21世纪CENTURY 高等学校规划教材

C语言程序设计

主编　田祥宏
主审　荣　政

西安电子科技大学出版社
http://www.xduph.com

面向21世纪高等学校规划教材

C语言程序设计

主　编　田祥宏

副主编　沈　奇　王旭辉　吕艳琳

主　审　荣　政

西安电子科技大学出版社

2007

内 容 简 介

本书是作者在多年教学实践经验的基础上编写而成的，以程序设计的需要带动语言知识的学习，系统介绍了C语言及其程序设计技术。全书共12章，内容包括程序设计基础、数据及数据类型、数据运算、选择结构、循环结构、数组和字符串、指针、函数、构造数据类型、文件、图形设计和实验指导，并通过将知识点融入实例，以实例带动知识点的掌握，详尽介绍了相应的算法知识。每章都编排了大量的习题，题型丰富，由浅入深，以帮助读者在掌握C语言的基础上，着重培养读程序与写程序的能力。本书最后一章(实验指导)供学生进行上机练习。

本书可作为普通高校应用型本科学生或工科本/专科学生学习 C 语言程序设计的教材或参考书，特别适合作为计算机等级考试(二级C语言)的教材或参考书，也可作为有关程序设计人员和自学者的参考书。

★ 本书配有电子教案，有需要的老师可与出版社联系，免费提供。

图书在版编目(CIP)数据

C语言程序设计 / 田祥宏主编. —西安：西安电子科技大学出版社，2007.12

面向21世纪高等学校规划教材

ISBN 978-7-5606-1935-4

Ⅰ.C… Ⅱ.田… 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(2007)第161539号

策　　划　马乐惠

责任编辑　张　玮　马乐惠

出版发行　西安电子科技大学出版社(西安市太白南路2号)

电　　话　(029)88242885　88201467　邮　编　710071

http://www.xduph.com　E-mail: xdupfxb@pub.xaonline.com

经　　销　新华书店

印刷单位　陕西华沐印刷科技有限责任公司

版　　次　2007年12月第1版　2007年12月第1次印刷

开　　本　787毫米×1092毫米　1/16　印　张　21.5

字　　数　510千字

印　　数　1～4000册

定　　价　29.00元

ISBN 978-7-5606-1935-4/TP·1003

XDUP 2227001-1

*** * * 如有印装问题可调换 * * ***

本社图书封面为激光防伪覆膜，谨防盗版。

前　言

C 语言由于其卓越的优点，而在计算机的各个领域内得到了广泛的应用，从系统软件的编写到应用程序的设计，特别是在图形处理和底层应用方面目前应用广泛。此外，C 语言是一门结构化程序设计语言，有利于学生掌握程序设计的思想，因此，C 语言已成为目前高校学生掌握程序设计的一门基础性语言。

本书以程序设计思想的掌握为主线，由浅入深，先讲述基本知识及例题，再讲述应用，重在训练学生的编程思想，提高学生应用 C 语言的能力。本书的编写结合了多年来应用型本科人才培养的经验，重点体现了应用型本科人才培养的要求。

本书共分 12 章。第 1 章介绍了程序设计的基础知识，第 2～11 章系统介绍了 C 语言的基础知识及用 C 语言设计程序、解决问题的方法，包括 C 语言的基本语句、结构和函数以及一些算法的实现。各章配有大量的例题解析和一定数量的习题，以便让读者通过对例题的学习和习题的练习进一步理解和掌握 C 语言。这部分用于课堂讲授，建议用 56～64 学时讲授较为合适。第 12 章是实验指导，用于学生上机练习。该部分分为 7 个实验，通过这些实验，使学生更加深刻地掌握 C 语言的基础知识和程序设计方法。学生上机实验时间建议在 16～24 学时较为合适。

本书可作为普通高校应用型本科或工科本/专科学生学习C语言程序设计的教材或参考书，特别适合作为计算机等级考试(二级 C 语言)的教材或参考书，还可作为有关程序设计人员和自学者的参考书。本书打*标记部分对于专科生只作为了解的内容。

