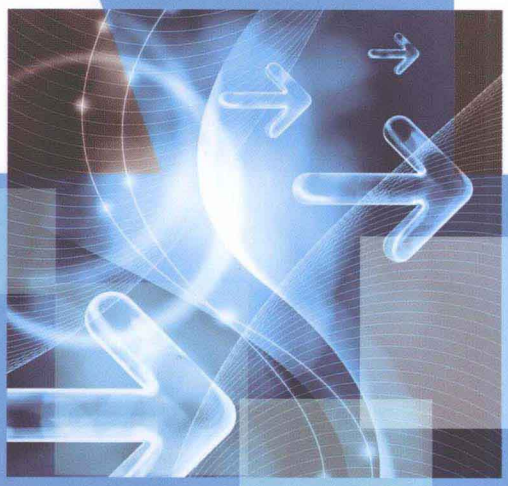
21世纪高等学校计算机科学与技术规划教材



Visual Basic
Chengxu Sheji Jiaocheng

Visual Basic 程序设计教程

主编 郭 理 裘祖旗 张 丽
主审 梁 斌

北京邮电大学出版社
www.buptpress.com

21世纪高等学校计算机科学与技术规划教材

Visual Basic 程序设计教程

主编　郭　理　裘祖旗　张　丽
主审　常　森　武子川

北京邮电大学出版社
·北京·

内容简介

本书是 Visual Basic 的基础教程，以 Visual Basic 6.0 中文企业版为平台，通过大量实例，深入浅出地阐述了 Visual Basic 的基本概念、程序设计的基本方法、事件驱动的编程机制以及面向对象的程序设计思想。

本书共 11 章，主要内容包括：Visual Basic 集成开发环境、数据类型及其运算、基本控制结构、常用内部控件、数组、过程、用户界面设计、图形设计、文件操作和数据库设计等。本书上在编排上注意了由简及繁、由浅入深和循序渐进，力求通俗易懂。

本书可作为高等学校计算机程序设计教材，也可作为计算机培训班的教材以及供自学使用。

本书配有《Visual Basic 程序设计实践教程》，并配有教师用电子教案。

图书在版编目(CIP)数据

Visula Basic 程序设计教程/郭理，裘祖旗，张丽主编．--北京：北京邮电大学出版社，2011.1
ISBN 978-7-5635-2524-9

Ⅰ.①V… Ⅱ.①郭… ②裘… ③张… Ⅲ.①BASIC 语言—程序设计—教材 Ⅳ.①TP312
中国版本图书馆 CIP 数据核字(2010)第 255181 号

书　　名 Visual Basic 程序设计教程
主　　编 郭　理　裘祖旗　张　丽
责任编辑 张雪祥
出版发行 北京邮电大学出版社
社　　址 北京市海淀区西土城路 10 号(100876)
电话传真 010-82333010　62282185(发行部)　010-82333009　62283578(传真)
电子信箱 ctrd@buptpress.com
经　　销 各地新华书店
印　　刷 北京市梦宇印务有限公司
开　　本 787 mm×1 092 mm　1/16
印　　张 18.5
字　　数 450 千字
版　　次 2011 年 1 月第 1 版　2011 年 1 月第 1 次印刷

ISBN 978-7-5635-2524-9　　定价：32.00 元

编 委 会

前　言

Visual Basic 是 Microsoft 公司推出的一种基于对象的可视化编程语言，是目前 Windows 平台上设计应用程序最为快捷的计算机高级语言之一，具有简单易学、功能强大、应用广泛的特点。近些年来，Visual Basic 已经成为许多高等院校首选的教学用程序设计语言。

本书共分为 11 章，涵盖了 Visual Basic 程序设计的主要内容。第 1 章主要介绍程序设计语言的发展、算法基础和程序设计方法。第 2 章主要介绍 Visual Basic 的集成开发环境及可视化编程的基本概念。第 3 章主要介绍 Visual Basic 的语言中的数据类型、变量、运算符、表达式和常用内部函数等基本知识。第 4 章主要介绍结构化程序设计的 3 种基本结构，即顺序、选择、循环结构程序设计所涉及的基本概念、基本语句及相关应用。第 5 章主要介绍 Visual Basic 中的常用控件及其使用。第 6 章主要介绍 Visual Basic 中数组的基本概念、数组类型和数组的使用方法。第 7 章主要介绍过程的定义、调用及过程的应用。第 8 章主要介绍菜单、工具栏、多文档界面以及各种对话框的设计。第 9 章主要介绍 Visual Basic 图形设计的基本知识和方法。第 10 章主要介绍 Visual Basic 的文件处理功能和操作。第 11 章主要介绍 Visual Basic 的数据库访问功能及实际案例。

