计算机基础课程系列教材

Visual Basic 程序设计教程

本书配有
电子教案

曹青　邱李华　郭志强　编著

机械工业出版社
China Machine Press

计算机基础课程系列教材

Visual Basic 程序设计教程

曹 青 邱李华 郭志强 编著

机械工业出版社
China Machine Press

本书作者在多年教学经验基础上，并根据学生的认知规律精心组织了本教材内容，并通过大量有现实意义的例题，深入浅出地介绍了VB程序设计的有关概念和编程技巧。书中例题都经过了仔细的调试，并配有大量上机实习题。

本教材循序渐进，知识结构及深度合理，非常适合大专院校本、专科，成人继续教育及自学人员使用。

本教材还配套出版了习题集，并为教师备有电子教案。

图书在版编目（CIP）数据

Visual Basic程序设计教程/曹青等编著. -北京：机械工业出版社，2002.2
（计算机基础课程系列教材）
ISBN 7-111-09593-6

Ⅰ. V… Ⅱ. 曹… Ⅲ. BASIC语言－程序设计－教材 Ⅳ. TP312

中国版本图书馆CIP数据核字（2002）第006271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：刘立卿
北京忠信诚印刷厂印刷·新华书店北京发行所发行
2002年2月第1版第1次印刷
787mm×1092mm 1/16·18.25印张
印数：0 001-5 000册
定价：26.00元

前　言

根据我国当前教学改革和建设的需要，国家教育部提出了计算机基础教学的基本目标，即计算机基础教学要达到目标中规定的三个层次的基本要求，其中，第一层次为计算机文化基础，第二层次为计算机技术基础，第三层次为计算机应用基础。计算机程序设计语言是高等院校各专业学生的一门基础课程，属于计算机技术基础教育，是当代大学生必须掌握的一种应用技能。随着Windows图形用户界面的普遍使用，讲授面向过程的程序设计语言已不能完全适应社会的需要，要求当代大学生具备使用当今流行的系统平台和开发工具构造应用系统的初步能力。

1991年Visual Basic语言的诞生，为人们开发图形用户界面的应用程序提供了有力的工具，它是近年来被广泛使用的一种高级语言。Visual Basic继承了BASIC语言简单易学的优点，又增加了许多新的功能，它采用当前最新的程序设计思想：面向对象与事件驱动，使编程变得更加方便、快捷。使用Visual Basic既可以开发个人或小组使用的小型工具，又可以开发多媒体软件、数据库应用程序、网络应用程序等大型软件。

本书是在作者这几年从事Visual Basic教学的基础上编写而成的，作者根据多年的教学经验和学生的认知规律精心组织教材内容，做到深入浅出、循序渐进。通过大量例题使读者能迅速掌握有关概念及一些编程技巧，书中的例题都经过了仔细的调试；书中同时配有大量的上机实习题，使读者能够通过上机实践掌握所学内容。

本教材配套出版的习题集，供读者课外巩固所学的概念。

本教材以应用为中心，以初学者为对象，以提高程序设计能力为宗旨，为读者使用Visual Basic开发Windows平台下的应用程序提供了捷径。

本教材适合于大专院校学生、成人继续教育、自学人员使用。

建议本教材授课时数32～48学时，另外还要安排大量的上机练习，以巩固所学知识。

本书的第1、2、3、4、9章由曹青编写；第5、6、7、8、10、13章由邱李华编写；第11、12章由郭志强编写。

由于计算机技术发展迅速，加上作者水平有限，书中难免存在缺点和错误，请读者不吝赐教。

编　者

2001年11月

*　　*　　*　　*

本教材为教师配有免费光盘，光盘中的电子教案制作成PowerPoint演示文稿，例题可直接在Visual Basic 6.0环境下执行。选用本教材的教师若有需要，请与机械工业出版社华章公司市场部联系。

电话：010-68995264

传真：010-68995260

Email：marketing@hzbook.com

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了不断地发展和提高。

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。勿庸质疑，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。

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就称为汇编语言，也称为符号语言。表1-2列出了用汇编语言来实现9+8运算的有关指令。

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”。如图1-1所示。

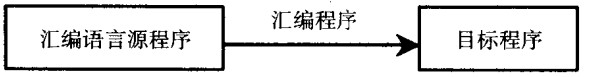


图1-1 汇编程序的作用

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、实时处理领域中。

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，它大大地提高了编程的效率，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 Baisc语言编程，就变得十分简单，而且易于理解，见表1-3。

表1-3 高级语言程序举例

BASIC语言程序	C语言程序	Visual Baisc语言程序

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

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、QuickBASIC、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环境的高级语言，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。

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。

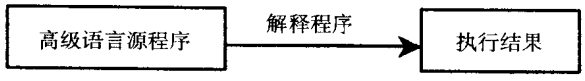


图1-2 解释方式示意图

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。

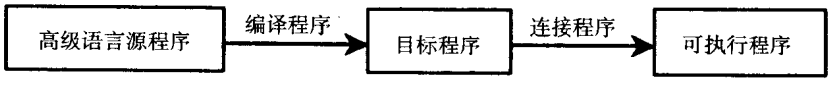


图1-3 编译方式示意图

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，功能也在不断更新、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的、规律性的东西——程序设计方法。