参与本书编写工作的有：田祥宏(第 1、5、8、10、11、12 章)，沈奇(第 3、6 章)，王旭辉(第 2、7 章)，吕艳琳(第 4、9 章)，全书由田祥宏统稿。

在本书的编写过程中，得到了金陵科技学院计算机系许多老师的关心和支持，作者在此深表谢意。由于编者水平有限，书中的不当和疏漏之处在所难免，恳请使用本书的老师和同学提出宝贵意见：avatian@sohu.com。

本书相关资源网站：http://it.jit.edu.cn/kc/t/index.asp

http://kczx.jit.edu.cn/Netcourse/c36/Asp/Root/index.asp?moduleId=1

作　者

2007 年 8 月

目　录

试读结束，需要全本PDF请购买 www.ertongbook.com

第 1 章　程序设计基础

本章学习要求：

1. 了解程序设计的语言分类，掌握算法的概念、算法描述方法、流程图的三种基本结构、结构化流程图的画法以及结构化程序设计的方法。

2. 了解 C 语言的发展及应用现状，认识 C 程序的结构，掌握 C 语言的特点及其编译，熟练运用 C 程序的上机步骤。

人们要利用计算机解决实际问题，比如数值计算问题就需要人与计算机之间的交流，这种交流能使计算机按照人的意愿完成一系列的操作，实现这种交流的一种桥梁就是计算机语言，又称为程序设计语言。由程序设计语言编写，用来使计算机完成一定操作任务的“文章”称为程序，编写程序的工作称为程序设计。

随着计算机技术的迅速发展，程序设计语言经历了由低级到高级的多个发展阶段，程序设计方法也得到了不断的发展和提高。

1.1　程序设计语言

程序设计语言是人们根据计算机的特点以及描述问题的需要设计出来的。随着计算机技术的发展，不同风格的语言不断出现，逐步形成了计算机语言体系。

计算机语言按其发展过程一般分为机器语言、汇编语言和高级语言。

1.1.1　机器语言

机器语言是最底层的计算机语言。用机器语言编写的程序，计算机硬件可以直接识别。在用机器语言编写的程序中，每一条机器指令都是二进制形式的指令代码，即由一连串的二进制 0 和 1 组合起来的编码，每一条指令规定了计算机要完成的某个操作。在指令代码中，一般包括操作码和地址码，其中操作码告诉计算机进行何种操作，即“干什么”；地址码则指出被操作的对象存放在哪里。

机器语言程序都是由二进制 0 和 1 组成的系列，程序编写起来非常繁琐，可以用“难学、难记、难写、难检查、难调试”来概括，尤其是用机器语言编写的程序完全依赖于机器，程序的可移植性差。但是，由于用机器语言编写的程序直接针对计算机硬件，因此它的执行效率比较高，能充分发挥出计算机的速度性能，这也是机器语言的优点。

1.1.2 汇编语言

为了克服机器语言的缺点，人们对机器语言进行了改进，用一些容易记忆和辨别的有意义的符号代替机器指令，如：用指令助记符来代替机器语言指令代码中的操作码，用地址符号来代替地址码。用这样一些符号代替机器指令所产生的语言就称为汇编语言，也称为符号语言。

例如：为了计算表达式“9 + 8”的值，用汇编语言编写的程序与用机器语言(8086CPU的指令系统)编写的程序如表 1.1 所示。

表 1.1　汇编语言程序与机器语言程序举例

语句序号	汇编语言指令	机器指令	指 令 功 能
1	MOV AL,9	10110000　00001001	把加数 9 送到累加器 AL 中
2	ADD AL,8	00000100　00001000	把累加器 AL 中的内容与另一个数相加，结果存在累加器 AL 中(即完成 9+8 运算)
3	HLT	11110100	停止操作

