高等院校计算机系列规划教材

Visual Basic程序设计

孙志辉　编著

机械工业出版社
CHINA MACHINE PRESS

高等院校计算机系列规划教材

Visual Basic 程序设计

孙志辉　编著

机械工业出版社

本书是学习 Visual Basic 的基础教程，采用语法与实例相结合的方法，并以实例作为教程内容的主线，将 Visual Basic 的基本语法和对象融入精心挑选的 6 个综合实例中。全书共分 7 章，第 1 章介绍程序设计基本概念、Windows 程序设计特点以及 Visual Basic 6.0 的集成开发环境，其余各章则分别介绍简单屏幕保护程序、记事本程序、数独游戏程序、简单计算器程序、电子闹钟程序和学生信息管理程序的设计方法。每章实例还通过 API 函数调用实现了功能的进一步提高。对于每个实例均采用循序渐进的方法，逐步实现其各项功能。

通过这种以实例讲语法的内容安排，不仅教会学生 Visual Basic 的基本语法，还教会学生程序设计的思想，真正做到学以致用，成为合格的 Visual Basic 程序员。

本书可作为高等院校计算机语言课程的教材，也可作为 Visual Basic 语言爱好者自学或培训机构的教材，还可作为 Visual Basic 二级等级考试的学习参考书。

图书在版编目（CIP）数据

Visual Basic 程序设计 / 孙志辉编著. —北京：机械工业出版社，2009.6
（高等院校计算机系列规划教材）
ISBN 978-7-111-27511-4

Ⅰ. V… Ⅱ. 孙… Ⅲ. BASIC 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2009）第 114396 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
责任编辑：赵　轩　　李　宁
责任印制：洪汉军

北京市朝阳展望印刷厂印刷

2009 年 8 月第 1 版 · 第 1 次印刷
184mm×260mm · 22.25 印张 · 552 千字
0001—3000 册
标准书号：ISBN 978-7-111-27511-4
定价：37.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
销售服务热线电话：（010）68326294　68993821
购书热线电话：（010）88379639　88379641　88379643
编辑热线电话：（010）88379753　88379739
封面无防伪标均为盗版

前　言

为了适应信息化时代的需求，各高校对大多数专业都开设了一定的计算机语言课程，有的讲授 C 或 C++语言，有的讲授 Visual Basic 语言。Visual Basic 以其语法简单、可视化设计、功能强大，成为许多学校计算机语言课程的首选语言。

目前，国内各出版社出版了大量的 Visual Basic 语言教程，它们都具有同样的特点：介绍语言的语法等基本知识，再辅助一些简单的示例。学生通过学习，可能能够考一个不错的成绩，但所掌握的仍然是一些支离破碎的知识，不能将所有的知识融会贯通，遇到具体的程序设计，还是无从下手。

本书正是为了解决这个问题，对 Visual Basic 语言教程，不再以讲解语法为主线，而是以精心挑选的 6 个实例程序为主线，将 Visual Basic 语言的基本语法融入到这些例子中。每个例子均是一个完整的应用程序，并且具有一定的实用价值。通过这些实例，不但教会学生 Visual Basic 语言的基本语法，还教会他们编程的思想，培养学生解决实际编程问题的能力，真正做到学以致用。

第 1 章介绍程序设计的基本概念以及 Visual Basic 6.0 的集成开发环境，重点介绍 Windows 程序设计的特点。

第 2 章介绍具有简单文字移动界面的屏幕保护程序设计方法。通过该程序，介绍 Visual Basic 程序设计的基本方法，以及标签、文本框和定时器控件的使用。

第 3 章介绍具有自动翻页功能记事本程序的设计方法。通过该程序，进一步介绍主要控件的使用方法，并讲解 Visual Basic 文件操作的基本方法以及菜单、工具栏、状态栏等设计方法。

第 4 章介绍数独游戏程序的设计方法。通过该程序，介绍了 Visual Basic 6.0 普通数组、动态数组和控件数组的有关语法，并讲解与游戏程序有关的计时、排行榜等功能实现。

