普通高等教育“十一五”国家级规划教材

HZ BOOKS
华章教育

Visual Basic 程序设计教程

第3版

本书配有
电子教案

邱李华 曹青 郭志强 等编著

机械工业出版社
China Machine Press

普通高等教育“十一五”国家级规划教材

Visual Basic 程序设计教程

第3版

邱李华 曹青 郭志强 等编著

机械工业出版社
China Machine Press

本书是普通高等教育“十一五”国家级规划教材。全书以Visual Basic 6.0为语言背景，结合大量的实例，深入浅出地介绍了程序设计的基本概念和基础知识、Visual Basic 6.0的集成开发环境、结构化程序的三种基本结构、数组、过程、Visual Basic常用控件、界面设计、图形设计、文件、数据库基础和软件开发基础。

本书概念叙述严谨、清晰，内容循序渐进、深入浅出，示例丰富，趣味性和实用性强，包含大量常见算法，并配有大量的上机练习题，在注重程序设计基本概念和基础知识介绍的同时，重在强调程序设计能力的培养，配套的习题集提供了大量多种题型的练习题并附有参考答案。

本书可作为高等学校或培训机构计算机程序设计基础课程的教材，也可作为Visual Basic程序设计语言的自学用书或参加计算机等级考试的参考用书。

图书在版编目（CIP）数据

Visual Basic程序设计教程　第3版/邱李华等编著. —北京：机械工业出版社，2011.3
（普通高等教育“十一五”国家级规划教材）

ISBN 978-7-111-33368-5

Ⅰ. V…　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（2011）第020352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）
责任编辑：邹朝怡
北京诚信伟业印刷有限公司印刷
2011年5月第3版第1次印刷
185mm × 260mm · 20.75印张
标准书号：ISBN 978-7-111-33368-5
定价：33.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

前　言

Visual Basic自问世以来，一直是深受欢迎的程序设计语言，其简练的语法、强大的功能、结构化程序设计思想、方便快捷的可视化编程手段和事件驱动的编程机制，使得编写Windows环境下的应用程序变得非常容易，因此，Visual Basic目前已经成为许多高等院校首选的教学用程序设计语言，也是全国计算机等级考试的程序设计语言之一。

2002年1月，我们出版了《Visual Basic程序设计教程》及配套的习题集，该教材完全由从事Visual Basic课程教学的一线教师编写，凝聚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。

2006年9月，教育部高等学校计算机科学与技术教学指导委员会正式制定了《关于进一步加强高等学校计算机基础教学的意见暨计算机基础课程教学基本要求（试行）》（以下简称“要求”），该“要求”对计算机程序设计基础课程教学提出了“一般要求”和“较高要求”，在充分领会“要求”精神的基础上，我们对原教材进行了修订，形成了第2版。第2版涵盖了“要求”中有关“Visual Basic程序设计”的“一般要求”和“较高要求”提出的所有内容，给不同办学层次的学校或不同专业提供了选择余地。第2版被许多院校选为教材，深受广大师生的欢迎，是普通高等教育“十一五”国家级规划教材。

经过3年多的教学实践，我们对第2版进行了进一步修订，形成了第3版，配套的习题集以及教学辅助材料也进一步丰富和完善。修订后的第3版秉承了第2版的特点，注重对学生基本概念、基本理论、基本技能的培养，条理清晰、深入浅出、示例丰富。各章后的上机练习题知识点覆盖全面，配套出版的习题集紧密结合教材编写，包含了大量各种题型的练习题，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。

和第2版相比，第3版在以下几方面进行了改进和提高：

1）语言更加精练：在文字叙述上更加准确和精练，基本概念的介绍更加精简。

2）例题更加丰富、有趣和实用：书中增加了具有一定趣味性或实用性的例题，以避免学生在刚开始学习Visual Basic程序设计语言时觉得内容抽象、基本概念枯燥，让学习过程更加具有趣味性和成就感。