本书在结构编排和内容的选择上，集一线教师多年的教学经验，注重深入浅出、循序渐进，强调理论和实践的结合，在内容上以加强基础、突出应用提高能力为目标，力求使教材具有可读性、实用性和适用性。书中包含了大量典型算法的分析及示例，各章例题尽量做到既能说明有关概念，又具有一定的实际意义，每章最后均配有一定数量的习题，以检查学生的学习效果。

为了方便教学和读者上机操作练习，作者还组织编写了《Visual Basic 程序设计实践教程》一书，作为与本书配套的辅助教材。另外，还有与本书配套的教学课件，供教师教学参考。

本书由郭理、裘祖旗、张丽主编。第 1 章、第 5 章由田敏编写，第 2 章、第 9 章由郭理编写，第 3 章、第 10 章由张丽编写，第 4 章由郑瑶编写，第 6 章由戴建国编写，第 7 章由秦怀斌编写，第 8 章由裘祖旗编写，第 11 章由蔡文青编写。在本书编写的过程中，得到了许多同行专家、教师的帮助，在此表示衷心的感谢。

由于作者学识水平有限，书中的错误或不足之处在所难免，敬请读者批评指正。

编　者

目　录

第1章　程序设计基础

自然语言是一套具有语法、词法规则的系统。语言是思维的工具，思维是通过语言来表述的。计算机语言，也称程序设计语言(Programming Design Language)，是计算机可以识别的语言，用于描述解决问题的方法，供计算机阅读和执行，是人和计算机沟通的桥梁。为了让计算机帮助人们解决实际问题，必须事先把处理问题的方法、步骤以计算机可以识别和执行的操作表示出来，也就是说要编写程序。使用程序设计语言编写的用来使计算机完成一定任务的一段有限指令序列称为程序，编写程序的工作则称为程序设计。

随着计算机技术的迅速发展，程序设计语言经历了由低级向高级发展的几个阶段，程序设计方法也得到了不断的发展和提高。本章主要讲述程序设计语言的发展、算法基础、程序设计方法等。

1.1　程序设计语言的发展

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，而用各种程序设计语言编写程序和利用计算机执行程序，是人类利用计算机解决问题的主要方法和手段。随着计算机技术的发展，不同风格的计算机语言不断出现，逐步形成了计算机语言体系。计算机语言按其发展程度可以划分为：机器语言、汇编语言和高级语言。其中机器语言和汇编语言属于低级语言，高级语言又分为面向过程的语言和面向对象的语言。

1.1.1　机器语言

在1952年以前，唯一可以使用的程序设计语言是现在被称为“低级语言”的机器语言。机器语言是由计算机能够直接识别的机器指令系统组成，机器语言的每一条语句实际上就是由“0”和“1”组成的计算机指令，每一条机器指令规定了计算机要完成的某个操作。例如，机器指令1011000000001001表示把数字9送入累加器AL中。由于指令系统是依赖于计算机的，因此机器语言也与计算机相关，不同类型的计算机具有不同的机器语言。

用机器语言编写的程序表现为一系列二进制信息，编写起来非常烦琐，难读、难记、难写、难检查、难调试是其缺点，尤其是用机器语言编写的程序完全依赖于机器，所以程序的可移植性差。其优点是机器能直接识别、占用内存小、执行速度快，不需要做其他的辅助工作。

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(如ADD表示加，SUB表示减，MOV表示数据传送，HLT代表停止)代替机器指令。用这样一些符号代替机器指令所产生的语言称为汇编语言，也称为符号语言。例如，MOV

AL,9 表示把数字 9 送入累加器 AL 中。这些符号一般是人们容易记忆和理解的英文缩写,所以又称为助记符。用汇编语言编写的程序称为汇编语言源程序,这种程序计算机无法执行,必须通过翻译成机器语言目标程序之后才能执行。

