结构
了解程序设计的一般步骤

重、难点

重点：算法、程序、程序设计的概念，算法的表示
难点：算法的 N－S 图法，数据结构的概念

1.1　算　法

1.1.1　算法的概念

为什么要介绍算法？算法是什么？要搞清这些问题，先要搞清何谓程序设计以及进行程序设计的目的。其实，程序设计是高等学校计算机基础教学中的一门核心课程。简单地说，程序设计就是对要解决的问题进行分析，而后给出解决的方案，也就是解决问题的方法与步骤，并用计算机能够识别的语言（称为程序设计语言）来表示这个方案，这个方案就称为程序。这样做的目的就是要在计算机上执行这个方案，也就是让计算机按照用户设计好的方法与步骤一步步地执行，从而为用户解决实际问题。换言之，进行程序设计的最终目的就是通过让计算机执行用户编写好的程序，从而为人类解决一些实际问题。用程序设计语言为计算机写出解决问题的方法与步骤的过程称为编程。

一般地，用户为解决某个问题，并不是一开始就用计算机语言来描述这个方法与步骤，而是用自己所熟悉的方法来描述。例如，可用人类的自然语言来描述解决问题的方法和步骤，也可用一目了然的传统流程图形式来描述，或采用其他任何一种描述方法，只要能清晰而明确地表达出解决问题的方法与步骤，而且这种描述没有丝毫的含糊其辞与模棱两可之处即可。比如说，“付 10 元钱给你”很明确，但“付几元钱给你”就不明确，这里说的几元，计

算机不知道到底是多少元，2元、3元或者8元、9元都是几元，计算机无法判断到底要付几元，也就无法付钱给你。这种清晰而明确的解决问题的方法和步骤就是算法。“算法”(Algorithm)一词是从9世纪阿拉伯数学家al－khovarizmi的名字派生的，他发现了很多算术问题的算法，当然现在所说的算法并不仅局限于算术问题。用户所说的算法既可是数值运算算法，亦可是非数值运算算法。数值运算的目的是求数值解，例如求一个方程的根。非数值运算包括的面十分广泛，种类繁多，要求各异，很难规范化，故对非数值运算问题需要借鉴已有类似算法而重新设计出相应算法。总之，算法就是解决问题的方法和步骤，是程序设计的灵魂，这也正是开篇就介绍算法的缘由。

1.1.2 算法的特征

算法是用户为了让计算机做某件事而告诉它的具体的方法与步骤，这个方法应该是明确的、具体的，不可含糊的，步骤应该是有限的，就是说，算法应满足一定的条件，否则就不是一个正确的算法，或者根本就不能称其为算法。通常算法具有如下特征：

第一，有穷性。一个算法应在有限步内结束，而这种“有限”是指人们可接受的，而不是无限的或是人们不可接受的。若算法是无限的，就意味着计算机要永无止境地执行下去，又何以能给出最终执行结果，这种算法当然不能称之为“算法”。而即便算法是在有限步内结束，可如果这个“有限”对实际问题而言有点离谱，那么这种“有限”也就是“无限”。例如：计算某大学数学专业07级68位新生入校时总分的平均值，并打印输出。通常让计算机解决该问题，最多只要几分钟的时间，而如果你的算法要计算机运行1天才能给出结果，那么即使最终能给出正确的结果，也不能称其为算法。就如同有一批急救物资要在1h内从A地运到B地，假设用户设计的方案，要在10h后才能运到目的地，物资虽已运到，但已没有任何实际意义，这种所谓的方案也就不能称之为方案。

第二，确定性。一个算法中的每一步都必须有清晰的、明确的、唯一的含义，而不能是含糊其辞、模棱两可的，即不能存在二义性。这点上面已有叙述。

第三，有一个或多个输出。算法的目的，是向计算机描述解决问题的方法与步骤，让计算机按算法逐步执行，最后得出结果，并且要把这个结果告诉用户。否则结果虽然求出，但只是放在计算机内部，这和没做无区别。故算法要有输出，一个算法至少要有一个输出，当然根据问题的需要，一个算法也可有一个以上的结果输出。没有输出的算法没有任何意义。

第四，有效性。算法中的每一步都要能有效地被执行，并且要能够得到确定的结果，例如：算法中若含有“x＝1，y＝0，z＝x/y；”部分，那么因除数不能为零，z＝x/y就不能被有效执行。

第五，有或者无输入。一般说来，一个问题的解决，都会需要有一个或是一批原始数据的输入。例如要让计算机求出99和98的乘积，算法中就需要对99、98这两个数进行输入。当然也有些问题的解决，不需要输入任何数据。例如让计算机打印输出九九表，不需输入任何数据，即可求解。