从表 1.1 中可以看出，在该汇编语言程序中，以 MOV(MOVE 的缩写)代表“数据传送”，ADD 代表“加”，HLT(HALT 的缩写)代表“停止”等。这些符号含义明确，容易记忆，因而又称为助记符。由这些助记符编写的程序，可读性好，容易查错，修改方便；但计算机硬件不能直接识别，必须由一种专门的翻译程序将汇编语言程序翻译成机器语言程序后，计算机才能识别并执行。这种翻译的过程称为“汇编”，负责翻译的程序称为汇编程序，翻译出的程序称为目标程序，而翻译前的程序称为源程序。汇编程序的作用如图 1.1 所示。

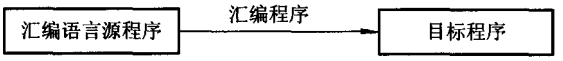


图 1.1　汇编程序的作用

汇编语言也是一种面向机器的语言，但比机器语言易读、易改，执行速度与机器语言相当，比高级语言快很多，因此直到现在仍在实时控制、实时处理领域中广泛应用。

1.1.3 高级语言

机器语言和汇编语言都是面向机器的语言，一般称为低级语言。低级语言对机器的依赖性很大，而且与自然语言相距甚远，不符合人们的表达习惯。

为了从根本上改变语言体系，必须从两方面入手：一方面是力求接近于自然语言；另一方面是力求脱离具体机器，使语言与指令系统无关，达到程序通用的目的，这就要求程序的可移植性好，一个可移植性好的程序，应用在各种计算机和操作环境中都能运行，并得到同样的结果。因此，从 20 世纪 50 年代中期开始逐步发展出面向问题的程序设计语言，称为高级语言。高级语言与具体的计算机硬件无关，其表达方式接近于被描述的问题，易为人们接受和调试，使编程效率得到大幅度的提高。高级语言的显著特点是独立于具体的计算机硬件，通用性和可移植性好。例如前面计算“9 + 8”的问题，如果使用 BASIC、C

语言或 PASCAL 语言来编程，就变得十分简单，而且易于理解，如表 1.2 所示。

表 1.2　高级语言程序举例

BASIC 语言程序	PASCAL 语言程序	C 语言程序

100　s=9+8	Program　addit
	var　s:integer;
	begin
	　s:=9+8;
	end
		main()
		{ int s;
		　s=9+8;
		}

在使用高级语言设计程序时，可以有两种设计方法：一种是面向过程的程序设计方法，另一种是面向对象的程序设计方法。例如：早期出现的 BASIC、PASCAL、FORTRAN、COBOL、C 等高级语言，采用的是面向过程的程序设计方法；而较新的 Visual Basic、Visual C++、Visual Foxpro、Delphi、Java 等采用的是面向对象的程序设计方法。

必须指出，用任何一种高级语言编写的程序(源程序)，都要通过编译程序翻译成机器语言程序(称为目标程序)后计算机才能执行，如 C 语言程序；或者通过解释程序边解释边执行，如 BASIC 语言程序。

从程序设计语言的发展过程和上述的例子可以看出，程序设计语言越低级，就越靠近计算机硬件，其描述的程序就越复杂，其中的每一条指令(或语句)也就越难懂。反之，程序设计语言越高级，就越靠近人的表达与思维方式，其描述的程序就越简单，其中的每一条语句也就越容易理解，也越接近人的自然语言。

1.2　程 序 设 计

程序是由程序设计语言编写的，用于完成特定的任务。计算机之所以能够自动地、有条不紊地工作，正是因为计算机能够按照程序所规定的步骤一步一步地执行相应的操作。什么是程序设计？对于初学计算机的人来说，往往简单地把它理解为编制一个程序。其实不然，至少是不全面的。实际上，程序设计包括多方面的内容，而具体编制程序只是其中的一个方面。著名的计算机科学家沃思(Nikiklaus Wirth)提出：

程序 = 数据结构 + 算法

实际上随着程序设计技术的发展，一个程序除了有以上两个主要要素之外，还应涉及到程序设计方法和语言工具等。因此，可以这样表示：

程序 = 算法 + 数据结构 + 程序设计方法 + 语言工具和环境

也就是说，以上四个方面是一个程序设计人员所应具备的知识。在设计一个程序时要综合运用这几方面的知识。在这四个方面中，算法是灵魂，数据结构是加工对象，语言是工具，编程需要采用合适的方法。

