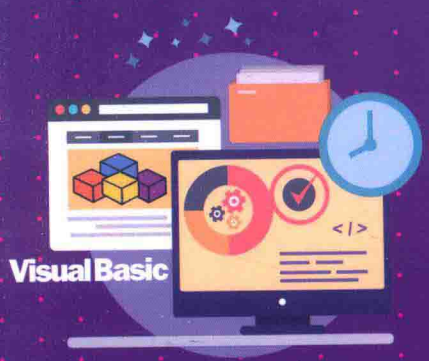
21世纪普通高校计算机
公共课程规划教材

Visual Basic程序设计

◎ 陈明晰 杨谨全 编著

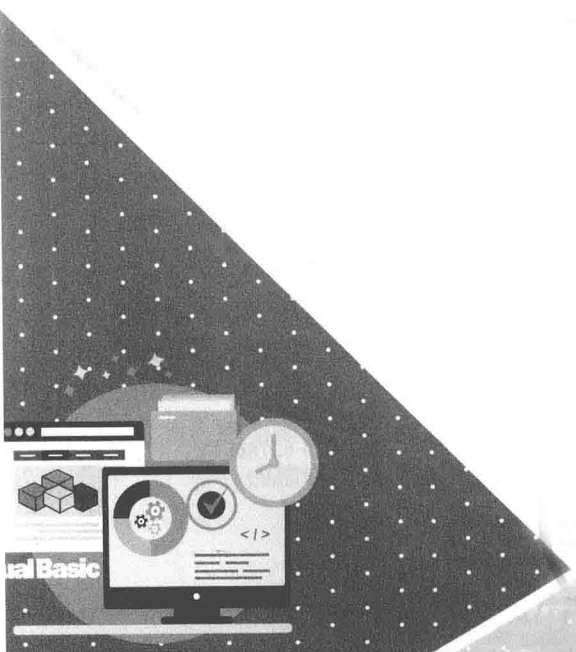


清華大學出版社

21世纪普通高校计算机
公共课程规划教材

Visual Basic程序设计

◎ 陈明晰 杨谨全 编著



清華大學出版社
北京

内 容 简 介

“Visual Basic 程序设计”是为普通高等院校非计算机专业学生开设的程序设计语言课程。本书是将Visual Basic 作为第一门程序设计课程编写的，全书共分为 13 章，内容主要包括程序设计基础及算法与面向对象的概念，应用程序与常用控件，数据类型与表达式，顺序、选择、循环 3 种基本结构程序设计，数组，子过程与函数过程，高级控件，文件，用户界面设计与工程应用，数据库应用基础以及上机练习。

本书内容全面，知识由浅入深、注重实践，各章节均安排了适量的习题，并在最后一章中给出 12 组上机练习题以方便上机实践，适合作为高校“Visual Basic 程序设计”课程教材，也可作为全国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/陈明晰，杨谨全编著. —北京：清华大学出版社，2019(2019.5重印)
(21 世纪普通高校计算机公共课程规划教材)
ISBN 978-7-302-52430-4

Ⅰ. ①V…　Ⅱ. ①陈… ②杨…　Ⅲ. ①BASIC 语言－程序设计－高等学校－教材　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2019)第 039292 号

责任编辑：郑寅堃
封面设计：刘　键
责任校对：梁　毅
责任印制：丛怀宇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954
印 装 者：北京九州迅驰传媒文化有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：20.25　　**字　　数**：495 千字
版　　次：2019 年 4 月第 1 版　　**印　　次**：2019 年 5 月第 3 次印刷
印　　数：1301～1600
定　　价：59.00 元

产品编号：081657-01

前言

程序设计是高等院校重要的基础课程之一。根据教育部高等学校计算机基础课程教学指导委员会提出的《关于进一步加强高校计算机基础教学的意见》精神，“程序设计基础”课程一般定位为各专业大学生第二门计算机公共基础课，该课程的学习可以让学生掌握一种高级程序设计语言，了解程序设计的思想和方法，培养其程序设计的能力。

Visual Basic，简称 VB，是 Microsoft 公司推出的一种 Windows 应用程序开发工具，是当今世界上使用最广泛的编程语言之一，它也被公认为编程效率最高的编程方法之一。无论是开发功能强大、性能可靠的商务软件，还是编写处理实际问题的实用小程序，VB 都是快速、简便的方法之一。

VB 具有所见即所得的友好界面设计，但程序设计除了界面设计，很重要的一部分是程序设计，即算法的实现。所以 VB 程序设计也分为 Visual 可视化界面设计和 Basic 程序设计两个部分。本书在介绍这两个方面的内容时，仍然以程序设计为重点，对程序设计的基本知识、基本语法、编程方法和常用算法进行了较为系统、详细的介绍，使学生学会分析问题、掌握简单问题的编程能力；而可视化界面设计在实际应用中是不可缺少的，也是 VB 的特点，相对比较容易掌握和实现。