第 5 章介绍简单计算器程序的设计方法。通过该程序，讲解控件数组及其应用，以及简单图形操作。

第 6 章介绍电子闹钟程序的设计方法。通过该程序，讲解 Visual Basic 6.0 的日期时间操作方法以及基本图形操作。

第 7 章介绍学生信息管理程序的设计方法，包含了 Visual Basic 6.0 有关数据库操作的所有语法和对象。

在每章内容安排上，首先介绍需要的基本语法和对象，然后采用循序渐进的方法，逐步实现实例程序的基本功能，并注重细节，讲解代码设计中的难点和问题。在完成基本功能的基础上，通过 API 函数调用等实现功能的进一步提高。基本功能部分满足 Visual Basic 语言的基本要求，涵盖了国家计算机二级考试的大纲范围。具体安排教学内容时，可以只讲授基本功能部分，提高功能部分则作为学生自学或有一定基础的学生参考。

全书由北京科技大学孙志辉编写。

由于编者水平有限，所编写的实例程序不一定为最佳，书中难免会存在一些错误，殷切希望广大读者批评指正。

编　者

目　录

第1章　计算机程序设计基础

1.1　编程语言及其发展

1.1.1　计算机编程语言

计算机程序是计算机所执行的一系列指令的集合。通过这些指令集合，计算机可以实现数值计算、信息处理、信息显示等功能。

计算机系统采用电信号表示其内部的所有信息，而每个电信号往往采用通、断两个状态表示。因而，计算机内的所有信息均采用二进制格式保存，无论是执行指令、需要处理的数据，还是显示的文字符号。例如，文字处理时，在屏幕某个位置显示字母“A”，实际是将65这个数据的二进制格式送给了显卡，由显卡根据65对应的字母A的点阵特征，输出视频信号给显示器，从而在显示器的某个位置“画”出字母“A”。

从第一台电子计算机开始，就采用二进制格式存储计算机指令。这种格式的指令称为机器语言，是CPU唯一能够识别的内容。例如，将数18送给普通微型计算机CPU内部的某个寄存器BX（寄存器是CPU内部用于存放数据的单元），其机器语言格式如下：

```
10111011　00010010　00000000
```

这样以若干0和1组成的指令不便于记忆，一般人往往无法记住CPU某个指令的二进制格式。为此，引入了助记符的概念，即采用便于记忆的英文单词或其缩写格式代表相应的机器语言。例如，对上面的机器语言，采用下面格式表示：

```
MOV BX,18
```

用“MOV”这样的缩写表示传送数据，一般程序员只要了解代表指令的助记符，就可以编写程序。

但是，这样书写的计算机程序，计算机的CPU是无法识别的。为此，需要把助记符格式的程序翻译成对应的机器语言，这个过程称为编译（Compile），是由专门的工具实现的，如微软公司提供的MASM工具就可以进行编译工作。

采用助记符格式的编程语言，称为汇编语言，所有的CPU系统都具有自己的汇编语言。汇编语言虽然解决了程序设计的基本问题（不需要记忆那些0和1的组合），但仍然存在如下问题，即汇编语言需要程序员了解CPU的结构和基本工作原理。如果需要计算18+20的结果，必须先将参与计算的一个数送到计算机内部的某个寄存器中（如BX寄存器），然后才能执行加法指令，加的结果还需要再送回内存的某个区域，以便CPU进行下一步的计算。程序员必须知道CPU内部有哪些寄存器，其中又有哪些寄存器能够用于存放参与计算的数据。

计算18+20的结果用汇编语言书写如下：

```
MOV BX,18
ADD BX,20
MOV [1000],BX
```

对编程人员来说，只需要确定让 CPU 计算加法，而不想了解其细节。基于这样的目的，发明了各种高级编程语言，如 FORTRAN、BASIC、C 等，它们均采用符合人类自然描述语言的语法书写计算机程序。例如，BASIC 语言实现上述计算的格式如下：

```
A=10+20
```