1.2.1　算法

算法是解决“做什么”和“怎么做”的问题。算法体现在程序中就是程序操作语句。

1. 算法及其特性

1) 算法

所谓算法，就是解决某类问题的方法。确切地说，就是对某一类特定的问题，给出解决该问题的一系列(有穷的)操作，而每一操作都有其确定的意义，并在有限时间内可以计算出结果。一个算法有多个输入量，它是问题给出的初始数据，经过算法的实现，它有一个或多个输出量，这就是算法对输入运算的结果，即问题的解答。

本书中所关心的只限于计算机算法，即计算机能执行的算法。例如：让计算机计算1+2+3+4，或将100个学生的C语言成绩按高低分排序，这些计算机可以做到，而让计算机去执行诸如“买一份报纸”或“炒青菜”，则是做不到的(至少目前还做不到，除非以后人工智能达到一定程度才行)。

计算机算法可分为两大类：数值算法和非数值算法。数值算法是为了解决求数值解的问题，例如求方程的根、求方程组的解和求定积分等，数值算法的研究将在“数值计算方法”(或称“计算方法”)这门课中解决。非数值算法能解决的问题非常广泛，最常见的是用于事务管理领域，如财务管理、人事管理、行车调度管理等，一部分非数值算法(如排序、查找等)将在“数据结构”这门课中得到体现。

2) 算法的特点

算法具有如下特点：

(1) 确定性。算法中的每一个步骤都应当是确定的，而不应当是含糊的、模棱两可的。也就是说，每一个操作步骤都有确切的含义，无二义性。

(2) 有穷性。一个算法必须在有限步骤后结束，要么计算出结果，要么非正常结束(说明算法有问题)。总之，算法的步骤不能是无限的。

(3) 可执行性。算法中的每一个步骤对计算机来说都应当是可以有效执行的，并得到确定的结果。

(4) 输入。输入是执行算法时需要从外界取得必要的信息。一个算法可以没有输入，也可以有一个或多个输入。

(5) 输出。算法是用来解问题的，使用算法就是要得到问题的“解”，即输出。因此，一个算法必须得到结果，也就是算法的输出。一个算法必须有一个或多个输出，如果没有输出，算法就没有意义，等于问题没有得到解决。

3) 算法的评价

如何评价一个算法呢？对算法的要求有：必须是正确的，必须能高效率执行，即占用内存空间少，所需运行时间短。于是，对算法的评价可以从两个方面进行：① 执行算法所需的时间长短；② 执行算法所需的计算机内存容量大小。

要想确切评价一个算法，必须采用算法分析方法。算法分析的标准是算法的时间复杂度和空间复杂度。这些概念可以参阅数据结构相关资料，这里不再一一赘述。

2. 算法的描述

一个算法设计好后，可以采用不同的表示形式，以便交流和阅读。常用描述算法的方法有自然语言、伪代码、流程图。

试读结束，需要全本PDF请购买 www.ertongbook.com

1) 自然语言

自然语言就是人们日常使用的语言，可以是汉语、英语或其他语言。用自然语言描述算法具有通俗易懂的优点，但缺点也比较多，体现在以下几个方面：

(1) 比较繁琐。往往要用一段繁琐的文字才能说清楚程序所要进行的操作。

(2) 容易出现“歧义性”。自然语言往往要根据上下文才能正确判断出其含义，不太严谨。

(3) 用自然语言容易描述顺序执行的步骤，但如果算法中包含判断和转移情况时，用自然语言就不那么直观清晰了。

例 1.1　将两个变量 x 和 y 的值互换。

问题分析：两个人交换座位，只要各自去坐对方的座位就行了，这是直接交换。一瓶酒和一瓶醋互换，就不能直接从一个瓶子倒入另一个瓶子，必须借助一个空瓶子，先把酒倒入空瓶，再把醋倒入已倒空的酒瓶，最后把酒倒入已倒空的醋瓶，这样才能实现酒和醋的交换，这是间接交换。