3）在示例的介绍上进一步清晰了解题步骤：多数示例按“界面设计→代码设计→运行效果”的思路进行介绍，使设计过程更加清晰。

4）更丰富的练习题：各章后的上机练习题进一步丰富，配套习题集中的练习题也进一步丰富和完善。

5）前面章节就将一些界面的设计方法、控件和绘图方法等引入到示例中，既增加了例题和练习题的趣味性，也使学生在较短的时间内尽可能了解更多的知识，这样，在界面设计、常用控件、绘图等章节的学习中就可以用较短的时间开展教学，也利于引导学生自学。

6）对过时的内容进行了更新。

7）增加了部分上机练习题的视频演示：前8章的部分上机练习题的设计过程已录制成视频文件，教师可以将其下发给学生，作为实验指导。该视频演示具有以下特点：

- 操作直观，容易理解。
- 视频文件以swf文件的形式给出，文件小，播放方便，学生可以重复观看并得到及时指导。
- 对于一些简单操作，视频中直接给出了操作过程和文字注释，让学生在模仿过程中逐渐记住这些操作，把教师花在学生身上不断重复辅导一些简单问题的时间节省了下来，以重点帮助学生解决一些更复杂或更关键的问题；对于一些当前章节需要重点考察的知识点，尤其是代码编写部分，视频中

会留出一些空白，通过文字提示、提出问题、给出主要思路、提示阅读有关示例等方法来引导学生思考并独立完成，避免学生一味地模仿和对视频演示产生依赖性。

- 整个视频录制过程依照练习内容的先后顺序，给出的操作步骤及代码遵循从详细到简化的原则，逐渐减少依赖性、加大难度，让学生从模仿逐步过渡到独立思考，直至能独立设计一些模块。
- 该视频特别注意调整界面的布局、代码的缩进等，有助于培养学生良好的编程习惯。
- 对于有运行效果的题目，即使在设计界面或代码时跳过了某些部分，最后也都会给出运行效果，让学生可以将自己的设计和运行效果进行对比，检查设计的正确性。

本书约定：使用符号“|”来分隔多级菜单操作。例如，使用“格式”菜单下“对齐”子菜单下的“左对齐”命令，在书中描述为：使用“格式|对齐|左对齐”命令。

本书第1～4章由曹青编写，第5～8章由邱李华编写，第9～11章由郭志强编写，第12～14章由刘春贵（山西大同大学）编写。

为满足广大教师的教学需要，本书免费向教师提供配套的电子教案、教材中所有示例的源程序、教材各章后的上机练习题参考答案以及部分上机练习题视频演示，需要的教师可登录华章网站www.hzbook.com下载。

由于编者水平有限，书中难免存在不足或疏漏之处，恳请读者批评指正，帮助我们不断改进和完善。

邱李华

2010年10月

教学建议

依据附录中给出的“计算机程序设计基础”课程教学基本要求，这里给出总学时数为96学时（48学时讲课+48学时上机）的课程安排建议，教师可根据本校的实际教学大纲要求和学生的基础及后续课程的安排等进行相应的调整。例如，对于在“大学计算机基础”课程中学习过“程序设计基础”的学生，可以不讲授第1章；对于开设“数据库基础（技术）及应用”课程的专业，可以不讲授第13章；在课程总学时数较少的情况下，可以在前面的章节中结合示例引入第9～11章的内容，并通过引导学生自学、自行上机练习、课外辅导等方式完成这几章的学习。

内　容	讲课	上机	小计
第1章　程序设计基础	2		2
第2章　Visual Basic简介	4	4	8
第3章　Visual Basic程序设计代码基础	4	4	8
第4章　顺序结构程序设计	3	4	7
第5章　选择结构程序设计	3	4	7
第6章　循环结构程序设计	4	6	10
第7章　数组	4	4	8
第8章　过程	6	4	10
第9章　Visual Basic常用控件	4	4	8
第10章　界面设计	2	2	4
第11章　图形设计	2	2	4
第12章　文件	4	4	8
第13章　数据库	4	4	8
第14章　软件开发基础	2	2	4
总　计	48	48	96

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有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语言和面向对象的语言。

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。每一条机器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所示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中的内容与另一个数8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所示的机器指令序列构成了一个简单的机器语言程序。可以看出，用机器语言编写的程序由一系列二进制信息组成，编写起来非常烦琐，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二进制指令。用这样一些符号代替二进制指令所产生的语言称为汇编语言，也称符号语言。表1-2列出了用汇编语言来实现求9+8和的有关指令。