高级语言简化了程序设计的难度，程序员不必了解细节，编写的程序由专门的编译工具转换成机器语言。正是这些高级语言的产生，才使得计算机编程能够推广开来。

常见的高级编程语言如下：

1）DOS 应用程序：FORTRAN、BASIC、Pascal、C。

2）Windows 应用程序：Visual C++、Visual Basic、Delphi、C++ Builder、Java。

编程语言除了进行简单的加、减、乘、除计算外，有时还需要进行更复杂的科学计算，如三角、指数、对数等函数计算，而 CPU 并没有与这些函数对应的指令，只能采取一些近似的数值算法。这些数值算法并非每个编程人员都清楚，因而每个编程语言都提供了这些函数的通用算法，并以库函数的形式提供给程序员。编程时，程序员只需要简单地调用这些标准函数即可，如 Visual Basic 采用 sin(x)进行正弦函数计算。

当编译工具把程序员编写的高级语言程序（称为源程序）编译成机器语言时，遇到其中的函数，并不能转换成机器语言。这样编译的程序称为目标程序，以.obj 为扩展名。不管是什么编程语言，编译后的目标程序都是统一的机器格式。

为了产生真正可以运行的程序，还需要将编译好的目标程序与编程语言提供的库文件中某些函数的指令连接在一起。这个步骤称为链接（Link），只有经过链接的程序才能产生可执行的.exe 文件。

所有语言的编程步骤如下：

1）编辑（编写源程序）。

2）编译（转换成目标程序）。

3）链接（生成可执行程序）。

需要说明的是，不同语言编译的方式不同。有的语言是先将所有程序代码一起编译成机器语言，再链接生成可执行文件，如 C 语言和 Pascal 语言，这种语言称为编译型语言，最后以可执行的.exe 文件运行；有的语言则可以边编译边执行，如 BASIC 语言和 Java 语言，这种语言称为解释型语言；也有些语言既提供编译运行的方式，也提供解释运行的方式，如 Visual Basic，在调试程序时可以采用解释型，一旦调试完成，则采用编译型，将源程序编译成可执行的.exe 文件。编译型语言的程序执行速度比解释型语言的程序执行速度快。

编程语言种类很多，初学者究竟应该选择哪种语言？在常用的 Windows 应用程序编程语言中，Visual Basic 难度最小，功能也最弱；而 Visual C++难度最大，功能也最强大。如果只希望编写简单的初级程序，尽量选择 Visual Basic；如果需要编写功能强大的程序，特别是编写与计算机硬件有关的程序，则选择 Visual C++。其实，语言本身的语法都比较简单，掌握起来也不难。

Visual Basic 将 Windows 应用程序设计的许多细节都隐藏起来了，程序员不需要了解其细

节，只要依照其语法编程即可。而采用 Visual C++语言编程，必须了解 Windows 程序设计的细节，以及封装这些细节的几百个类（即微软基本类库 MFC），而这正是 Visual C++学习的难点。

虽然 Visual Basic 隐藏了许多细节，但了解和掌握 Windows 程序设计的细节对学好 Visual Basic 有很大的帮助，还可以应用这些细节编写出功能强大的应用程序。

本章将介绍 Windows 程序设计的特点，更多的细节可以参考有关 Windows 程序设计的书籍。

1.1.2 编程语言涉及的主要内容

各种编程语言的语法不同，但都涉及变量定义、赋值语句、条件语句、循环语句、结构与类等。

1．变量与类型

在程序设计过程中，往往涉及到大量的数据，这些数据一般都存放在存储器中（主要是内存中）。所谓变量定义，就是在内存中申请一个区域保存某个数据。内存以 8 位二进制为单元保存信息，这样一个单元称为字节（Byte）。实际申请内存时，可能需要一个或多个字节单元，申请的内存单元数量越多，可以保存的数据越大。变量的类型就是申请内存时的单元数量和数据存放在这些单元中的格式。