汇编语言在编写、阅读和调试等方面比机器语言有了很大的进步,且能够直接对硬件操作,具有很强的灵活性,因此直到现在仍广泛应用于实时控制、实时处理等领域。但是,汇编语言仍然是一种面向机器的语言,不同类型的计算机具有不同的汇编语言。因此,使用汇编语言时,需深入了解计算机硬件的一些细节。

1.1.3 高级语言

1. 高级语言简介

汇编语言虽然较机器语言有所改善,但未从根本上摆脱指令系统的束缚,它与机器指令仍然是一一对应的,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,必须从两方面下工夫:一是力求接近于自然语言;二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。

从 20 世纪 50 年代末起,出现了许多高级程序设计语言。高级程序设计语言是一种与计算机指令无关、表达形式更接近于被描述的问题的程序设计语言。这里所谓的"高级"并不意味着它"高不可攀"、"深奥莫测",而是表示它有别于与计算机硬件有关的机器语言和汇编语言,它独立于计算机的类型,从而能够更加容易被人掌握、更加便于程序的编写。使用这些语言编写程序时,程序设计者不必关心机器的内部结构和工作原理,而只需把主要精力集中在解决问题的思路和方法上。高级语言较之汇编语言更接近于自然语言,描述计算公式时与数学上的表示大体一致。例如,Visual Basic 中的 Print 5+6 语句表示打印 5+6 的值。

在使用高级语言设计程序时,可以有两种方法:一种是面向过程的程序设计方法;一种是面向对象的程序设计方法。早期的高级程序设计语言都是面向过程的,在编写程序时不但要说明需要什么数据、进行什么处理,而且要说明获得这些数据以及对数据进行处理的具体过程。例如,早期出现的 BASIC、Quick BASIC、PASCAL、FORTRAN、COBOL、C 等高级语言,适用于 DOS 环境的编程,采用的是面向过程的程序设计方法;而较新出现的 Visual Basic,Visual C++、Visual FoxPro、Delphi、Java 等适用于 Windows 环境的编程,采用的是面向对象的程序设计方法,是面向对象技术和程序设计语言相结合的产物,其特征是封装性、继承性和多态性。使用面向对象程序设计语言编写程序时不必关心具体操作的细节,从而提高了程序设计的质量和效率。

任何一种高级程序设计语言都有严格的词法规则、语法规则和语义规则。词法规则规定了如何从语言的基本符号集构成单词;语法规则规定了如何由单词构成各个语法单位(如表达式、语句、程序等);语义规则规定了各个语法单位的含义。灵活、巧妙地应用高级程序设计语言可以设计出各种解决实际问题的程序。

2. 翻译程序

用高级语言编写的程序(称为源程序)需要翻译为机器语言才能被计算机所执行,这种将高级语言源程序翻译成机器能识别的目标程序的处理程序称为翻译程序。翻译程序也是一种软件,人们称之为"程序的程序"。每种高级语言都有自己的翻译程序,相互不能代替。根据其

翻译的方式，可分为解释方式和编译方式。翻译程序的主要职能有：

- 对源程序进行词法分析和检查。
- 对源程序进行语法检查和语义分析。
- 对变量分配存储空间。
- 生成目标程序。

解释方式的翻译工作由解释程序来完成，解释程序对源程序的代码逐行翻译为可执行指令代码，翻译一句执行一句，如图 1-1 所示。这个过程有点像同声翻译。解释程序在执行过程中不生成可执行的文件，因此程序运行速度较慢，而且应用必须在解释环境下运行，如果源程序出错，它将停止程序的执行。早期的计算机使用解释程序可以在较小的内存中运行大程序。BASIC 语言就是解释型语言。发展到今天的 Visual Basic 保留了这个特点，同时它也可以使用编译方式运行。

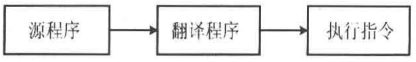


图 1-1　解释程序

编译方式的翻译工作由编译程序来完成，编译程序的翻译过程如图 1-2 所示。图中虚线部分为语言处理程序，包括两个部分。首先由编译器(Compiler)将源程序翻译为目标程序代码，然后由连接程序(Linker)将目标程序和系统资源(如函数库、系统过程等)连接在一起形成可执行程序。产生的可执行程序可以脱离编译系统独立运行。这个有点像外文文献的书面翻译过程。