本书是由具有多年教学经验的一线教师编写的，内容编排由浅入深、循序渐进、通俗易懂；通过大量的例题介绍，既向学生传授传统的语法规则和对算法的分析思路，也注重培养学生面向对象的概念和控件、属性、方法的使用。本书除各章配有大量的习题外，还配有相应上机练习，并将上机练习集中于书的最后一章，以加强实践教学环节，提高学生的动手实践能力。

本书由陈明晰、杨谨全编著。在编写过程中参阅了大量的参考文献、资料，在此对相关作者表示衷心的感谢。由于编者水平有限，加之时间仓促，书中不足之处在所难免，恳请读者批评指正。

编　者

2018 年 10 月

目 录

第1章 Visual Basic 程序设计概述

Visual Basic，简称 VB，是 Microsoft 公司推出的一种 Windows 应用程序开发工具，是基于对象的程序设计语言。由于其简单易学和开发效率高等特点被广泛用于应用软件的开发，特别是基于数据库应用程序的开发中。本章主要介绍程序设计和算法的基本概念；面向对象程序设计的基本概念；VB 的发展、功能及特点；VB 的集成开发环境；创建应用程序的过程。

1.1 程序设计基础

实际上计算机不是“智能”的，计算机的每一步操作都是根据人们事先设置好的指令实现的。人们为了解决问题，使计算机执行一组操作，就必须事先编写好一条一条的指令，输入到计算机中，这个过程就是程序设计。

计算机程序（简称程序）是指一组指示计算机或其他具有消息处理能力的装置每一步动作的指令，用某种程序设计语言编写，运行于某种目标体系结构上。打个比方，一个程序就像一个用汉语（程序设计语言）写下的某道菜的菜谱（程序），用于指导懂汉语和烹饪手法的人（体系结构）来做这道菜。通常，计算机程序要经过编译和链接而成为一种人们不易看懂而计算机可解读的格式，然后运行。

计算机的所有操作都是由程序控制的，计算机实质上是执行程序的机器，程序和指令是计算机系统中最基本的概念。只有理解程序设计，才能真正了解计算机的工作过程，才能更好地使用计算机来解决实际问题。

1.1.1 程序设计语言

人和人之间的交流需要通过语言。人和计算机之间交流信息，也需要通过语言，这就是人和计算机都能够理解的语言——计算机语言，即计算机程序设计语言，简称程序设计语言。

计算机语言经历了以下几个发展阶段。

1. 机器语言

我们目前使用的计算机是电子计算机，其对信息的处理和存储工作是基于二进制的，也就是说计算机只能识别 0 和 1 组成的指令。机器指令（Machine Instructions）是 CPU 能直接识别并执行的指令，它的表现形式是二进制编码。机器指令的集合称为机器语言（Machine Language）。

机器语言和我们习惯使用的自然语言差别很大，只有少数计算机专业人员才会编写机

器语言程序，很难推广使用。

2. 汇编语言

为了克服机器语言难学、难记的缺点，人们设计出一种用英文字母和数字表示指令（助记符）的计算机语言，这就是汇编语言（Assembler Language）。

计算机并不能直接识别和执行汇编语言程序，需要用汇编程序，把汇编语言的指令翻译成机器语言指令。

虽然汇编语言比机器语言简单，而且好记忆一些，但还是很难普及，只在专业人员中使用。不同型号的计算机的机器语言和汇编语言是互不通用的，机器语言和汇编语言是完全依赖于具体机器的，是面向机器的语言，所以称为低级语言。

3. 高级语言

由于汇编语言依赖于硬件体系，且助记符量大难记，于是人们又设计开发了更加易用的所谓高级语言。在这种语言中，其语法和结构更类似普通英文加数学符号，且由于远离对硬件的直接操作，使得一般人经过学习之后都可以编写程序。

高级语言与计算机的硬件结构及指令系统无关，它有更强的表达能力，可方便地表示数据的运算和程序的控制结构，能更好地描述各种算法，而且容易学习掌握。但高级语言编译生成的程序代码一般比用汇编语言设计的程序代码要长，执行的速度也慢。所以汇编语言适合编写一些对速度和代码长度要求高的程序和直接控制硬件的程序。高级语言、汇编语言和机器语言都是用于编写计算机程序的语言。