另外，对算法还有几点说明：

(1) 并非只对“计算”方面的问题才有算法而言。任何一个问题，无论它是“计算”方面的，还是非计算方面的，要解决它，都得要有具体的方法和步骤，这种方法与步骤就是所谓的算法。算法可分两大类：一类是数值运算算法，一类是非数值运算算法。前者的目的是求数值解，例如求一个一元二次方程的根。目前对这类算法的研究相对而言比较深入，算法也较

成熟，对一般的数值运算问题都有较成熟的算法可供选用。而非数值运算包括的面很广，种类繁多，要求也各不相同，故对非数值运算问题，需参考已有类似算法，重新设计相应算法。例如人口普查、航空调度管理等。

（2）不同的人或同一个人，对同一个问题，都可能给出不同的算法。而这些不同的算法，对问题的解决，可能会得到相差无几的效果，也有可能得到差别很大的效果。算法不仅要正确，同时还要有好的质量，对存在多种算法的问题，要从中选择最佳的那个算法。例如：现在有一批新鲜的、大而甜的新品西瓜要从合肥运到北京。不同的人或同一个人都可有不同的解决方案，这些方案中有些效果可能相差无几，也可能某种方案其效果最佳，或其效果很差，用户应从中选出运费最低、费时最少、损耗最小的方案。

（3）并非是任何用户能描述出来的算法，计算机都一定能按用户的描述来实现。目前为止，有些问题即使能描述出来，甚至可能是很简单的描述，但计算机却做不到。例如，用户让计算机“帮我的妈妈捶捶背”，计算机就无法按用户的旨意去完成。这里讨论的算法，是针对那些计算机能够做到的而言。

1.1.3　算法的表示

由前已知，只要人们能看懂，无论采用什么样的方式来描述算法皆可，用语言、流程图或是其他方法都行。目前，常用以下几种方法来描述算法。

（1）用人类的自然语言描述算法。自然语言就是用户日常使用的语言，可以是汉语、英语或其他任何一种语言。这种表示方法的特点是通俗易懂，但自然语言本身所表示出的含义有时不严格，相对来说这种方法表示的算法容易出现“歧义性”。且对含有分支或是循环的算法，这种表示方法用起来也不太方便，所以用得不是很多，除非要表示的问题很简单。例如：

【例 1.1】　求 3＋13 的和，并输出。

分析：　为设计算法，先要分析问题，搞清题目要求后，想想怎样来解决问题，也就是给出解决的方法与步骤，经分析可得如下算法：

第 1 步：先输入 3 放在变量 x 中，13 放在变量 y 中；

第 2 步：求 x＋y 的和，并将和放在变量 z 中；

第 3 步：将 z 输出。

【例 1.2】　现有 100 块酥糖，要平均分给宿舍的 7 位同学，要求不要把糖弄碎，即每人得到糖的块数是整数，请判断这些糖能否刚好被分尽。

经分析，可设计算法如下：

第 1 步：100⇒y，7⇒x；

第 2 步：看变量 y 能否被变量 x 整除，若能，到第 3 步；否则，跳过第 3 步到第 4 步；

第 3 步：显示“能”，转到第 5 步；

第 4 步：显示“不能”；

第 5 步：结束。

算法可以比较粗略，也可以比较详细。一般来说，对较粗的算法，较易看懂其总的轮廓，但依据其编程时，有些细节必须要重新考虑处理的方法；而较细算法的特点和前者正相反，依据其编程较容易，但较细算法看起来不简洁，不利于算法总体轮廓的呈现，可能要读较长

时间才能明白。采用粗框还是细框，要根据问题本身的复杂程度以及自己的喜好而定。有些相对复杂的问题，起初只能给出较粗的轮廓，接下来再对这些粗的轮廓进行细化。若需要还可再进一步细化，如此反复，直到算法中的每一步都可被实现。例如，此例中开始分析问题时，并不知怎样来判断 100 是否能被 7 整除，但没关系，可假定知道，先给出个相对粗略的算法，接下来再去考虑细节问题。其实这就是结构化程序设计中所谓的“自顶向下，逐步求精”的原则。当然对此例，也可一开始就给出如下稍细一些的算法（请思考：细在哪里？）：

第 1 步：7⇒x，100⇒y；

第 2 步：y%x＝0 吗？若为 0，到第 3 步；否则，跳过第 3 步去第 4 步；

第 3 步：显示“能”，转到第 5 步；

第 4 步：显示“不能”；

第 5 步：结束。

（2）用传统流程图表示算法。传统流程图就是由一些带有文字的矩形框（亦称功能框）、菱形框（亦称判断框）、带有箭头的线段（亦称流向线）以及必要的文字或字符的标注而组成的图。这种图看起来直观、形象，图中的流向线用来指出各框的执行顺序。这种方法表示算法曾经很流行，直到现在，也还有很多人习惯用此法，特别是初学时，用这种方法描述算法相对好理解。但这种图所占篇幅相对较多，算法复杂时，画这种图既费时又不方便，而且由于这种方法中对流向线的使用没有严格限制，设计者可随意在图中转来转去，使得这种算法相对难读、难改。正因如此，结构化程序设计方法中，很多时候人们用 N－S 图来取代这种传统的流程图。图 1.1 是【例 1.2】的传统流程图表示的算法。