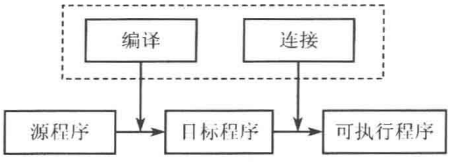


图 1-2　编译程序过程

编译方式比解释方式执行速度快，但不灵活；若修改了源程序，则必须重新编译。FORTRAN、PASCAL、COBOL、C 等语言均采用编译方式处理。

在计算机文件中，可执行程序大多以 .exe 为后缀，因此很好辨认。现在的许多编译系统不再区分编译的连接过程，而是直接将它们组合在一起将用户源程序“构造(Builder)”为可执行程序。我们也注意到，在 Windows 系统中，将这类文件统称为“应用程序文件”。Visual Basic 源程序也可以使用编译系统直接编译为可执行文件。

计算机的应用领域很广泛。为了适应不同的需要，程序设计语言又各具特点。例如，有适合编写系统软件的，有适合进行科学计算的，有适合数据库管理的，有适合图形设计的，还有适合人工智能的，更有一些语言同时具备多种功能。从应用的角度难以对程序设计语言进行严格分类。而且，随着计算机科学的发展及应用领域的迅速扩展，各种语言版本都在不断变化，功能在不断更新、增强。

每个时期都有一批语言在流行，又有一批语言在消亡。因此，我们应掌握程序设计语言中本质性的、规律性的东西，即程序设计方法。

1.2　算法基础

一个程序应包括以下两个方面。

(1)对数据的描述。在程序中要指定数据的类型和数据的组织形式，即数据结构(Data Structure)。

(2)对操作的描述。即操作步骤,也就是算法(Algorithm)。

数据是操作的对象,操作的目的是对数据进行加工处理,以得到期望的结果。数据结构和算法是程序最主要的两个要素。因此,著名计算机科学家沃思(Nikiklaus Wirth)提出了一个公式:

程序=数据结构+算法

除了以上两个要素外,程序与采用的具体设计方法和计算机语言有关。因此,上述公式可进一步完善为:

程序=数据结构+算法+程序设计方法+语言工具和环境

算法是程序的灵魂,它解决"做什么"和"怎么做"的问题,它是程序设计的前提,而计算机语言只是实现算法的工具。有了正确的算法,就可以利用任何一种计算机语言编写程序,使计算机进行工作,得出正确的结果。

1.2.1 算法的概念

做任何事情都有一定的步骤。例如,厨师炒菜,首先要买菜,洗菜,切菜,然后生火,加配料进行炒菜,直到完成;要买家用电器,先要选好货物,然后开票,付款,拿票,取货,回家;要到医院看病,先挂号,再问诊、检查、诊断,然后取药等。这些步骤都是按一定的顺序进行的,缺一不可,次序错了也不行。在日常生活中,由于已养成习惯,所以人们并没有意识到每件事情都需要事先设计出"行动"步骤,事实上都是按照一定的规律进行的。

算法是对解决某一特定问题的操作步骤的具体描述。广义地说,算法就是为解决某个问题而采取的方法和步骤。例如,厨师炒菜的操作步骤就是"烹调算法";到医院看病就是"看病算法"。在这里,只讨论计算机算法。计算机中的算法就是为计算机解决问题而设计的有明确意义的操作步骤的有限集合。

对同一问题,可以有不同的解题方法和步骤。例如,求 1+2+3+…+100。有人可能先进行 1+2,再加 3,再加 4,一直加到 100;而有的人采取:100+(1+99)+(2+98)+…+(49+51)+50 的方法。因此,为了有效地进行解题,不仅需要保证算法的正确,还要考虑算法的质量,选择合适的算法。

计算机算法可分为两大类:数值计算算法和非数值计算算法。数值计算算法的目的是求数值解,例如求方程的根、求函数的定积分等;非数值计算算法包括的范围很广,最常见的是用于管理领域,如用于文字处理和图形图像处理以及信息的排序、分类、查找等方面的算法。