在计算机中交换两个变量的值不能用两个变量直接交换的方法，而必须采用间接交换的方法。因此，这里需设一个中间变量 z(相当于空瓶子)。

用自然语言描述如下：

步骤 1，将 x 值存入中间变量 z 中：x→z。

步骤 2，将 y 值存入变量 x 中：y→x。

步骤 3，将中间变量 z 的值存入 y 中：z→y。

2) 伪代码

伪代码(pseudo code)一般介于自然语言与程序设计语言之间，它具有自然语言灵活的特点，同时又接近于程序设计语言的描述。但需指出，用伪代码所描述的算法，一般不能直接作为程序来执行，最后还需转换成用某种程序设计语言所描述的程序。伪代码与程序设计语言最大的区别就在于，伪代码描述比较自由，不像程序设计语言那样受语法的约束，只要使得人们能理解就行，而不必考虑计算机处理时所要遵循的规定或其他一些细节。

例如，在例 1.1 中，将两个变量 x 和 y 的值互换，可以用伪代码描述如下：

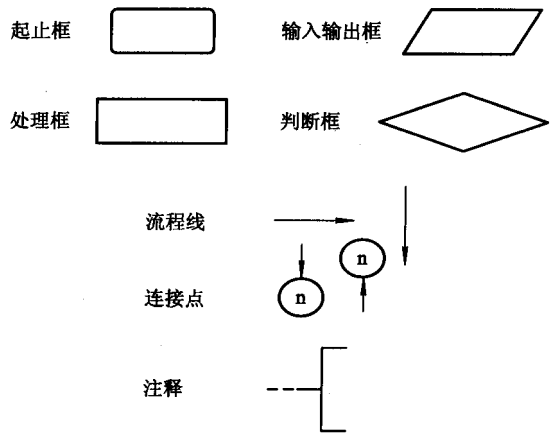
```
BEGIN
  x→z
  y→x
  z→y
END
```

在伪代码描述算法中，关键字可以使用英文，也可以用中文，还可以使用一些符号来表示，并无固定的、严格的语法规则，只要将意思表达清楚，并且书写的格式要清晰易读就可以了，总之，在伪代码描述算法时只需遵循便于书写和阅读的原则即可。

3) 流程图

流程图是用一些图框、流程线以及文字说明来表示算法。用流程图来表示算法，更加直观、形象、容易理解。

(1) 传统流程图。美国国家标准化协会 ANSI(American National Standard Institute)规定了一些常用的流程图符号，各种流程图符号表示如下：