表1-2所示的三条指令构成了完成9+8运算的汇编语言程序。可以看出，在该程序中，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”。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。

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中的内容与另一个数8相加，结果存在累加器AL中（即完成9+8运算）
3	HLT	停止操作

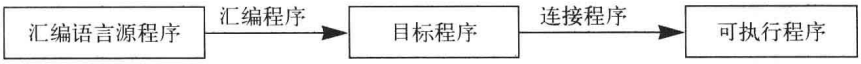


图1-1 汇编程序的作用

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

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。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强大的程序。

随着计算机技术的发展，不同风格的高级语言不断出现。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；而较晚出现的Visual Basic、Visual C++、Delphi、Java等适用于Windows环境的高级语言，采用的是面向对象的程序设计方法。面向过程的语言致力于用计算机能够理解的逻辑来描述需要解决的问题以及解决问题的具体方法和步骤；面向对象的语言则站在更高、更抽象的层次上来解决问题，将客观事物抽象为一系列的对象，程序的执行是靠在对象间传递消息来完成的。面向对象的语言通过继承与多态可以很方便地实现代码的重用，已经成为当前流行的一类程序设计语言。

由于高级语言比较接近于自然语言，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，不能够互相代替。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，而直接产生执行结果，如图1-2所示。由于解释方式边解释边执行，因此执行速度慢，但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，早期的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。编译方式的工作过程如图1-3所示。

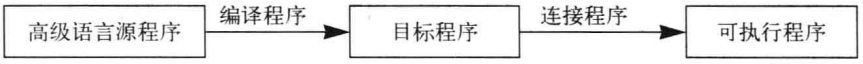


图1-3 编译方式示意图

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，Visual Basic 6.0既可以在解释方式下工作，也可以在编译方式下工作。

为了适应不同应用领域的需要，程序设计语言各具特点。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。许多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做严格的分类。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——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部分，计算机工作离不开程序。

程序设计就是使用某种程序设计语言编写一些代码来驱动计算机完成特定功能的过程。为了有效地进行程序设计，至少应当具备两个方面的知识：一是要掌握解题的方法和步骤；另一个是要掌握一种或一种以上的高级语言，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”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，先简要介绍算法的概念和算法的表示。

1.2.1 算法

1. 什么是算法

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”；期末考试前的复习计划就是“复习算法”；到医院看病，先挂号，再问诊、检查、诊断，然后取药等，这就是“看病算法”。

在这里我们只讨论计算机算法，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。

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、查找等的算法。

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明确，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个到若干个输入：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个到若干个输出：算法的目的是用来解决一个给定的问题，因此，它应向人们提供解题的结果。

3. 算法的表示形式

描述算法的方法有多种，常用的有自然语言、流程图、伪代码和PAD图等，其中最普遍的是流程图。下面简要介绍用自然语言和用流程图表示算法。

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，因此用自然语言表示的算法较容易理解。

例如：将两个变量X和Y的值互换。

交换存在直接交换和间接交换两种方式。比如，两个人交换座位，只要各自去坐对方的座位就行了，这种交换就是直接交换。再比如，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（醋）倒入空瓶子，再把醋（酒）倒入已倒空的酒瓶子，最后把酒（醋）倒入已倒空的醋（酒）瓶子，这样才能实现酒和醋的交换，这种交换就是间接交换。

计算机中交换两个变量的值不能用直接交换的方法，而必须采取间接交换的方法，因此，需要引入一个中间变量Z。用自然语言表示该算法，可以描述为：

步骤1 将X的值存入中间变量Z中：X→Z

步骤2 将Y的值存入变量X中：Y→X

步骤3 将中间变量Z的值存入变量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。流程图用一些图框、流程线以及文字说明来描述操作过程。用流程图表示算法，直观、形象、容易理解。

传统流程图采用了美国国家标准化协会ANSI（American National Standard Institute）规定的一些符号，常见的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：表示流程图中向其他地点或来自其他地点的输出或输入

