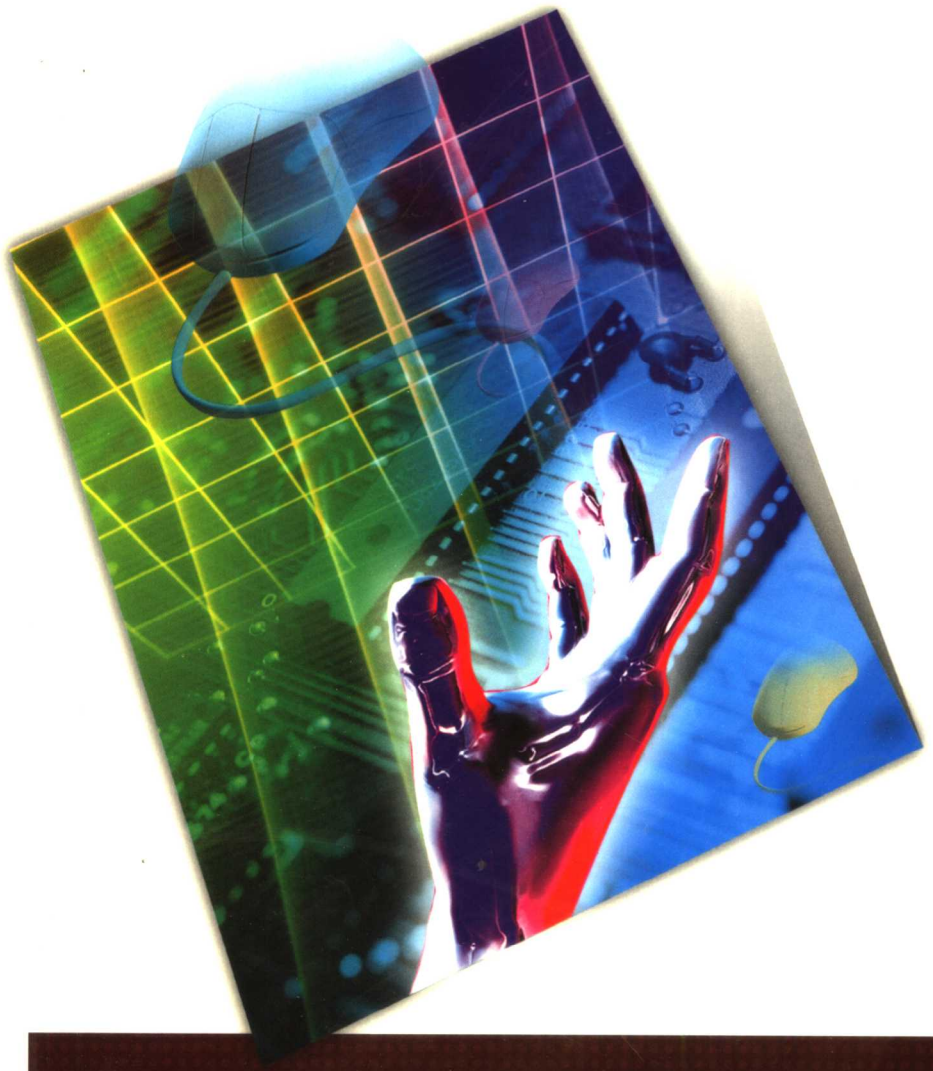
计算机基础教育课程体系规划教材



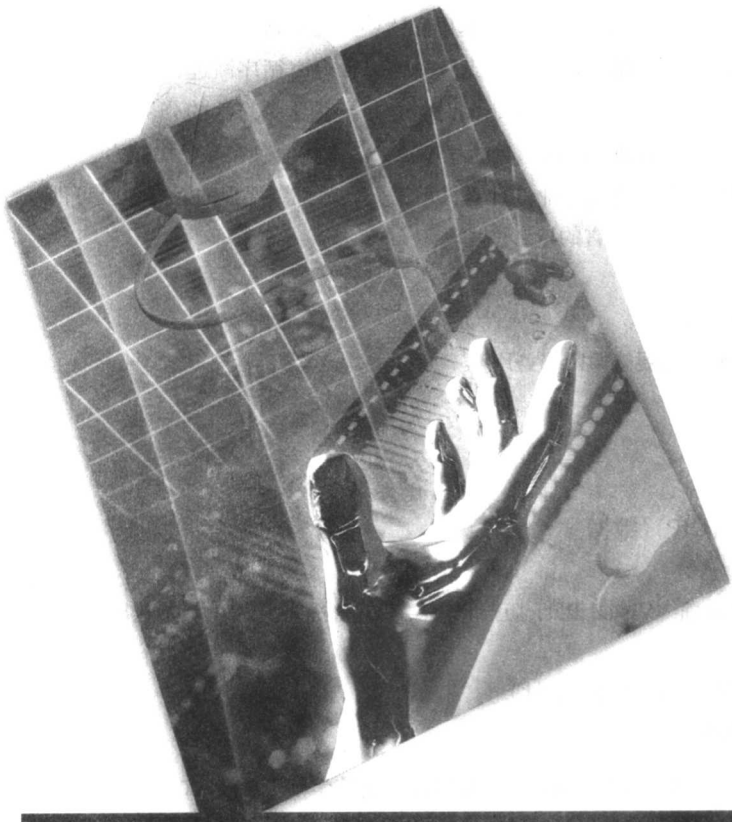
计算机程序设计基础

Visual Basic版

邱李华 郭志强 曹青 编著

机械工业出版社
China Machine Press

计算机基础教育课程体系规划教材



计算机程序设计基础

Visual Basic版

邱李华 郭志强 曹青 编著

机械工业出版社
China Machine Press

本书是根据教育部“非计算机专业计算机课程教学指导分委员会”提出的《非计算机专业计算机基础课程教学基本要求》中有关“计算机程序设计基础”课程的教学要求编写的。

全书共分为14章，主要包括Visual Basic的集成开发环境、程序设计的基础知识、结构化程序的三种基本结构、数组、过程、文件、Visual Basic常用内部控件及ActiveX控件、界面设计、图形设计、数据库基础和软件开发基础。

本书由从事Visual Basic课程教学的一线教师编写，内容循序渐进、深入浅出，注重程序设计基本概念和基本方法的介绍，同时包含了大量常见算法的分析及示例，并配有大量的上机练习题，有助于提高学生的程序设计基本技能和实际操作能力。

本书可作为高等学校非计算机专业“计算机程序设计基础”课程的教材，也可供其他学习Visual Basic程序设计语言的读者使用。

图书在版编目（CIP）数据

计算机程序设计基础：Visual Basic版/邱李华等编著. －北京：机械工业出版社，2005.6
（计算机基础教育课程体系规划教材）