例 1.2　判断一个数 n 是否是素数的算法用流程图表示，如图 1.2 所示。

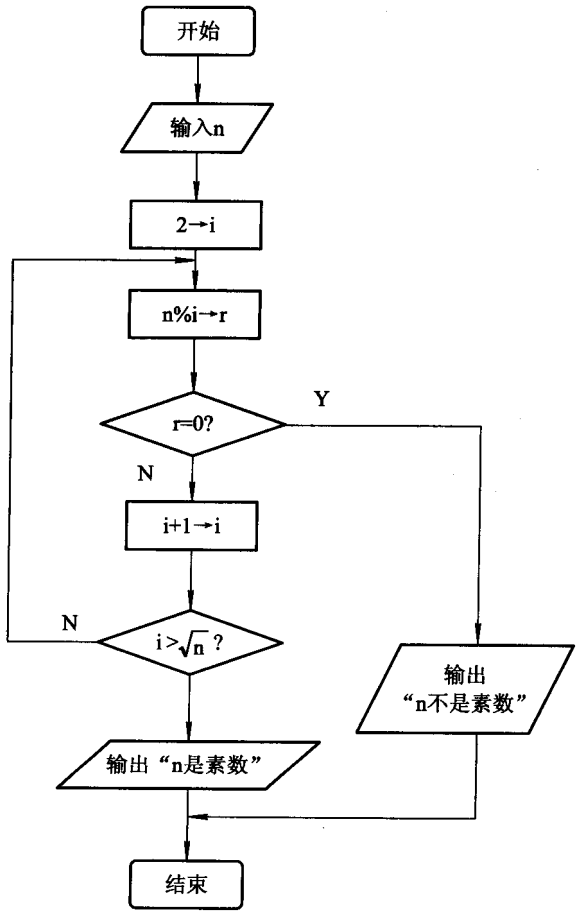


图 1.2　判断 n 是否是素数

这种传统流程图虽然形象直观，但对流程线的使用没有限制，流程随意转移，甚至可能变得毫无规律，难以阅读和维护。

(2) 结构化流程图。传统流程图中流程线的用法不受限制，可能会导致流程图毫无规律。针对以上问题，1966 年，Bohra 和 Jacopini 提出了三种基本结构：顺序结构、选择结构、循环结构。用这三种基本结构作为表示一个良好算法的基本单元，可以改进传统流程图，使传统流程图结构化，从而大大提高了流程图的规律性，也便于人们阅读和维护。

① 顺序结构。顺序结构是最简单的一种基本结构，计算机在执行顺序结构的程序时，按语句出现的先后次序依次执行。如图 1.3(a)所示，计算机将先执行 A 操作，再执行 B 操作。

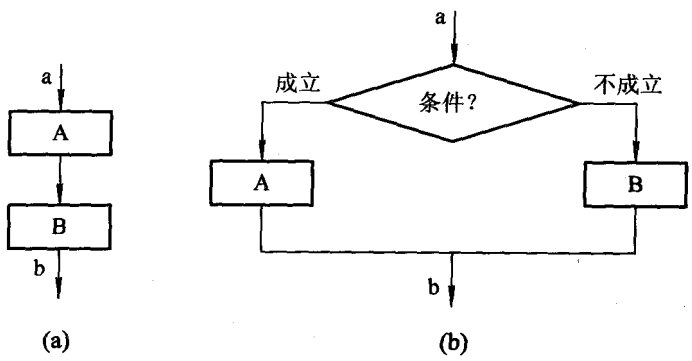


图 1.3　顺序结构与选择结构流程图

② 选择结构。当程序在执行过程中需要根据某种条件的成立与否有选择地执行一些操作时，就需要使用选择结构。图 1.3(b)表示了选择结构的流程图。这种结构包含一个判断框，根据给定的条件是否满足，从两个分支路径中选择执行其中的一个。从图 1.3(b)中可以看出，无论执行哪一个分支路径都将通过汇合点 b，b 点是选择结构的出口点。

③ 循环结构。循环结构用于规定重复执行一些相同或相似的操作。要使计算机能够正确地完成循环操作，就必须使循环在执行有限次数后退出，因此，循环的执行要在一定的条件下进行。根据对条件的判断位置不同，可以有两类循环结构：当型循环和直到型循环。

当型循环结构如图 1.4(a)所示。当程序运行到 a 点时，从 a 点进入当型循环。首先判断条件是否成立，如果条件成立，则执行 A 操作；执行完 A 操作后，再判断条件是否成立，若仍然成立，再执行 A 操作。如此反复执行，直到某次条件不成立时为止，这时不再执行 A 操作，而是从 b 点退出循环。显然，在进入当型循环时，如果一开始条件就不成立，则 A 操作一次都不执行。

直到型循环结构如图 1.4(b)所示。当程序运行到 a 点时，从 a 点进入直到型循环。首先执行 A 操作，然后判断条件是否成立，如果条件成立，则继续执行 A 操作；再判断条件是否成立，若仍然成立，再执行 A 操作。如此反复执行，直到某次条件不成立时为止，这时不再执行 A 操作，而是从 b 点退出循环。显然，在进入直到型循环时，A 操作至少执行一次。