高级语言程序“看不见”机器的硬件结构，不能用于编写直接访问机器硬件资源的系统软件或设备控制软件。为此，一些高级语言提供了与汇编语言之间的调用接口。用汇编语言编写的程序，可作为高级语言的一个外部过程或函数而被调用执行。

程序设计语言从机器语言到高级语言的抽象，带来的主要好处是：

（1）高级语言接近算法语言，易学、易掌握，一般工程技术人员经过短时间的培训就可以设计并编程实现解决工程问题的应用程序。

（2）高级语言为程序员提供了结构化程序设计的环境和工具，使得设计出来的程序可读性好，可维护性强，可靠性高。

（3）高级语言远离机器语言，与具体的计算机硬件关系不大，因而所写出来的程序可移植性好，重用率高。

（4）高级语言由于把繁杂琐碎的事务交给了编译程序去做，所以自动化程度高，开发周期短，且程序员得到解脱，可以集中时间和精力去从事对于他们来说更为重要的创造性劳动，以提高程序的质量。

随着计算机的发展，高级语言也不断发展，种类繁多，但基本可归为以下几类。

（1）命令式语言。命令式语言的语义基础是模拟“数据存储/数据操作”的图灵机可计算模型，十分符合现代计算机体系结构的自然实现方式。其中产生操作的主要途径是依赖语句或命令。现代流行的大多数语言都是这一类型，例如 Fortran、Pascal、Cobol、C、C++、Basic、Ada、Java、C# 等，各种脚本语言也被看作此种类型。

（2）函数式语言。函数式语言的语义基础是基于数学函数概念的值映射的 λ 算子可计算模型。这种语言非常适合进行人工智能等工作的计算。典型的函数式语言如 Lisp、Haskell、ML、Scheme 、F# 等。

(3) 逻辑式语言。逻辑式语言的语义基础是基于一组已知规则的形式逻辑系统。这种语言主要用在专家系统的实现中。最著名的逻辑式语言是 Prolog。

(4) 面向对象语言。面向对象语言(Object-Oriented Language)是一类以对象作为基本程序结构单位的程序设计语言,用于描述的设计是以对象为核心,而对象是程序运行时刻的基本成分。该语言中提供了类、继承等成分。

1.1.2 算法和流程图

程序设计,首先要掌握解题的方法和步骤,也就是如何将待求解的问题分解成一系列的操作步骤。这就是"算法"(Algorithm)需要研究的问题。

1. 算法的定义

通俗地讲,"算法"是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精确描述,是一系列解决问题的清晰指令。算法代表着用系统的方法描述解决问题的策略机制,也就是说,能够对一定规范的输入,在有限时间内获得所要求的输出。如果一个算法有缺陷,或不适合于某个问题,执行这个算法将不会解决这个问题。不同的算法可能用不同的时间、空间或效率来完成同样的任务。处理任何问题都有"算法"的问题。设计算法要确定两个问题,一是必须做什么,二是按什么顺序去做。例如发电子邮件,首先登录邮件系统,然后选择"新建邮件",填写收件人邮箱地址、邮件主题,输入邮件内容,最后发送邮件。在这个过程中有些必须做,如必须填写邮件地址,有些事情可以不做,如填写邮件主题。对同一个问题,可以有不同的解题方法和步骤。

【例 1.1】 计算 1+2+3+…+100,即 $\text{sum}=\sum_{1}^{100} n$。

算法一:

先计算 1+2,再加 3,再加 4,一直加到 100。

sum=1+2+…+100=5050

算法二:

先计算 99+1,98+2,一直到 51+49,再加上 100 和 50。

sum=100+(99+1)+(98+2)+…+(51+49)+50=100+49×100+50=5050

为了有效地解决问题,不仅需要保证算法正确,还要考虑算法的质量,即方法简单,运算步骤少。

2. 算法的特征

一个算法应该具有以下 5 个重要的特征。

(1) 有穷性:算法必须在执行有限个步骤之后终止,不应当出现无终止的循环或永远执行不完的步骤。

(2) 确定性:算法中的每一步骤必须有确定的含义,不能有二义性,即不应含混不清或模棱两可。

(3) 输入性:一个算法可以有输入的数据,也可以没有输入的数据,即有 0 个或多个输入,以描述运算对象的初始情况。

(4) 输出性:一个算法至少有一个输出的数据。因为算法的目的就是求解问题,求解

的结果必须向用户输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

(5) 有效性：算法中执行的任何计算步骤都是可以被分解为基本的、可执行的操作步骤，即每个计算步都可以在有限时间内完成。

3. 算法的表示

计算机算法分为数值运算算法和非数值运算算法两大类。数值运算算法的目的是求数值解，例如求方程的根等。非数值运算算法包括的范围非常广，如企业管理、车辆调度等。目前应用最广的应用领域是非数值运算。描述算法的目的是便于理解，并利用算法解决问题。