ISBN 7-111-16705-8

Ⅰ.计…　Ⅱ.邱…　Ⅲ.BASIC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（2005）第067780号

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）
策划编辑：温莉芳
责任编辑：刘立卿
北京昌平奔腾印刷厂印刷·新华书店北京发行所发行
2005年6月第1版第1次印刷
787mm×1092mm1/16·21印张
印数：0 001-5000册
定价：30.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

丛书序

根据1997年教育部高教司颁发的“加强非计算机专业计算机基础教学工作的几点意见”（简称155号文件）的要求，各校的计算机基础条件已经明显改善，计算机基础教学进入了一个新阶段。

本届非计算机专业计算机基础课程教学指导分委员会分析了当前高校遇到的计算机基础教学的新形势，根据人才培养的基本要求，针对计算机基础教学中普遍存在的问题，提出了“关于进一步加强高等学校计算机基础教学的意见”（俗称白皮书）。并在其附件“计算机基础教学内容的知识结构与课程设置”中提出了“1 + X”的课程方案，即1门“大学计算机基础”（必修）加上几门重点课程（必修或选修）。白皮书及附件自2003年底在高校征求意见以来，受到了普遍的关注，引起巨大反响。教指委根据征求到的意见做了进一步的修改，不久将正式发布。这无疑将直接影响今后高校计算机基础教学的整体架构，也将推动新一轮的计算机基础教材的面世。