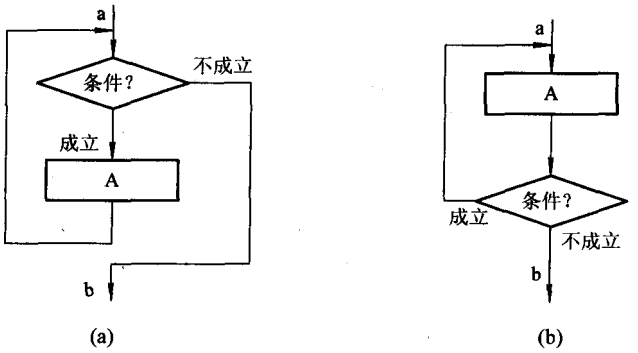


图 1.4　循环结构流程图

以上三种基本结构有以下共同的特点：

- 只有一个入口。
- 只有一个出口。
- 每一个基本结构中的每一部分都有机会被执行到。也就是说，对每一个框来说，都应当有一条从入口到出口的路径通过它。
- 结构内不存在“死循环”(即无终止的循环)。

已经证明，由以上三种基本结构组成的算法，可以解决任何复杂的问题，并且由基本结构构成的算法属于“结构化”的算法，不存在无规律的转移。

(3) N-S 流程图。1973 年，美国学者 I.Nassi 和 B.Shneiderman 提出了另一种流程图形式。在这种流程图中完全去掉了流程线，全部算法写在一个矩形框内，在框内还可以包含其他的框，该流程图称为 N-S 流程图。这种流程图用以下三种基本元素框来表示程序设计中的三种基本结构。

① 顺序结构。如图 1.5(a)所示，A 和 B 两个框组成一个顺序结构。A 框或 B 框可以是一个简单的操作(如读入数据或输出等)，也可以是三种基本结构之一。

② 选择结构。如图 1.5(b)所示，当条件 P 成立时执行 A 框操作，P 不成立时则执行 B 框操作。这里 A 和 B 不可能同时都操作，这是两个分支的选择结构。

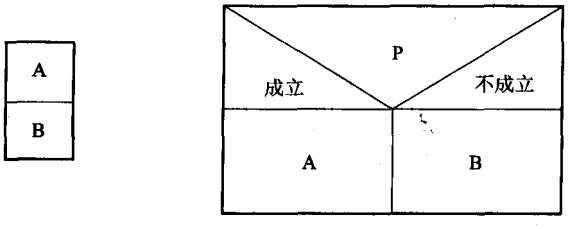


图 1.5　顺序结构与选择结构

③ 循环结构。当型循环如图 1.6(a)所示，当条件 P1 成立时反复执行 A 框中的操作，直到 P1 条件不成立时为止。直到型循环如图 1.6(b)所示，反复执行 A 框中的操作，直到 P1 条件不成立时为止。当型循环与直到型循环的区别：当型循环先判断条件是否成立，再执行循环中的 A 框；而直到型循环先执行一次 A 框，再判断条件是否成立；直到型循