1.2 程序设计

程序由程序设计语言编写，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分，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，逐步解决问题的。
- 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，而语言只是形式。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，都是按照一定规则，一步一步地进行的。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

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，设一中间变量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

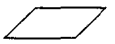
用自然语言表示算法，虽然容易表达，但文字冗长，而且往往不严格，同一段文字，不同的人会有不同的理解，容易产生“二义性”。因此，除了很简单的问题外，一般不用自然语言表示算法。

（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 

流向线：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。

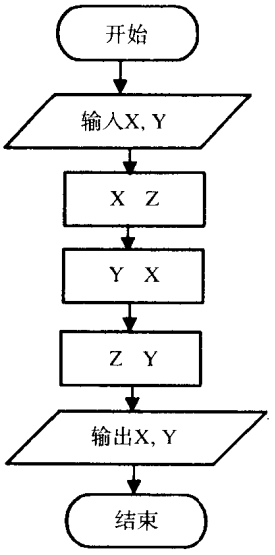


图1-4 交换两个变量流程图

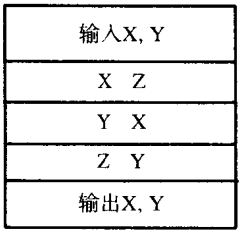


图1-5 交换变量

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、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

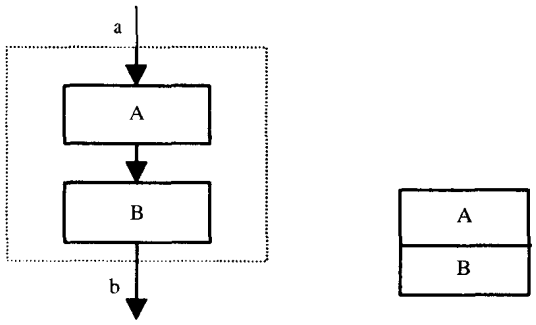
在1966年，Bohra和Jacopini提出了三种基本结构，认为算法和程序都可以由这三种基本结构组成。这三种基本结构是：顺序结构、选择结构和循环结构。

（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。

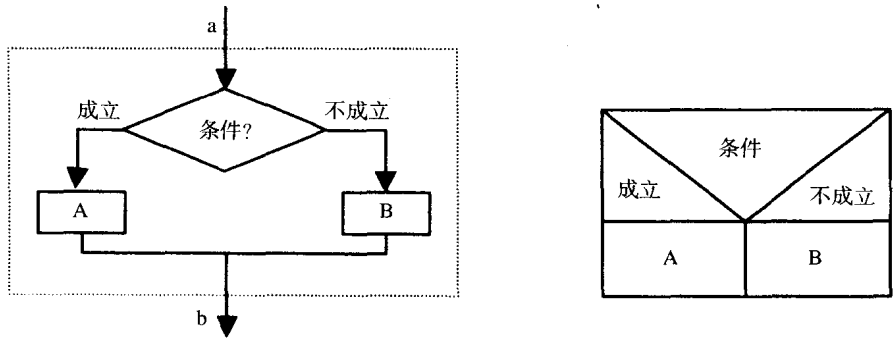
（2）选择结构

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点是本基本结构的出口点。



a) b)

图1-6 顺序结构

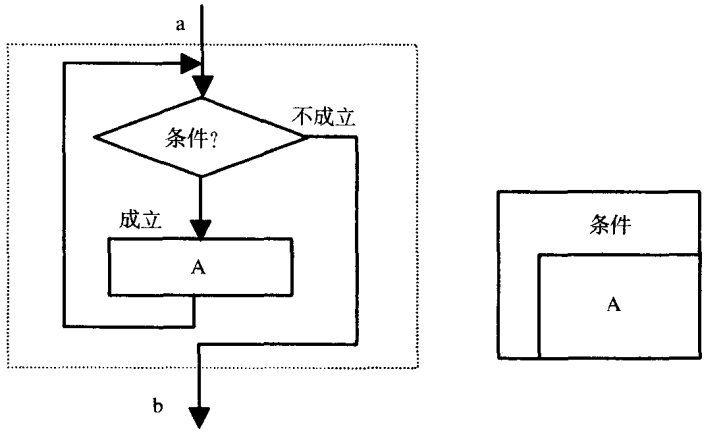


a) b)

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。



a) b)

图1-8 当型循环结构

- 当型循环结构：图1-8用两种流程图表示了当型循环的结构。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。
- 直到型循环结构：图1-9用两种流程图表示了直到型循环的结构。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。