机械工业出版社以其敏锐的眼光和雄伟的魄力，怀着为计算机基础教学作贡献的责任感，遵循白皮书提出的理念，于2004年在全国范围内邀请计算机基础教学一线教师，组织编写“1 + X”中规定的6门核心课程及其若干门整合课程。本丛书参考白皮书对于教材建设所提出的建议，努力在以下几个方面做出特色：

1. 对于重点核心课程的教材，体现课程内容的基础性和系统性，基本概念、基本技术与方法的讲解要准确明晰。

2. 体现非计算机专业计算机基础教材的特点，内容要激发学生学习兴趣，通俗易懂，理论联系实验，每一门课都要使学生真正学到一些有用的知识和技术。

3. 保证教材内容的先进性，特别对于技术性、应用性的内容更应如此。

4. 重视实验教材的建设，重点教材都要配备实验教材。

我们希望本丛书的出版对推动计算机基础教育有所帮助，并在使用中不断改进。书中不足之处恳望读者不吝指正。

冯博琴

2005.4

前 言

2002年1月，我们出版了“Visual Basic程序设计教程”及配套的习题集，该套教材是在总结了我们多年讲授程序设计语言（包括Visual Basic）的体会和实践心得的基础上形成的，自出版后在许多大专院校中得到广泛使用并获得了好评。随着我国计算机基础教育改革的深入，2004年7月清华大学出版社出版了“中国高等院校计算机基础教育课程体系”（蓝皮书），对计算机基础教育课程体系提出了新的参考方案。随后，教育部非计算机专业计算机课程教学指导分委员会又提出了“非计算机专业计算机基础课程教学基本要求”（白皮书）。“蓝皮书”和“白皮书”对程序设计基础课程的要求总体上是一致的，“白皮书”对课程内容作了更详细的要求。为此，我们结合新的要求，对原有教材进行改编，形成了本套教材，以适应新时期人才培养的需要。

Visual Basic是当今深受欢迎的程序设计语言之一，其简练的语法、强大的功能、结构化程序设计以及方便快捷的可视化编程手段，使得编写Windows环境下的应用程序变得非常容易。因此，Visual Basic已经成为目前许多高等院校首选的教学用程序设计语言。

本教材在内容的选择、深度的把握、上机练习题及其他习题的设计上，均以“白皮书”的要求为核心，在内容的设计上力求做到深入浅出、循序渐进，既包含程序设计语言的基本知识和程序设计的基本方法与技术，又能与可视化编程有机地结合。在界面的设计上，除了介绍一些常用的内部控件外，还介绍了设计Windows应用程序界面时常用的一些ActiveX控件，使读者在学习完本书后能够编写出较完整的Windows应用程序。书中包含了大量的典型算法的分析及示例，所有示例均通过调试，可以在Visual Basic环境下直接运行。例题尽量做到既能说明有关概念，又具有一定的实际意义，以激发学生的学习兴趣。各章之后配有大量的上机练习题，使学生能够通过上机实践掌握所学内容，提高动手能力和编程技能。最后一章“软件开发基础”使读者在学习完Visual Basic程序设计语言之后，对使用系统的方法进行程序设计及软件的开发过程有一个初步的认识。

与本教材配套出版的习题集包含了大量的不同题型的练习题，同时附有参考答案，有利于学生在课外进行自主练习，巩固所学的知识。

为满足广大教师的需要，本教材同时提供有配套的电子教案、教材中的所有例题的源程序以及教材各章之后的上机题答案。

由于作者水平有限，书中错误或不足之处在所难免，敬请读者批评指正。

邱李华

2005.4

目 录

第1章 程序设计基础

使用计算机时，要让计算机能按人的规定完成一系列的工作，就要求计算机具备理解并执行人们给出的各种指令的能力。因此在人和计算机之间就需要一种二者都能识别的特定的语言，这种特定的语言就是计算机语言，也叫程序设计语言，它是人和计算机沟通的桥梁。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“文章”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多个阶段，程序设计方法也得到不断发展和提高。