编程语言往往都提供一些变量类型以存放整数、小数和字符。其中，整数的存放可以申请的内存单元一般为 1 个、2 个、4 个和 8 个；小数的存放可以申请的内存单元一般为 4 个和 8 个；而字符的存放可以申请的内存单元一般为 1 个和 2 个。

以 Visual Basic 6.0 为例，整数类型分成字节型（Byte）、整型（Integer，占 2 个单元）、长整型（Long，占 4 个单元）；小数类型分成单精度（Single，4 个单元）、双精度（Double，8 个单元）；字符类型为 String。

每一种类型表示的数据范围一定，在定义时一定要先了解数据的范围，再选择合适的类型。例如，Visual Basic 6.0 的字节型（Byte）可以保存 0～255 的数据，如果把 256 赋给它，则保存不下，运行程序时就会提示出错。

不同的编程语言，各种类型需要的内存单元数量和表示的数据范围不同。例如，整数（Integer），Visual Basic 6.0 采用 2 个内存单元，表示数据范围为–32768(2 的 15 次方)～32767；而 Visual Basic .NET 则采用 4 个内存单元，表示数据范围为–2147483648（2 的 31 次方）～2147483647。

计算机保存正整数比较容易，采用其二进制格式即可，负整数则需要采用特殊的补码格式保存，对于 n 位的负整数，其表示范围为 $-2^{n-1} \sim 2^{n-1}-1$。

小数的存放则比较麻烦，一般采用浮点格式，即把小数表示为指数和尾数的格式分别保存，单精度和双精度小数的表示范围大约为 10^{38} 和 10^{308}，它们的有效位数分别为 7 位和 15 位。例如，Visual Basic 6.0 将某个变量定义成单精度（Single），则数据 100000.01 保存时，最后的 1 不能存放，实际保存为 1000000.0。在处理小数时一定注意这样的问题。

小数保存时，先转换成对应的二进制格式，再按照指数和尾数格式存放。并非所有的十进制小数都能准确地转换成相应的二进制格式，如十进制小数 0.1 表示成二进制 $0.\dot{0}0\dot{0}1\dot{1}$（乘 2 取整法，即每次乘以 2，将整数部分提出，保留小数部分继续乘以 2，直到结果为 1），为无

限循环，如果保存为单精度，则保存从1～23位二进制（单精度格式共4个单元32位，其中1位表示正负，8位表示指数，23位表示尾数）。绝大多数小数保存的都是近似值。

对于同一类型的一批数据，可以采用数组形式进行定义，这样就可以使用循环结构进行处理。

程序执行时，可以申请的内存区域有三处：

1）程序中定义的数据段。这是每个应用程序保存其特定数据的区域，在应用程序进入内存开始运行时，就已经确定了其位置和大小。其中保存什么样的数据，每个数据存放在哪儿，都是固定的。这段内存在应用程序执行的整个过程中都是有效的，任何别的应用程序都不能占用，只有应用程序结束后才可以释放内存再分配给其他应用程序。只要在这个区域定义的变量，在整个应用程序执行过程中都有效，这样的变量称为全局变量。实际上，应用程序在开始执行时，就是按照全局变量的定义情况来分配数据段的。另外，子程序或函数中定义的静态变量也是在数据段内申请内存，因而在子程序或函数结束后数据仍然有效。静态变量与全局变量的区别在于静态变量只能由某个子程序或函数访问，而全局变量可以由所有的子程序和函数访问。

2）程序中定义的堆栈段（Stack）。这是每个应用程序保存其临时数据的区域，该区域的位置和大小也是确定的，但程序开始运行时并不确定其中保存什么数据。在应用程序运行过程中，根据需要把特定数据保存在其中。保存在这里的数据按照先进后出的方式管理，在不同的时刻，其中的数据代表的含义不同。定义在这里的变量称为局部变量，只在某个特定范围内有效。一般，在子程序或函数内部定义的普通变量都是定义在堆栈中的，进入子程序时申请堆栈中的内存存放这些变量，数据有效，一旦子程序运行结束，系统自动释放这些内存，则数据无效。