1.2.2 算法的特性

一个算法应该具有以下特点。

- 有穷性。一个算法应包含有限的操作步骤,而不能是无限的。事实上有穷性指的是"在合理的范围之内"。
- 确定性。算法中的每一步操作必须含义确切,不能有二义性。
- 有效性。算法中的每一步操作都必须是可执行的。
- 有零个或多个输入。算法常需要对数据进行处理,因此,算法常需要数据输 λ。
- 有一个或多个输出。算法的目的是用来解决一个给定的问题,因此,它应提供算法产生的结果。

1.2.3　算法的表示

为了表示一个算法，可以用不同的方法。常用的表示算法的方法有自然语言、传统流程图、N-S 流程图、伪代码、PAD 图等。本书主要介绍用自然语言、传统流程图、N-S 流程图表示算法。

1. 用自然语言表示算法

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。用自然语言表示算法通俗易懂，但文字冗长，容易出现“二义性”。自然语言表示的含义往往不太严格，要根据上下文才能判断其正确含义。因此，除了很简单的问题以外，一般不使用自然语言来描述算法。

2. 用传统流程图表示算法

传统流程图是用一些图框、流程线以及文字说明来描述操作过程。用图形表示算法，直观形象，易于理解。美国国家标准化协会（American National Standard Institute，ANSI）规定了一些常用的流程图符号，各种流程图符号表示如表 1-1 所示。

表 1-1　各种流程图符号的表示

流程图符号	名称	作用
	起止框	表示流程开始或结束
	输入/输出框	表示数据输入或输出
	判断框	表示对一个给定的条件进行判断，根据给定的条件是否成立来决定如何执行其后的操作。它有一个入口，两个出口
	处理框	表示基本处理功能的描述
	连接点	用于将画在不同地方的流程线连接起来
↓或→	流程线	表示流程的路径和方向
……[	注释框	表示对流程图给出必要的文字说明

用流程图表示算法直观形象，比较清楚地显示出各个框之间的逻辑关系。但是，这种流程图占用篇幅较多，尤其当算法比较复杂时，画流程图既费时又不方便。在结构化程序设计方法推广之后，许多书刊已用 N-S 结构化流程图代替这种传统的流程图。但是，每一个程序设计人员都应当熟练掌握传统流程图，会看会画。

3. 用 N-S 流程图表示算法

传统的流程图用流程线指出各流程图框的执行顺序，但对流程线的使用没有严格限制。因此，使用者可以不受限制地使流程随意地转来转去，使流程图变得毫无规律。阅读者要花很大精力去追踪流程，使人难以理解算法的逻辑。为此，人们对流程图进行了改进。1973 年，美国学者 I. Nassit 和 B. Shneiderman 提出了一种新的流程图，这种流程图称为 N-S 流程图（其中的 N 和 S 取自两位学者的英文名字的第一个字母）。在这种流程图中，完全去掉了带箭头的流程线，全部算法写在一个大矩形框中，在该大矩形框内还可以包含一些从属于它的小矩形框。这种流程图特别适合于描述结构化程序设计算法。

【例 1-1】　求 5！ $=1\times2\times3\times4\times5$ 的算法。

算法思路：可以设两个变量，一个变量代表被乘数，一个变量代表乘数。不另设变量存放乘积结果，而直接将每一步骤的乘积放在被乘数变量中。设 p 为被乘数，i 为乘数，算法如下：

①用自然语言描述算法。

步骤 1:将 1 赋值给 p,可表示为 1⇒p。

步骤 2:将 2 赋值给 i,可表示为 2⇒i。

步骤 3:计算 p×i ,乘积仍放入变量 p 中,可表示为 p×i⇒p。

步骤 4:使 i 的值加 1,即 i+1⇒i。

步骤 5:如果 i 不大于 5,返回重新执行步骤 3 以及其后的步骤 4 和步骤 5;否则,执行步骤 6。

步骤 6:输出 p,即 p 的值就是 5!,算法结束。

②用传统流程图描述算法(见图 1-3)。

③用 N-S 流程图描述算法(见图 1-4)。

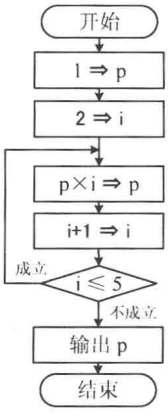


图 1-3 传统流程图

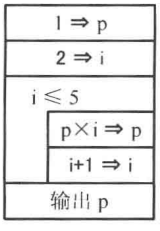
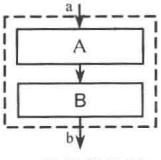