环最少会执行一次 A 框，而当型循环中如果第一次判断时条件就不成立，则 A 框一次都不执行。

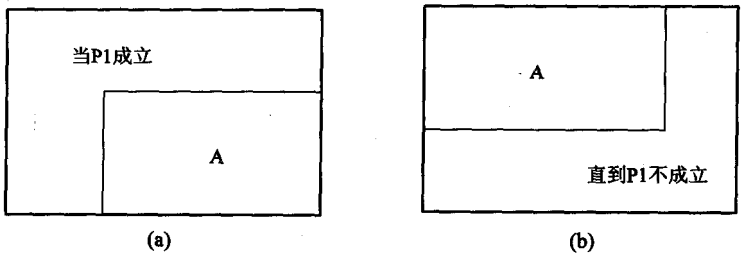


图 1.6　循环结构

例 1.1 算法的流程可以表示为图 1.7 所示的形式。

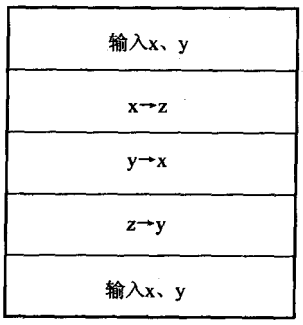


图 1.7　交换变量 x 和 y 的值

例 1.3　求最大公约数的算法用 N-S 流程图描述，如图 1.8 所示。

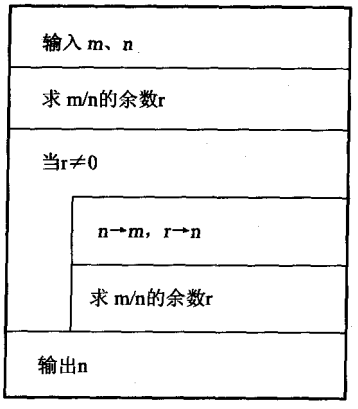


图 1.8　求 m、n 的最大公约数

N-S 流程图具有以下明显的优点：

- 功能明确，即图中的每个矩形框所代表的特定作用域可以明确地分辨出来。
- 能够保证程序整体是结构化的，它不允许任意转移和设置出口，因此可以保证单入口、单出口的程序结构。
- 很容易实现和表示嵌套结构，这为较复杂的程序设计提供了方便的途径。

但由于 N-S 流程图仅使用三种基础结构形成流程，因此在某些程序设计时可能会比较繁琐，有一定的困难。

1.2.2 结构化程序设计

结构化程序设计方法要求将程序的结构规定为顺序、选择和循环三种基本结构。一个结构化程序就是用高级语言表示的结构化算法。用三种基本结构组成的程序必然是结构化的程序，这种程序便于编写、阅读、修改和维护，并且会减少程序出错的机会，从而提高程序的可靠性，保证程序的质量。

结构化程序设计强调程序设计风格和程序结构的规范化，提倡清晰的结构。怎样才能得到一个结构化的程序呢？具体说，采取以下方法可以保证得到结构化的程序。

1. 自顶向下、逐步细化的设计过程

自顶向下、逐步细化的设计过程包括以下两个方面：

(1) 将一个复杂问题的解法分解和细化成由若干模块组成的层次结构。

(2) 将一个模块的功能逐步分解细化为一系列的处理步骤，直到细化为某种程序设计语言的语句或某种机器指令。

自顶向下、逐步细化的设计过程具有以下两个优点：

(1) 自顶向下、逐步细化的方法符合人们解决复杂问题的普遍规律，可以显著提高程序设计的效率。

(2) 用先全局后局部、先整体后细节、先抽象后具体的逐步细化过程，设计出的程序具有清晰的层次结构，容易阅读和理解。

正如我们要写好一篇文章那样，首先要订出总纲，初步拟订好文章分为哪几部分，然后考虑每一部分中应包括哪几方面内容。这样一直细分下去，最后确定出章、节、段的目录提纲。这种先确定总纲，再细分确定详细提纲，最后才开始动手写文章的方法同样适用于程序设计。

2. 模块化设计

模块化设计是指把一个大程序按人们能理解的大小规模进行分解。由于经过分解后的各模块比较小，因此容易实现，也容易调试。

在进行模块化程序设计时，应重点考虑以下两个问题：

(1) 按什么原则划分模块？

(2) 如何组织好各模块之间的联系？

1) 按功能划分模块

划分模块的基本原则是使每个模块都易于理解。按照人类思维的特点，按功能来划分模块最为自然。在按功能划分模块时，要求各模块的功能尽量单一，各模块之间的联系尽量少。满足这些要求的模块有以下几个优点：

(1) 模块间的接口关系比较简单，并且每个模块都是人的智力所能及的。因此，这种程序的可读性和可理解性都比较好。

(2) 各模块的功能比较单一，当需要修改某一功能时，一般只涉及到一个模块，不会影响到其他模块。因此，这种程序的可修改性和可维护性比较好。

(3) 人们脱离程序的上下文也能单独验证一个模块的正确性，便于对程序进行模块化测

试读结束，需要全本PDF请购买 www.ertongbook.com