图1-4所示为交换两个变量的传统流程图。

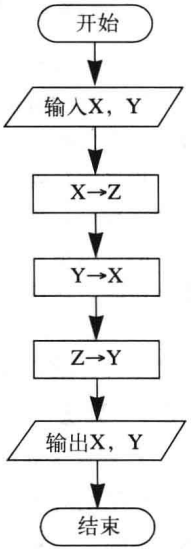


图1-4 交换两个变量的传统流程图

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。这种流程图特别适用于结构化程序设计。

例如，用N-S流程图表示交换两个变量X和Y的值的算法如图1-5所示。

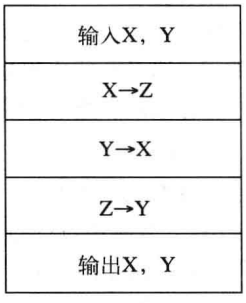


图1-5 交换两个变量的N-S流程图

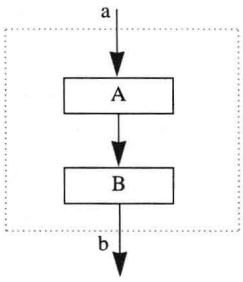
1.2.2 结构化程序设计

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都是只有一个入口、一个出口。由这些基本结构组成的程序就避免了任意转移、难以阅读的问题。这就是结构化程序设计的基本思路。

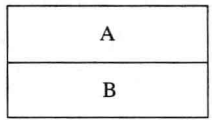
1. 三种基本结构

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：顺序结构是一种最简单的基本结构，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所示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所示的虚线框内是一个顺序结构，其中，A和B表示操作步骤。计算机先执行A操作，再执行B操作。



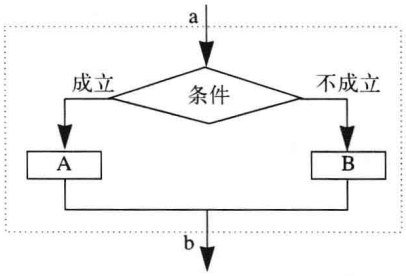
a) 用传统流程图表示顺序结构



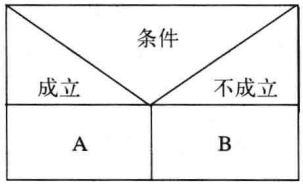
b) 用N-S流程图表示顺序结构

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所示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所示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成立，从两个分支路径中选择其中的一个执行。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。



a) 用传统流程图表示选择结构



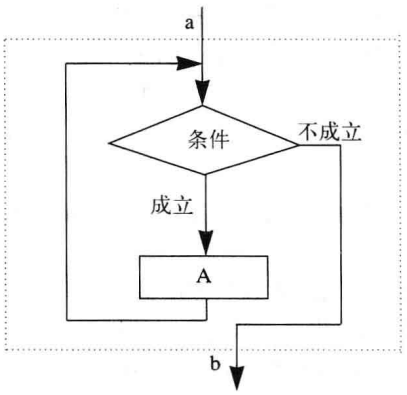
b) 用N-S流程图表示选择结构

图1-7 选择结构

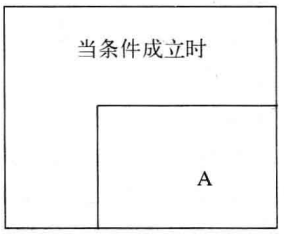
3）循环结构：循环结构用于规定重复执行一些相同或相似的操作。要使计算机能够正确地完成循环操作，就必须使循环能够在执行有限次后退出，因此，循环的执行要在一定的条件下进行。根据对条件的判断位置不同，可以有两类循环结构：当型循环和直到型循环。

①当型循环结构：图1-8所示是用两种流程图表示当型循环结构。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点进入当型循环时，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。