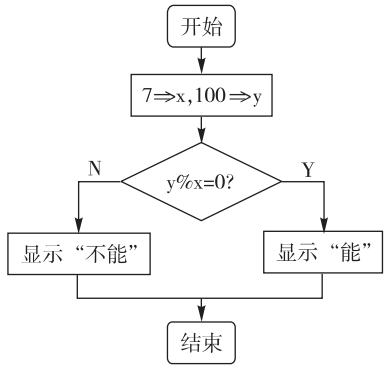

图 1.1 【例 1.2】的流程图

（3）用 N－S 图表示算法。为提高算法的质量，不能借助流向线无规律地改变程序的走向，因流程的走向代表着程序执行的顺序。但实际问题中，遇到分支（指程序执行到此，要根据具体条件的判断才能决定程序接下来的走向）和循环（指反复多次执行若干步操作）是常有的事。框图中会存在一些流程的向前或向后的非顺序跳转，这个问题必须面对。为此，人们设想，规定几个基本结构，再把它们按一定规律组成算法，如同搭积木一般，整个算法的结构由这些基本结构从上至下顺序排列而成。于是人们设计了顺序结构、选择结构（亦称分支结构）、循环结构三种基本结构。

顺序结构，如图 1.2 所示，虚线框内为一顺序结构（A 和 B 两框内部也是顺序结构）。这种结构中，语句被执行的顺序按语句本身在程序中出现的次序进行，当 A 框内所有步骤执行完以后，就会顺序执行 B 框，三种基本结构中，顺序结构是最简单的一种。

选择结构，指依据一定的条件把语句分成不同的分支，程序只执行其中一个分支，而不执行其他分支。选择结构又可细分为双路选择结构（图 1.3）、单路选择结构（图 1.4）、多路选择结构（图 1.5、图 1.6）。图 1.3 中，虚线框内是一选择结构，其中一定含有一个判断框，条件 C 的成立与否决定了是执行 A 框还是 B 框，根据条件若是执行 A 框，就不再执行 B 框，反之亦然。绝不会既执行 A 框，又执行 B 框，反之亦然。不过，有时两框之中，可有其一为空框，即所谓的单路选择结构，如图 1.4 所示，这时条件成立执行 A 框，然后接着向下；否则，什么也不做，直接向下。图 1.5 及图 1.6 则如同现实生活中的多岔路口。图 1.5 所示结构中，首先判定第一个菱形框中的条件 C_1，若为真，则执行 A_1 框后转去执行 A_{n+1} 框之后的部分，若条件 C_1 为假，则对第二个菱形框做类似处理，若条件 C_1、C_2、…、C_n 全为假，则执行 A_{n+1} 框后接着执行本结构的后续部分，即 A_{n+1} 框之后的部分。图 1.6 与图 1.5 类似，所不同的是，对图 1.6 而言，当条件 C_1、C_2、…、C_n 全为假时，则直接执行本结构的后续部分。