图 1-4 N-S 流程图

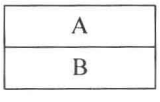
1.2.4 3 种基本结构

为了提高算法的质量,使算法的设计和阅读方便,不允许无规律地使流程随意转向,只能顺序地进行下去。但是,算法上难免会包含一些分支和循环,而不可能全部由一个一个框顺序组成。为了解决这个问题,1966 年,Bohra 和 Jacopini 提出了以下 3 种基本结构,用这 3 种基本结构作为表示一个良好算法的基本单元。这 3 种基本结构是顺序结构、选择结构和循环结构。

(1)顺序结构:如图 1-5 所示。图 1-5(a)的虚线框内是一个顺序结构,其中 A 和 B 两个框是顺序执行的。即在执行完 A 框所指定的操作后,接着执行 B 框所指定的操作。顺序结构是最简单的一种基本结构。



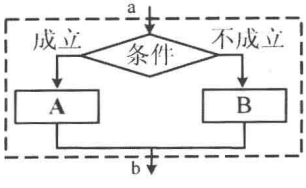
(a)传统流程图



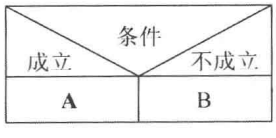
(b)N-S 流程图

图 1-5 顺序结构

(2)选择结构:又称选取结构或分支结构,如图 1-6 所示。图 1-6(a)的虚线框内是一个选择结构。此结构中包含一个判断框,根据给定的条件是否成立而选择执行 A 框或 B 框。请注意,无论条件是否成立,只能执行 A 框或 B 框之一,不可能既执行 A 框又执行 B 框。无论走哪一条路径,在执行完 A 或 B 之后,都经过 b 点。



(a)传统流程图

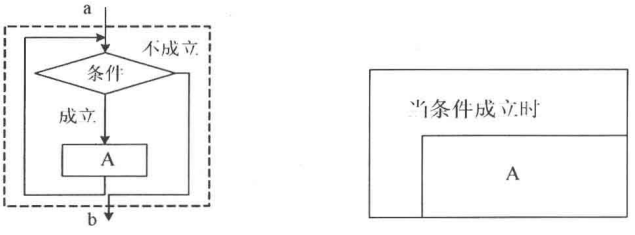


(b)N-S 流程图

图 1-6 选择结构

(3)循环结构：又称重复结构，即反复执行某一部分的操作。有两类循环结构，即当型循环结构和直到型循环结构。

①当型循环结构。如图 1-7 所示。以图 1-7(a)为例，虚框以内是一个当型循环结构。它的功能是当给定的条件成立时，执行 A 框操作，执行完 A 后，再判断条件是否成立，如果仍然成立，再执行 A 框，如此反复执行 A 框，直到某一次条件不成立为止，此时不执行 A 框，而从 b 点脱离循环结构。显然，如果一开始条件就不成立，则 A 操作一次都不执行。

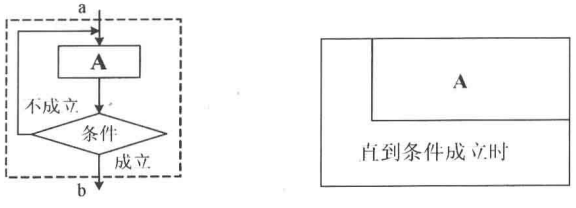


(a)传统流程图　　(b)N-S 流程图

图 1-7　当型循环结构

②直到型循环结构。如图 1-8 所示。以图 1-8(a)为例，虚框以内是一个直到型循环结构。它的功能是先执行 A 框，然后判断给定的条件是否成立，如果条件不成立，则再执行 A，然后再对条件作判断，如此反复执行 A，直到给定的条件成立为止，此时不再执行 A，从 b 点脱离本循环结构。显然，无论条件是否为真，A 操作至少执行一次。