3）当前系统剩余的内存。应用程序执行时根据需要从系统剩余内存申请，这种内存往往称为堆（Heap），只要系统还有空闲内存，就可以申请。数据段和堆栈段的大小是固定的，而堆内存则不固定。需要注意的是，从堆中申请内存后，系统以后能够再次申请的内存数量就减少，随着堆内存申请的增多，系统内存越来越少，最终导致使用硬盘代替内存，降低执行速度，甚至有可能导致系统崩溃。有时候，计算机系统感染病毒后，病毒程序不断申请内存，导致堆中可以申请的内存越来越少，可以明显感觉到程序执行速度变慢，甚至很长时间没有反应，就像死机一样。编写程序时，对于从堆中申请的内存，使用后应该立即释放。

2．变量赋值

赋值是将具体的数据保存到定义该变量时申请的那段内存区域内。如果赋值数据超过所申请的区域，不同语言处理方法不同。Visual Basic提示出错，而Visual C++则以能够存放进去的部分作为数据，超过部分放弃。例如，对于占用1个单元的正整数，Visual Basic中赋值256则出错，Visual C++中赋值256时只把其中的8位部分保存到该单元中，高8位部分放弃，实际存放数据为1。

3．程序控制结构

大多数应用程序并不是按照指令存放的顺序执行程序，往往需要根据条件改变指令的执行顺序，常用的程序控制结构包括转移、条件和循环。

转移结构一般采用Goto语句进行，只要条件满足，就转移到指定的位置执行程序，程序中需要定义转移的目标位置。转移语句会导致程序结构混乱，多次转移后，往往不能够清晰看出程序执行的流程，应该尽量少用Goto语句。实际上，绝大多数使用Goto语句的代码都

可以用条件或循环语句替代。

条件结构一般采用 If 语句进行，有的语言提供了更好的分支结构，如 C 和 C++的 Switch 以及 Visual Basic 的 Select 语句。条件的判断以 0 和非 0 为准则，非 0 则条件满足，0 则条件不满足；有些语言(Visual Basic、Visual C++)以 True 表示条件满足，False 表示条件不满足。不同的条件可以采用逻辑“与”（两个条件都满足）、“或”（只要有一个条件满足）、“异或”（两个条件只能有一个满足）进行组合。

循环结构一般采用 For、Do、While 等语句进行。实际上，循环是一种特殊的条件。满足条件时循环执行部分代码，不满足条件就结束循环。常用的 For 循环包含 4 个参量：循环变量、初值、终值和步长。循环开始时，把初值赋值给循环变量，然后比较循环变量与终值。如果循环变量大于终值（循环步长为正时）或循环变量小于终值（循环步长为负时），则不循环；否则，进入循环执行循环体内的代码，每次循环体内代码执行完成后，循环变量自动加或减步长，然后再比较判断是否结束循环。对于如下的 Visual Basic 循环语句：

```
For i=1 to 10 Step 1
  …
Next i
```

循环变量 i，初值 1，终值 10，步长 1。当 i 变成 11 时，循环正常结束。在循环体内可以采用语句提前结束（Visual Basic 用 Exit For；C 和 C++用 break）。

如果希望利用 For 循环查找数据，则可以根据循环结束后循环变量的值进行判断。例如，程序如下：

```
For i=1 to 10 Step 1
  If a(i)=100 Then Exit for
Next i
If i<=10 Then
    Print "找到"
Else
    Print "没有找到"
End
```

每次循环后，For 循环自动变化循环变量，一般在循环体内不要再改变循环变量，否则可能导致循环次数不正确，严重时可能导致进入死循环（无限循环）。例如，程序如下：

```
For i=1 to 10 Step 1
  i=i-1
Next i
```