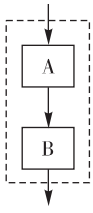

图 1.2　顺序结构

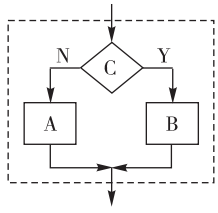

图 1.3　双路选择结构

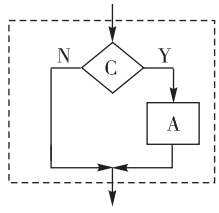

图 1.4　单路选择结构

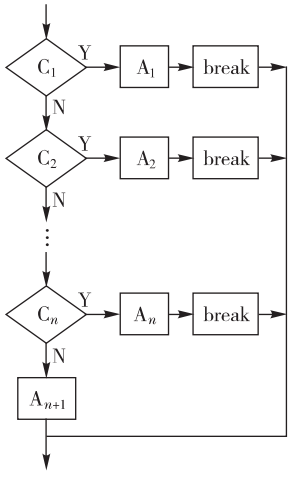

图 1.5　多路选择结构（一）

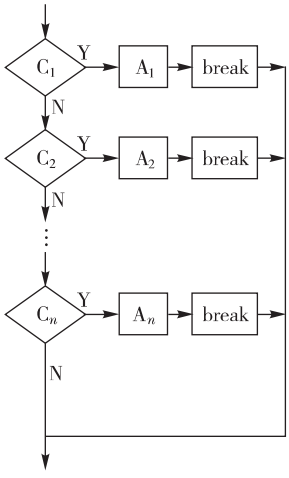

图 1.6　多路选择结构（二）

循环结构，亦称重复结构。该结构使得某些操作在一定条件下可反复多次被执行，这些需反复多次被执行的操作称之为循环体，这个条件称为循环条件。重复执行的次数可根据需要事先指定，也可事先不指定，而由循环体中语句的变化来决定。如图 1.7、图 1.8 和图 1.9 所示虚框内皆为循环结构。对图 1.7 所示结构，进入循环后，首先进行循环条件 C 的判断。若条件成立，执行框 A（框 A 内可含有若干步操作，称框 A 为“循环体”），框 A 执行完毕，再转回到循环条件的判断，只要条件成立就执行框 A，框 A 执行完转至框 C 判断，直至框 C 条件不成立，退出循环，接着执行算法的后面部分（图中将从框 B 开始接着向下执行）。图 1.8 中，进入循环后，首先执行框 A（同样框 A 内可含有若干步操作），框 A（即循环体）执行完毕，才进行循环条件的判断，若条件成立再执行框 A，框 A 执行完又到框 C 判断，直到

框 C 条件不成立，退出循环，接着执行算法的后续部分（图中将从框 B 开始接着向下执行）。

图 1.9 的执行过程与图 1.8 类似，不同的是，每次当条件 C 不成立时执行框 A，直至框 C 条件成立，退出循环，接着执行算法的后续部分（图中将从框 B 开始接着向下执行）。

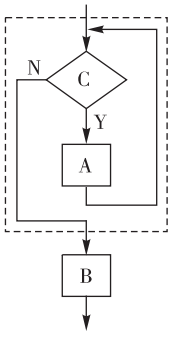

图 1.7　循环结构（一）

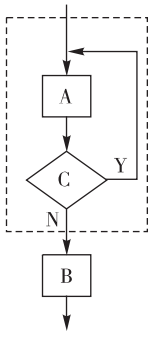

图 1.8　循环结构（二）

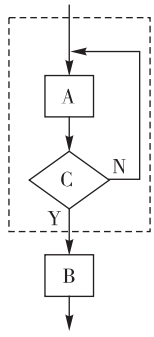

图 1.9　循环结构（三）

三种循环结构的相同之处：都要进行循环条件的判断，循环体都可能被反复多次执行。不同之处：对图 1.7 而言，因进入循环后首先要对循环条件进行判断，即所谓的先判断，后执行。若第一次判断，条件 C 就不成立，循环体就一次也不会被执行。而对图 1.8 及图 1.9 而言，因进入循环后首先要执行循环体，而后才进行条件 C 的判断，即所谓的“先执行，后判断”，故这种结构中，循环体至少要被执行一次。其实图 1.2 和图 1.3 中的框 A、框 B，图 1.4 中的框 A，图 1.5 中的框 A_1、A_2、…、A_n、A_{n+1}，图 1.6 中的框 A_1、A_2、…、A_n，以及图 1.7～1.9中的框 A、框 B，既可是若干基本操作，亦可是分支结构或是循环结构。若是后者，情况会相对复杂些，这些内容将在后续章节中逐一介绍。

由上述三种基本结构顺序组成的算法，可解决任何复杂问题。而且由它们组成的算法属于“结构化”的算法。所谓“结构化”也即是说这种结构是框架式的，是由一个个的块块所组成。在这样构成的“结构化”算法中，不存在无规律的转来转去，但在这种算法的基本结构内可以存在分支或是存在向前、向后的跳转。另外，基本结构也不是只能有上述三种，用户也可规定一个基本结构，而且用它们组成的算法也同样是“结构化”的算法，不过用户规定的这种结构应具有以下特点：

① 只有一个入口；

② 只有一个出口；

③ 结构内的每步操作都有机会被执行；

④ 结构内不能有死循环（所谓死循环就是永远无法退出的循环）。

既然用上述三种基本结构的顺序组合可给出任何一个复杂问题的算法，那么基本结构间的流向线就可不要了。由两位美国学者 I. Nassi 和 B. Shneiderman 提出的一种新的流程图中，已完全不见流向线，这种流程图称 N－S 结构化流程图，也简称 N－S 流程图或 N－S 图。在 N－S 图中，全部算法都写在一个矩形框内，若需要，此框内还可再含有另外一个（亦可是若干个）框，这种流程图适用于结构化程序设计，很受欢迎。

N－S 图中，顺序结构、分支结构、循环结构这三种基本结构可分别表示为图 1.10～1.15，它们的执行过程分别对应图 1.2～1.9 的执行过程，由这些基本结构可组成复杂的 N－S图，亦即使用这三种简单的基本结构可表示出任何复杂问题的算法。