描述算法的方法有多种，常用的有自然语言、伪代码、结构化流程图和 PAD 图等，其中最普遍使用的是流程图。

(1) 自然语言：即日常生活中所使用的语言。人们都懂，易于理解，但不严格，对一些问题的描述可能存在歧义或二义性。

(2) 伪代码：类似于高级语言的一种描述算法的方法。比较接近自然语言，表达方式简单，便于理解。伪代码通常接近于某种程序设计语言，比较容易将算法的描述转化为程序。

(3) 流程图：用一些图框、流程线以及文字说明来描述操作过程。

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，如图 1-1 所示。

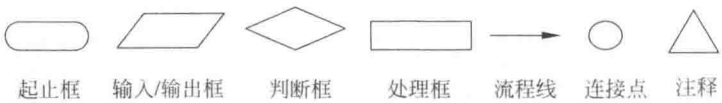


图 1-1　常用流程图符号

- 起止框：表示算法的开始和结束。
- 输入/输出框：表示输入、输出操作。
- 判断框：表示判断操作，框中标明判断条件。判断框具有两个或两个以上的出口，根据判断结果确定后续执行的路径。
- 处理框：表示具体的处理操作。
- 流程线：连接流程图中的各个组成部分。
- 连接点：用于将画在不同页面上的流程图(或多个流程图)连起来。连接点中标有数字或字母。连接点用于连接比较复杂或分别画的多个图。
- 注释：对流程图进行注释说明。

【例 1.2】 用流程图描述算法，分别输入 a、b，若 $a>b$，输出 a，否则输出 b。流程图如图 1-2 所示。

(4) PAD 图：PAD 是问题分析图(Problem Analysis Diagram)的英文缩写，是 1974 年由日本的二村良彦等人提出的又一种主要用于描述软件详细设计的图形表示工具。与方框图一样，PAD 图也只能描述结构化程序允许使用的几种基本结构。它用二维树形结构的图表示程序的控制流。以 PAD 图为基础，遵循机械的走树(Tree Walk)规则就能方便地编写出程序，且用这种图转换为程序代码比较容易。

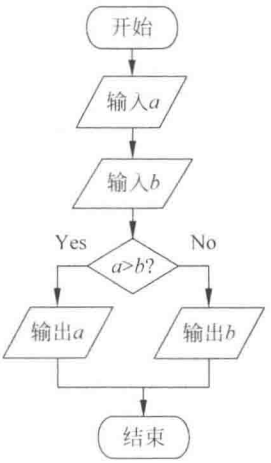


图 1-2 【例 1.2】的流程图

1.1.3 结构化程序设计

结构化程序设计(Structured Programming)是以模块功能和处理过程设计为主的软件设计方法。其概念最早由 E. W. Dijikstra 在 1965 年提出,是软件发展的一个重要的里程碑。它的主要观点是采用自顶向下、逐步求精及模块化的程序设计方法,使用三种基本控制结构构造程序,即任何程序都可由顺序、选择、循环三种基本控制结构构造。结构化程序设计主要强调的是程序的易读性。

1. 结构化程序设计的内容

结构化程序设计曾被称为软件发展中的第三个里程碑。该方法的要点是:

(1) 使用顺序、选择、循环三种基本结构来嵌套连结成具有复杂层次的"结构化程序",严格控制 GOTO 语句的使用。用这样的方法编出的程序在结构上具有以下效果。

- 以控制结构为单位,只有一个入口和一个出口,所以能独立地理解这一部分。
- 能够以控制结构为单位,从上到下顺序地阅读程序文本。
- 由于程序的静态描述与执行时的控制流程容易对应,所以能够方便、正确地理解程序的动作。

(2) 采用"自顶而下,逐步求精"的设计思想。其出发点是从问题的总体目标开始,抽象低层的细节。先专心构造高层的结构,然后再一层一层地分解和细化。这使设计者能把握主题,高屋建瓴,避免一开始就陷入复杂的细节中,使复杂的设计过程变得简单明了,过程的结果也容易做到正确、可靠。

(3) 采用"独立功能,单入口、单出口"的模块结构,减少模块的相互联系使模块可作为插件或积木使用,降低程序的复杂性,提高可靠性。程序编写时,所有模块的功能通过相应的子程序(函数或过程)的代码来实现。程序的主体是子程序层次库,它与功能模块的抽象层次相对应,编码原则使得程序流程简洁、清晰,增强了可读性。

2. 三种基本结构流程图

不论多么复杂的程序都可以使用顺序、选择、循环三种基本结构描述。