循环变量初值为 1，For 循环本身每次自动加 1，而循环体内则每次减 1，循环变量永远也不可能大于 10，程序停顿在这个循环内。

无论是什么类型的循环格式，都需要慎重地考虑其循环结束的条件，特别是一些循环条件隐含的情况。例如，读取文件或数据库数据时，一般根据是否到达结尾来判断循环是否结束，如果循环体内没有读取下一个数据的语句，则永远也不会到达结尾，形成无限循环。

4．子程序与函数

一般程序中往往存在某些代码需要重复执行，这些代码可以编写在一起，以函数或子程

序形式供程序的其他代码调用（Call）。子程序与函数没有本质区别，甚至有些语言合二为一（如C和C++语言统一采用函数）。一般语言的函数具有结果，即通过函数的调用，可以得到一个结果，称为返回值，而子程序没有返回值。

无论是函数还是子程序，其程序结构都是一致的。编写子程序时需要指定参数个数以及每个参数的类型（称为形参），而在调用子程序时则要按照定义时的参数个数和类型提供具体数值（称为实参）。调用时提供的实参通过堆栈传递给子程序使用。

如果参数多于1个，不同语言把实参传送到子程序或函数的顺序不同，C或C++按照从右到左的顺序，而Visual Basic则按照从左到右的顺序。

将实参数据传递给子程序或函数时，有两种方式：一种是把具体的数传递过去，称为按值传送；另一种是把数据在内存中的存放位置（地址）传递过去，称为按地址传送。按值传送的参数在子程序或函数内部不能再改变，而按地址传送的参数在子程序或函数内部可以改变。C和C++函数的参数默认按值传送，其指针方式为按地址传送；Visual Basic的参数默认按地址传送，只有用ByVal修饰的参数才是按值传送。

5．结构与类

每种语言都提供了一定的变量类型，它们能够存放数据的范围是固定的。特定情况下需要存放的数据可能超过了已有的类型，则需要自己定义变量类型，即以已有变量类型组合起来定义新的类型，这种方式称为结构（Structure）或自定义类型。结构中的每个基本类型称为其成员。只要定义过某个结构，就可以像基本类型一样用于定义变量。

如果在结构的成员中引入函数，则形成了类（Class），这是面向对象编程（OOP）思想的基础。由于类的成员既有变量（可以用于描述某个对象的特征），又有函数（可以用于执行对象的某种功能或动作），能够形象地表示某种类型的事物（Object）。类是某种类型对象的公共描述，具体到某个具体对象则需要一个具体化过程（也称为实例化过程），就像用普通类型定义变量一样，也需要用类定义对象，只有经过定义过的具体对象才能够对其进行操作。例如，“车”是一个类，不能直接描述其颜色、排量，只有具体化到某一辆实物车，才可以评价其颜色、排量等。

Windows操作系统是由大量的对象组成，每个窗口就是一个对象，窗口中的各种组成元件也都是对象。无论Visual Basic还是Visual C++，编写Windows应用程序都离不开对象。只不过，Visual Basic淡化了类的概念（但也提供了类的操作方法），而Visual C++则完全是采用各种类组成应用程序。

1.1.3 Windows程序设计特点

无论用什么语言编写的程序，总是由操作系统加载后才能执行。在DOS系统下，通过输入执行文件名称再按〈Enter〉键的方式加载程序，而Windows系统则通过双击执行文件或其快捷方式加载程序（也可以输入执行文件名称加载）。

Windows操作系统与DOS操作系统在加载执行文件后的处理方式上存在很大的区别。

DOS加载应用程序后，将控制权交给应用程序，只有应用程序执行完成后再将控制权交还DOS操作系统。在某个应用程序运行过程中，无论是键盘的输入，还是外部数据的到达，都需要通过该程序本身的代码去读取处理，如果在应用程序编写的代码中没有键盘输入处理，则在应用程序运行过程中的键盘输入都没有办法处理，这就要求编写程序时考虑有关的硬件