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。毋庸置疑，人们总是希望设计出来的语言好用，因此，计算机语言也经历了由低级向高级发展的历程。

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和面向对象的。

1.1.1 机器语言

从本质上说，计算机只能识别“0”和“1”，因此，计算机能够直接识别的指令是由一连串的0和1组合起来的二进制编码，称为机器指令，每一条指令规定了计算机要完成的某个操作。机器语言则是指计算机能够直接识别的指令的集合，它是最早出现的计算机语言。

例如，表1-1的机器指令用来完成一个简单的加法运算：9 + 8。

表1-1 机器语言程序举例

指令序号	机器指令	指令功能
1	10110000 00001001	把加数9送到累加器AL中
2	00000100 00001000	把累加器AL中的内容与另一数相加，结果存在累加器AL中（即完成9 + 8运算）
3	11110100	停止操作

表1-1的机器指令序列构成了一个简单的机器语言程序。可见，用机器语言编写的程序表现为一系列二进制信息，编写起来非常繁琐，可以用“难学、难记、难写、难检查、难调试”来加以概括，尤其是用这种语言编写的程序完全依赖于机器，所以程序的可移植性也很差。其优点是机器能直接识别、执行效率高，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。表1-2列出了用汇编语言来实现9 + 8的有关指令。

表1-2 汇编语言程序举例

语句序号	汇编语言指令	指令功能
1	MOV AL, 9	把加数9送到累加器AL中
2	ADD AL, 8	把累加器AL中的内容与另一数相加，结果存在累加器AL中（即完成9 + 8运算）
3	HLT	停止操作

表1-2的三条指令构成了完成9 + 8加法运算的汇编语言程序。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，所以又称为助记符。用这些助记符编出的程序，可读性好，容易查错，修改方便；但机器不能直接识别。为了解决这个问题，可以建立一个“符号与指令代码”对照表，在执行用汇编语言编写的程序之前，首先使用该“对照表”对每个助记符逐个扫描，把它转换为对应的机器语言程序。这个转换工作由一个叫做“汇编程序”的语言处理程序来完成，翻译出的程序叫做“目标程序”，而翻译前的程序称为“源程序”。虽然目标程序已经是二进制形式，但它还不能直接执行，需要使用连接程序把目标程序与库文件或其他目标程序（如别人已经编写好的程序段）连接在一起，才能形成计算机可以执行的程序（可执行程序），如图1-1所示。

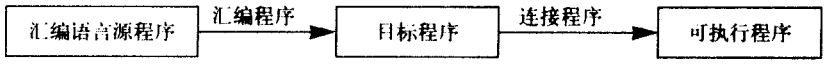


图1-1 汇编程序的作用

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仿，比高级语言快得多，所以直到现在仍广泛应用于实时控制、实时处理领域。

1.1.3 高级语言

1. 什么是高级语言

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，必须从两方面下功夫：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。在长期实践的基础上，在20世纪50年代末终于创造出独立于机型的、表达方式接近于被描述问题的、容易学习使用的高级语言。使用这些语言编写程序时，程序设计者可以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。高级语言的出现是计算机技术发展的里程碑，它大大提高了编程的效率，使人们能够设计出功能越来越强的程序。

高级语言较之汇编语言更接近于自然语言，描述计算公式与数学上的表示大体一致。例如，前面计算9 + 8的问题，若用BASIC、C语言或Visual Basic语言编程，就变得十分简单，而且易于理解，见表1-3。

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、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。程序设计方法将在1.2节介绍。

表1-3 高级语言程序举例

BASIC语言程序	C语言程序	Visual Basic语言程序

S = 9 + 8
END
	main()
	{
	int s;
	s = 9 + 8;
	}
		Private Sub Form_Load()
		s = 9 + 8
		End Sub