对同一个问题既可以用当型循环来处理，也可以用直到型循环来处理，当型循环结构的条件与直到型循环结构的条件互逆。



(a)传统流程图　　(b)N-S 流程图

图 1-8　直到型循环结构

以上 3 种基本结构有以下共同特点。

● 只有一个入口(基本结构图中 a 点)，一个出口(基本结构图中 b 点)。

● 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 3 种基本结构顺序组成的算法结构，可以解决任何复杂的问题。由基本结构所构成的算法属于“结构化”的算法，它不存在无规律的转向。

1.2.5　算法举例

说明：以下例子算法均用 N-S 流程图表示，如果用传统流程图表示算法，请读者自行练习。

【例 1-2】　将两个变量 X 和 Y 的值互换。

算法思路：计算机中交换两个变量的值不能直接交换，需要引入一个中间变量 Z 交换。例如，一瓶酒和一瓶水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

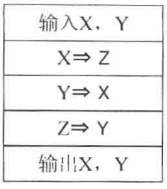


图 1-9 交换两变量

入空瓶，再把水倒入已倒空的酒瓶，最后把酒倒入已倒空的水瓶，这样才能实现酒和水的交换。算法流程图如图 1-9 所示。

【例 1-3】 比较 3 个数 a、b、c 的大小，要求按从小到大的顺序输出。

算法思路：首先比较 a 和 b 两个数，如果 a 的值比 b 的值大，互换 a 与 b 的值。其次比较 a 和 c 两个数，如果 a 的值比 c 的值大，互换 a 与 c 的值。再比较 b 和 c 两个数，如果 b 的值比 c 的值大，互换 b 与 c 的值。算法流程图如图 1-10 所示。

【例 1-4】 求 $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\cdots+\frac{1}{99}-\frac{1}{100}$。

算法思路：这是一个求累加和的问题，每次要累加的项是一个带符号的分数，可设一个存放和值的变量 sum，初值为 0，然后重复执行 sum＝sum＋累加项；累加项的分子为 1，分母是一个有规律的值，可设一个变量 deno 表示分母，初值为 2，每次累加 deno＝deno＋1，终止累加的条件是 deno＞100；符号的变化规律是：奇数项为负数，偶数项为正数，每次累加，符号交替改变。可设一个表示符号改变的变量 sign，初值为 1，每次累加都与“-1”相乘，以决定要累加的某项的符号。算法流程图如图 1-11 所示。

思考：如图 1-11 所示的循环结构为直到型循环结构，能否改成当型循环结构，请读者自行改之。

【例 1-5】 对于一个大于等于 3 的正整数，判断它是不是素数。

算法思路：所谓素数，是指除 1 和该数本身之外，不能被其他的任何整数整除的正整数。因此，可用循环结构，将正整数作为被除数，设为变量 n；将 2 到 n-1（由于数之间存在着倍数关系，实际上是从 2 到$\sqrt{n}$即可）的各个整数轮流作为除数，设为变量 i；另设变量 k 表示$\sqrt{n}$。如果 n 能被 2 到 k 之中的任何一个数整除，即余数为 0，则提前结束循环，此时 i 必然小于或等于 k；如果 n 不能被 2 到 k 之中的任何一个数整除，即余数都不为 0，因此 i＝k＋1，然后才终止循环。可在循环之后判别 i 的值是否大于或等于 k＋1，若是，则表明 n 为素数。算法流程图如图 1-12 所示。