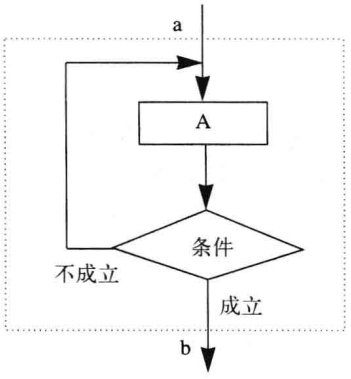
a) 用传统流程图表示当型循环



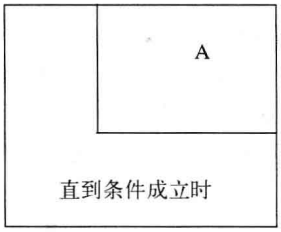
b) 用N-S流程图表示当型循环

图1-8 当型循环结构

②直到型循环结构：图1-9所示是用两种流程图表示直到型循环结构。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点进入直到型循环时，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。



a) 用传统流程图表示直到型循环



b) 用N-S流程图表示直到型循环

图1-9 直到型循环结构

以上三种基本结构具有以下共同特点：

- 只有一个入口，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

结构化程序设计的基本思想是把程序的结构规定为顺序结构、选择结构和循环结构三种基本结构，采

用自顶向下、逐步求精、模块化等程序设计原则。具体地说，就是首先从问题本身开始，找出解决问题的基本思路，并将其用结构化流程图表示出来，这个流程图可能是非常粗糙的，仅仅是一个算法的轮廓，但可以作为进一步分析的基础；接下来就应该对流程图中那些比较抽象的、用文字描述的模块做进一步的分析细化，每次细化的结果仍用结构化流程图表示；最后，对如何求解问题的所有细节都弄清楚了，就可以根据这些流程图直接写出相应的程序代码了。在分析的过程中用结构化流程图表示解题思路的优点是：流程图中的每一个程序模块与其他程序模块之间的关系非常简明，因为每个模块都具有“单入口单出口”的控制结构。每次可以只集中精力分解其中的一个模块而不影响整个程序的结构。据此就可以很容易地编写出结构良好、易于调试和维护的程序来。

我们知道，所有的程序都由两类元素组成，即代码和数据。代码规定了要进行的操作，数据决定了要操作的对象。多年来，人们设计程序所探索的主要问题一直是如何把代码和数据有效地结合起来。但早期的程序设计（包括结构化程序设计）方法一般只突出了实现功能的操作方法，其中心思想是用计算机能够理解的逻辑来描述和表达待解决的问题及其具体的解决流程，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。使用这种方法编写的程序在执行时是线性的，这种方法被称为面向过程的编程，像早期出现的BASIC语言、FORTRAN语言、Pascal语言、C语言等都是面向过程的编程语言。使用面向过程的编程语言编写小程序比较有效。然而，随着程序规模与复杂性的增长，程序中的数据结构变得与数据的操作同样重要。在大型结构化程序中，一个数据结构可能被许多过程处理，修改此数据结构将影响到所有这些过程。在由许多过程组成的程序中，这将带来极大的麻烦并且容易产生错误，甚至失去对代码的有效控制。

由于上述缺陷，面向过程的编程思想已不能满足现代化软件开发的要求，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

面向对象程序设计是一种新兴的程序设计方法，该方法建立在结构化程序设计基础上，最重要的改变是程序围绕被操作的数据来设计，而不是围绕操作本身。

面向对象编程的一个实质性要素是抽象。人们通过抽象来处理编程过程中遇到的复杂问题。例如，当看到一辆汽车时，人们不会把它想象成由几万个相互独立的零件组成的一套动力装置，而是把整个汽车看成是一个具有自己独特行为（停止、启动、运行、加速、减速、换向等）的对象。这种抽象使人们很容易地将一辆汽车开到目的地，而不至于因为组成汽车的零件过于复杂而不知所措。汽车驾驶员可以忽略发动机的工作原理，可以忽略汽车引擎、传动及刹车系统的工作细节，而将汽车作为一个整体来加以利用。

任何现实问题都是由一些基本事物组成的，这些事物之间存在着一定的联系，在使用计算机解决现实问题的过程中，为了有效地反映客观世界，最好建立相应的概念去直接表现问题领域中的事物以及事物之间的相互联系，此外，还需要建立一套适应人们一般思维方式的描述模式。面向对象技术的基本原理正是：按问题领域的基本事物实现自然分割，按人们通常的思维方式建立问题领域的模型，设计尽可能直接自然表现问题求解的软件系统。为此，面向对象技术中引入了“对象”来表示事物；用消息传递建立事物间的联系；“类”和“继承”是适应人们一般思维方式的描述模型。