2. 高级语言处理程序——翻译程序

由于高级语言比较接近自然语言，当然就远离了机器语言，计算机不能直接识别，因此用高级语言编写的源程序，必须由一个承担翻译工作的处理程序，把高级语言源程序翻译成机器能识别的目标程序，这个处理程序称为翻译程序。每种高级语言都有自己的翻译程序，互相不能够代替。

翻译程序的主要职能是：

- 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。

3. 翻译程序的两种工作方式

翻译程序有两种工作方式：一种是解释方式，另一种是编译方式。

解释方式的翻译工作由“解释程序”来完成，这种方式好比口译方式，解释程序对源程序一条语句一条语句地解释执行，不产生目标程序。程序执行时，解释程序随同源程序一起参加运行，如图1-2所示。解释方式执行速度慢，但可以进行人机对话。在程序的执行过程中，编程人员可以随时发现程序执行过程中的错误，并及时修改源程序。这种方式对初学者来说非常方便。例如BASIC语言多数采用解释方式。

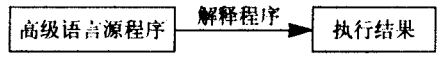


图1-2 解释方式示意图

编译方式的翻译工作由“编译程序”来完成。这种方式好比笔译方式，编译程序对源程序经过编译处理后，产生一个与源程序等价的“目标程序”，因为在目标程序中还可能要调用一些内部函数、内部过程、外部函数或外部过程等，所有这些程序还没有连接成一个整体，因此，这时产生的目标程序还无法运行，需要使用“连接程序”将目标程序和有关的函数库、过程库组装在一起，才能形成一个完整的“可执行程序”。产生的可执行程序可以脱离编译程序和源程序独立存在并反复使用。编译方式如图1-3所示。

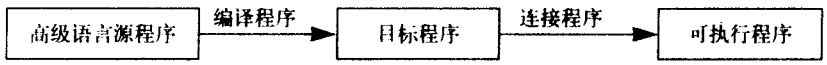


图1-3 编译方式示意图

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

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，等等，更有一些语言同时具备多种功能。从应用的角度，我们难以对程序设计语言作严格分类。

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地变化，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——程序设计方法。

1.2 程序设计

程序用程序设计语言编写，用于完成特定的任务。计算机所以能够自动地、有条不紊地工作，正是因为计算机能够按照程序所规定的操作步骤一步一步地执行相应的操作命令。程序是软件中最重要的成份，计算机工作离不开程序。程序应具有下列特性：

- 目的性：程序总是有明确的目的，是为解决某种特定的问题而设计的。
- 分步性：程序总是分成若干操作步骤，逐步解决问题。
- 有限性：程序所确定的操作步骤总是有限的，否则计算机就无法实现。
- 有序性：操作步骤必须是有先后次序的，否则就失去了程序设计的意义。
- 分支性：程序可以根据条件的不同，决定实施不同的操作步骤来解决问题。

编制程序的工作称为“程序设计”。为了有效地进行程序设计，至少应当具备两个方面的知识：一是要掌握一门或一门以上的高级语言；另一个是要掌握解题的方法和步骤，也就是说，在遇到一个需要求解的问题后，怎样将它分解成一系列的计算机可以实现的操作步骤，这就是“算法”（Algorithm）需要研究的问题。可以说，程序设计的灵魂是算法，而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语言编写程序，使计算机进行工作，得出正确的结果。因此本书在正式介绍Visual Basic语言之前，先介绍如何设计算法和表示算法。

1.2.1 算法

1. 什么是算法

在日常生活中做任何一件事情，都要按照一定规则，一步一步地进行。比如计算一个算术式，要先乘除后加减，这种规则实际上就是算法。厨师炒菜的操作步骤也是算法。在这里我们只讨论计算机算法，即计算机为解决一个问题而采取的方法和步骤，或者说是解题步骤的描述。