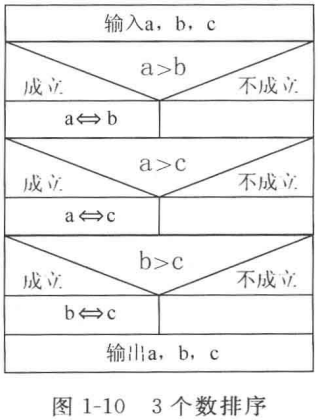


图 1-10 3 个数排序

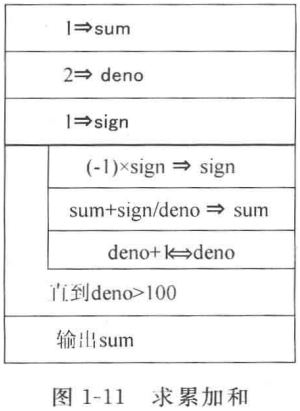


图 1-11 求累加和

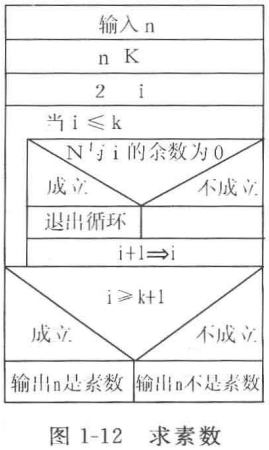


图 1-12 求素数

1.3 程序设计

程序设计是一个使用程序设计语言（如机器语言、汇编语言和高级程序设计语言）产生一系列的告诉计算机该做什么的指令的过程。程序设计是软件开发过程中的一个重要环节，程

序的质量主要取决于软件总体设计的质量，但所选择的程序设计语言的特性和程序设计的途径也会对程序的可靠性、可读性、可测性和可维护性产生深刻的影响。

因此，在进行程序设计之前必须根据实际需求确定使用什么程序设计语言来实现这个系统，并且要养成良好的程序设计风格，编写出逻辑简明清晰、易读易懂的好程序。

广义的程序设计并不仅仅是简单的编写程序代码，它是一个过程，共包括了需求分析、总体设计、详细设计、编码、测试、运行与维护 6 个步骤。

①需求分析。这项工作一般由与程序编写小组工作关系十分密切的系统分析员来完成的。在这个步骤中应完成的工作包括：严格准确地定义所需要解决的实际问题；确定问题的输入和输出；确定问题在技术上、经济上是否可行。

②总体设计。该步骤要设计解决问题的总体方案，用它来一步一步显示解决问题的过程，其中主要工作是将复杂的系统划分为功能相对独立的模块，并确定各个模块的层次结构。

③详细设计。该步骤的主要工作是给出各个功能模块的详细描述，设计实现各功能模块的算法。常用的描述工具有结构图、流程图和算法描述语言等。

④编码。该步骤的主要工作是根据详细设计的规格要求，选择适当的程序设计语言，编写程序代码并输入计算机。

⑤测试。程序测试是为了发现已编写好的程序中的错误而执行程序的过程。一般需要先进行单元测试，然后再进行集成测试，从而得到符合要求的程序模块和完整的软件系统。

⑥运行与维护。该步骤是将程序投入实际运行，并进行正确性、完整性、扩展性等各种维护工作。

使用高级语言进行程序设计时，可用面向过程的程序设计方法，也可用面向对象的程序设计方法。

1.3.1　面向过程的程序设计

在面向对象的程序设计方法出现以前，一般都采用面向过程的程序设计方法。早期的计算机是用于数学计算的工具，例如，用于计算炮弹的飞行轨迹。为了完成计算，就必须设计出一个计算方法，或解决问题的过程。因此，软件设计的主要工作就是设计求解问题的过程。

随着计算机硬件系统的高速发展，计算机的性能越来越强，用途也更加广泛，不再仅限于数学计算。由于所处理的问题日益复杂，程序也就越来越复杂和庞大。20 世纪 60 年代产生的结构化程序设计思想，为使用面向过程的方法解决复杂问题提供了有利的手段。因而，在 20 世纪 70 年代到 20 世纪 80 年代，结构化程序设计方法成为所有软件开发设计领域及每个程序员都采用的方法。

结构化程序设计(Structured Programming)的思路是自顶向下、逐步求精。其程序结构按功能划分为若干个基本模块，这些模块形成一个树状结构。各模块之间的关系尽可能简单，在功能上相对独立。每一模块内部均是由顺序、选择和循环 3 种基本结构组成。其模块化实现的具体方法是使用子程序。结构化程序设计由于采用了模块分解与功能抽象、自顶向下、分而治之的方法，从而有效地将一个较复杂的程序系统设计任务分解成许多易于控制和处理的子任务，便于开发和维护。

虽然结构化程序设计方法具有很多的优点，但它仍是一种面向数据/过程的设计方法，它把数据和过程分离为相互独立的实体，程序员在编程时必须时刻考虑所要处理的数据的格式。对于不同的数据格式，即使要做同样的处理或对相同的数据格式要做不同的处理，都需编写不同的程序。因此，结构化程序的可重用性不好。另外，当数据和过程相互独立时，总存在着用