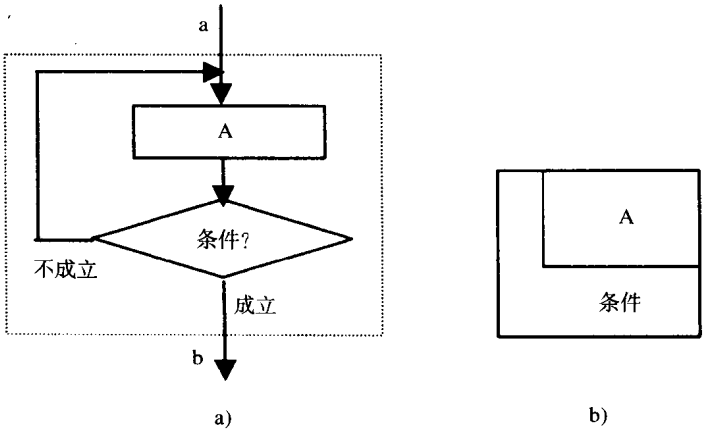


图1-9 直到型循环结构

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。

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，简称OOP）。

1.2.3 面向对象的程序设计

面向对象的程序设计在20世纪80年代初就提出来了，它起源于Smalltalk语言。这种方法引入了新的概念和思维方式，为使软件容易在程序设计中能够模仿建立真实世界模型的方法，面向对象的程序设计对系统的复杂性进行概括、抽象和分类，使软件的设计与实现形成一个由抽象到具体、由简单到复杂这样一个循序渐进的过程，从而解决大型软件研制中存在的效率低、质量难以保证、调试复杂、维护困难等一系列问题。

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；与可视化技术相结合，改善了工作界面。

在开始学习面向对象的程序设计语言之前，有必要先了解有关面向对象程序设计的一些基本概念。

（1）对象（(Object)

在自然界中，“对象”随处可见。对象用于描述某一实体，如公司、雇员、时间表、房屋、人、汽车等，对象有它的“属性”。例如，对于某个人，具有一个名字、职务和住址等属性；对于每辆汽车，都具有型号、颜色、各种性能指标等属性。对象还可以做事情——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”。对象可以是真实世界的事物，如刚才描述的人或汽车；对象也可以是概念性的事物，如工程进程或工资单。这些概念性的东西触摸不到，但它们也具有属性和方法。在计算机中，对象可以是窗体、模块、数据库和控件等。

（2）面向对象（Object Oriented，简称OO）

面向对象基本上意味着从该问题所涉及的对象入手来研究问题，面向对象的概念用在许多行业。例如，当设计一个办公室时，一个设计人员要考虑工作空间、基础、框架和管道系统，这些是真实世界的对象。设计人员不必关心打地基、钉钉子或连接管道等较低层次的工作过程，只需将精力集中在办公楼的设计等高层次工作上。Visual Basic为面向对象的开发提供了许多功能，可以用于实现面向对象的设计。

（3）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”。在面向对象的概念中，“类”用于指一组相似的对象。

一个属于某种类的特定对象称为该类的一个实例。类的每个实例具有已定义的一组属性值，可以执行已定义的行为。

Visual Basic工具箱中的命令按钮代表CommandButton类。每次向Visual Basic中的窗体添加命令按钮时，就在创建CommandButton类的一个实例。CommandButton类有特定的已定义方法和属性，如Move方法、Name属性、Caption属性。

（4）封装（Encapsulation）

将数据和操作数据的函数衔接在一起，构成一个具有类类型的对象的描述称为封装。封装要求一个对象应具备明确的功能，并有一个或几个接口以便和其他对象相互作用。同时，对象的内部实现（代码和数据）是受保护的，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，只需知道输入什么和输出什么，而对类内部进行什么操作不必追究。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。

（5）继承（Inheritance）

在面向对象的语言中，可以从一个类生成另一个类。派生类（也称子类）继承了其父类和祖先类的数据成员和成员函数。派生类代表父类的一种改良，因此增加了新的属性和新的操作。派生类一般通过声明新的数据成员和成员函数来增加新的功能。另外，派生类在继承的成员函数不适合时也可以弃之不用。

（6）多态性（Polymorphism）

多态性是一种面向对象的程序设计功能，当同样的消息被不同的对象接收时，却导致完全不同的行为，即完成不同的功能。消息指的是类成员函数的调用。

Visual Basic中的Move方法是多态性的一个示例。当Move方法被一个窗体对象执行时，该窗体知道如何将自身和它的全部内容移动到指定的坐标。当Move方法被一个按钮对象执行时，窗体不移动，而按钮移动到了正确的位置。同一命名的方法提供了多态性结果。

1.2.4 程序设计的步骤

介绍了算法的概念、算法的表现形式以及程序设计的方法之后，下面简单介绍完成一个正确的程序设计任务应采取的主要步骤。

1）分析问题：即分析任务的要求、要给出什么结果、提供什么资源、有无解决的可能。

2）建立数学模型：即用数学语言描述它。

3）选择计算方法：即选择用计算机求解该数学模型的近似方法。

4）算法设计：即制定出计算机运算的全部步骤。

5）编写程序：即选择一种计算机语言，根据前一步的算法编写程序。

6）运行、调试程序：即上机运行程序，用各种不同的数据测试在不同的情况下能否得到正确结果。

7）整理文档：即写出一份技术报告或程序说明书，其中应包括题目、任务要求、原始数据、数据结构、算法、程序清单、运行结果、所用计算机系统配置、操作说明等，以便作为资料交流或保存。