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、图形图像处理、信息的排序、分类、查找等算法。

2. 算法的特性

算法应具有以下特性。

- 有穷性：算法中执行的步骤总是有限次数的，不能无休止地执行下去。
- 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0到若干个输入：算法常需要对数据进行处理，因此，算法常需要数据输入。
- 有1到若干个输出：算法的目的是用来解决一个给定的问题，因此，它应向人们提供算法产生的结果。

3. 算法的表示形式

为了表示一个算法，可以采用不同的形式。

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，因此用自然语言表示一个算法便

于理解。

例如：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。

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，而必须采取间接交换的方法。因此，引入一个中间变量Z。

算法表示如下：

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

用自然语言表示算法，虽然容易表达，但文字冗长，而且往往不严格。同一段文字，不同的人会有不同的理解，容易产生“二义性”。因此，除了很简单的问题外，一般不用自然语言表示算法。

2) 用流程图表示算法：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的。用图来表示算法，直观、形象、容易理解。

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：

起止框：表示流程开始或结束

输入/输出框：表示输入或输出

处理框：表示基本处理功能的描述

判断框：根据条件是否满足，在几个可以选择的路径中，选择某一路径

流程线：表示流程的路径和方向

连接点：表示流程图中向/或来自其他地点的输出或输入

图1-4为交换两个变量的传统流程图。

这种传统流程图虽然形象直观，但对流程线的使用没做限制。使用者可以毫不受限制地使流程随意地转移，流程可能变得毫无规律，难以阅读和维护。

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结构化程序设计。

例如，用N-S流程图表示交换两个变量的值的算法如图1-5所示。

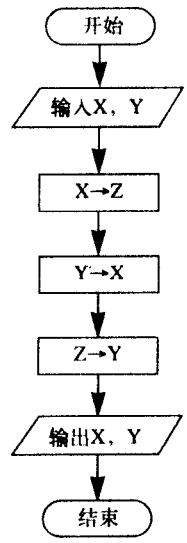


图1-4 交换两个变量的传统流程图

1.2.2 结构化程序设计

在一些高级语言中设置了无条件转移语句，当程序执行到此语句时，就会无条件地转移到某条语句去执行。对于编制一些小程序来说，无条件

转移语句使用起来很方便，可以转到程序的任意位置去执行；但是，在长期的程序设计实践中人们发现，当设计的程序较大，而且无条件转移语句稍多时，就会给程序的阅读、修改、维护带来很多的麻烦。任意地转移会使程序设计思路显得非常没有条理性且难以理解。于是，人们设想，能否使用一些基本的结构来设计程序，无论多么复杂的程序，都可以使用这些基本结构按一定的顺序组合起来。这些基本结构的特点就是只有一个入口、一个出口。由这些基本结构组成的程序就避免了任意转移、阅读起来需要来回寻找的问题。这就是结构化程序设计的基本思路。

1. 三种基本结构

在1966年，Bohra和Jacopini提出了三种基本结构，认为算法和程序都可以由这三种基本结构组成。这三种基本结构是：顺序结构、选择结构和循环结构。

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。图1-6是分别用传统流程图和N-S流程图表示的顺序结构（图1-6a的虚线框内是一个顺序结构），其中，A和B表示要操作的内容。计算机先执行A操作，再执行B操作。

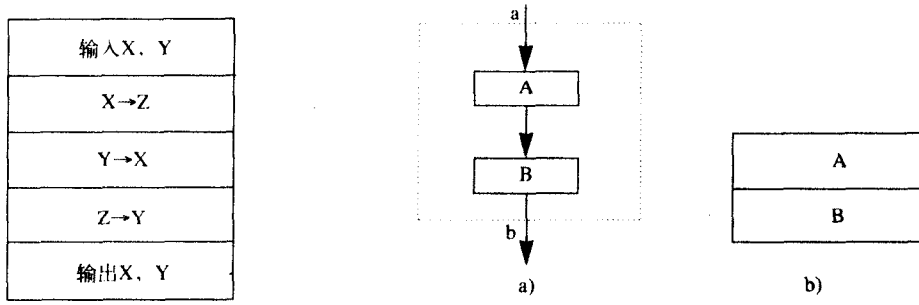


图1-5 交换两个变量的N-S流程图　　图1-6 顺序结构

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1-7是分别用传统流程图和N-S流程图表示的选择结构（图1-7a的虚线框内是一个选择结构）。这种结构中包含一个判断框，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1-7a可以看出，无论执行哪一个分支路径都通过汇合点b。b点是该基本结构的出口点。