当然，面向对象的程序设计并不是要抛弃结构化程序设计方法，而是站在比结构化程序设计更高、更抽象的层次上去解决问题。当它分解为低级代码模块时，仍需要结构化编程技巧。结构化的分解突出过程，即如何做（How to do），它强调代码的功能是如何得以实现的。面向对象的分解突出真实世界和抽象的对象，即做什么（What to do），它将大量的工作由相应的对象来完成，程序员在程序设计中只需说明要求对象完成的任务。

面向对象的程序设计方法符合人们习惯的思维方法，便于分析复杂而多变的问题；易于软件的维护和功能的增减；能用继承的方式缩短程序开发的时间。

面向对象程序设计使用对象、消息、类、封装、继承等基本概念来进行程序设计。下面介绍这些基本概念。

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、函数等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。对象可以是窗口、模块、数据库和控件等，也可以是整个应用程序。

2. 面向对象（Object Oriented，OO）

面向对象主要是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中。例如，当设计一幢办公楼时，一个设计人员只需要考虑工作空间、框架和管道系统等对象的设计需求，而不必关心如何钉钉子、连接管道等较低层次的工作过程。

面向对象不仅是一种新的程序设计技术，而且是一种全新的设计和构造软件的思维方法。它使计算机解决问题的方式更加类似于人类的思维方式，更能直接地描述客观世界。从程序设计的角度看，面向对象代表了一种通过模仿人类建立现实世界模型的方法（包括概括、分类、抽象、归纳等）进行软件开发的思想体系。

Visual Basic为面向对象的开发提供了许多功能，可以实现面向对象的程序设计。

3. 消息（Message）

如何要求对象完成指定的操作？如何在对象之间进行联系呢？所有这一切都只能通过传递消息来实现。传递消息是对象与其外部世界相互关联的唯一途径。对象可以向其他对象发送消息以请求服务，也可以响应其他对象传来的消息，完成自身固有的某些操作，从而服务于其他对象。例如，直升机可以响应轮船的海难急救信号，起飞、加速、飞赴出事地点并实施救援作业。在面向对象程序设计中，程序的执行是靠在对象间传递消息来完成的。

一个对象可以接收不同形式、不同内容的多个消息；相同形式的消息可以送往不同的对象；不同的对象对于形式相同的消息可以有不同的解释，能够做出不同的反应。

4. 类（Class）

人们喜欢将事物分类，以发现事物中的相似性，并将它们相应地组合。将带有相似属性和行为的事物组合在一起，可以称为一个“类”。例如，所有的汽车构成了“汽车”类，所有的动物构成了“动物”类，而所有的人构成了“人”类。

一个类具有成员变量和成员方法。成员变量相当于“属性”，比如“人”具有的变量有胳膊、手、脚等。而成员方法是该类能完成的一些功能，比如“人”可以说话、行走等。

类中的一个具体对象称为类的实例。如果说类是一个抽象概念，那么类的实例就是一个具体对象。例如，在“汽车”类中，每一辆汽车都是一个具体的对象，是“汽车”类的一个实例。又如，我们说“人”就是一个抽象概念，是“人”类，但是具体到某个人，比如你、我、他，就是“人”的实例，是具体对象。

5. 封装（Encapsulation）

封装是指将对象的属性和方法包装在一起，从而包含和隐藏对象的内部信息（如内部数据结构和代码）的技术。通过封装对象的方法和属性，可以将操作对象的内部复杂性与应用程序的其他部分隔离开来，这样对象的内部实现（代码和数据）受到了保护，外界不能访问它们，只有对象中的代码才可以访问对象的内部数据。对象的内部数据结构的不可访问性称为数据隐藏。封装简化了程序员对对象的使用，程序员只需知道输入什么和输出什么，而对类内部进行何种操作不必追究。封装还使得一个对象可以像一个部件一样用在程序中，而不用担心对象的功能受到影响。就像电视一样，我们不需要知道它的内部是由哪些零件组成、如何工作的，所以把它们封装起来了，我们只需知道电视机上的那些按钮和插口的使用或如何用遥控器来控制电视机就可以了。