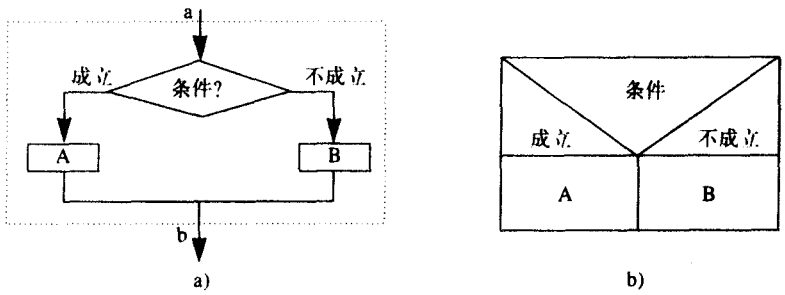


图1-7 选择结构

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，就必须使循环能够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：图1-8用两种流程图表示了当型循环结构。以图1-8a为例，当型循环的执行过程是：当程序运行到a点，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。

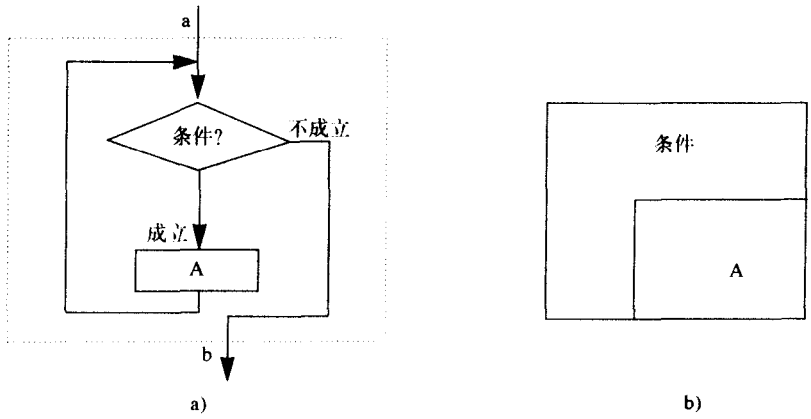


图1-8 当型循环结构

- 直到型循环结构：图1-9用两种流程图表示了直到型循环结构。以图1-9a为例，直到型循环的执行过程是：当程序运行到a点，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，如果条件不成立，则继续执行A操作；再判断条件是否成立，若仍然不成立，再执行A操作。如此反复执行，直到某次条件成立时为止，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。

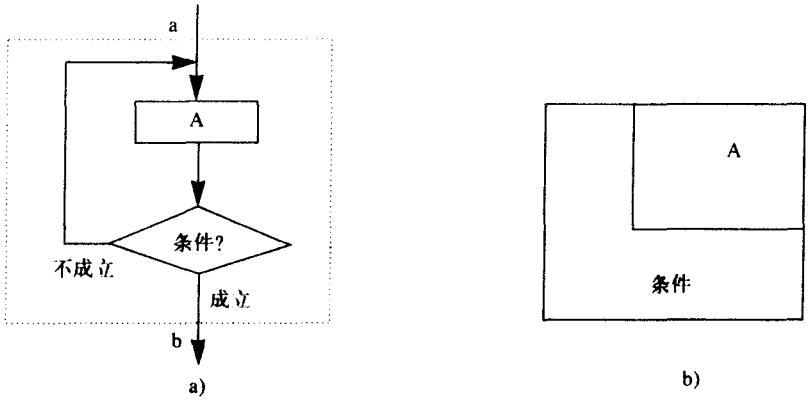


图1-9 直到型循环结构

以上三种基本结构有以下共同特点：

- 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。

2. 结构化程序设计方法

结构化程序设计方法要求把程序的结构规定为顺序、选择和循环三种基本结构，并提出了自顶向下、逐步求精、模块化程序设计等设计原则。结构化程序设计是把模块分割方法作为对大型系统进行分析的手段，使其最终转化为上述三种基本结构，其目的是为了解决由许多人共同开发大型软件时，如何高效率地完成高可靠系统的问题。程序的可读性好、可维护性好成为评价程序质量的首要条件。

结构化程序设计方法虽已得到了广泛的使用，但其存在的主要问题仍未得到很好的解决，即该方法实现中只突出了实现功能的操作方法，而被操作的数据处于实现功能的从属地位，使程序模块和数据结构松散地耦合在一起。因此，当程序复杂时，这种方法容易出错，难以维护。随着多媒体信息引入计算机，此矛盾更加突出。

由于上述缺陷已不能满足现代软件开发的要求，一种全新的软件开发技术应运而生，这就是面向对象的程序设计方法（Object Oriented Programming，OOP）。

1.2.3 面向对象的程序设计

面向对象的程序设计方法是一种系统化的程序设计方法，它允许抽象化、模块化的分层结构，具有多态、继承、抽象和封装等特性。面向对象的程序设计不同于标准的过程化程序设计，程序设计人员在进行面向对象的程序设计时，不再是单纯地从代码的第一行一直编写到最后一行，而是考虑如何创建对象，利用对象来简化程序设计，提高代码的可重用性，控制软件维护的复杂性和开销。

1. 对象（Object）

在自然界中，“对象”随处可见，大到整个宇宙及我们生活的地球，小到细胞、分子与原子，世间万物都可以是对象，如公司、雇员、时间表、房屋、人、汽车等，对象描述了某一实体。

在计算机中，将数据和处理该数据的过程、函数或子过程打包在一起而生成的新的数据类型称为对象，它是代码和数据的组合，可以作为一个单位来处理。对象可以是窗体、模块、数据库和控件等，也可以是整个应用程序。

对象有它的“属性”，属性是描述对象的数据。例如，对于某个人，有名字、职务和住址等属性；对于某辆汽车，有型号、颜色、各种性能指标等属性。

对象还可以做事情，如某个人可以建造楼房，汽车可以运载货物等。对象所能做的事情称为对象的“方法”。在程序设计中，方法是与对象相关联的过程，它紧密地和对象连接在一起，其内容是不可见的，编程人员只能使用它。

每个对象都可以对外界的动作进行识别和响应，事件即是一种预先定义好的特定动作，由用户或系统激活。在大多数情况下，事件是通过用户的交互操作产生的。例如，对汽车可以施行启动、停车、加速等操作。在程序设计中，可以触发事件的用户动作有鼠标单击、鼠标双击、鼠标移动等等。每个对象都有一组定义在其上的事件集合。

在过程化的应用程序中，应用程序自身控制了执行哪一部分代码和按什么顺序执行代码。从第一行代码开始并按应用程序中预定的路径执行程序，必要时调用过程。而现在普遍使用的Windows操作系统则属于“事件驱动型”的。在事件驱动的应用程序中，代码不是按照预定的路径执行的，而是在响应不同的事件时执行不同的代码片段。在没有事件发生时，整个程序处于停滞状态。应用程序代码执行的顺序由时间发生的先后顺序决定，因此应用程序每次运行时所经过的代码路径是不同的。事件可以由用户操作触发，也可以由来自操作系统或其他应用程序的消息触发，甚至由应用程序本身的消息触发。

2. 面向对象（Object Oriented，OO）

面向对象基本上意味着从问题所涉及的对象入手来研究该问题，面向对象的概念用在许多行业