贵州大学规划教材
贵州大学教材建设委员会审核批准

Visual Basic 程序设计基础

主 编：郭 剑 张 萍

贵州大学出版社
Guizhou University Press

贵州大学规划教材
贵州大学教材建设委员会审核批准

Visual Basic 程序设计基础

主　编：郭　剑　张　萍
编　委：陈郁明　李露蓉　龙桂珍　章　颖

贵州大学出版社

图书在版编目（CIP）数据

Visual Basic 程序设计基础 / 张萍，郭剑主编．
-- 贵阳：贵州大学出版社，2011.8
ISBN 978-7-81126-372-5
Ⅰ．①V… Ⅱ．①张… ②郭… Ⅲ．①BASIC
语言－程序设计－教材 Ⅳ．①TP312
中国版本图书馆 CIP 数据核字 (2011) 第 167271 号

Visual Basic 程序设计基础

编　　者：郭　剑　张　萍
责任编辑：徐　言
出版发行：贵州大学出版社
印　　刷：贵阳佳迅印务有限公司
开　　本：890 毫米 ×1240 mm 毫米　1/16
印　　张：21.5
字　　数：563 千
版　　次：2011 年 8 月 第 1 版
印　　次：2011 年 8 第 1 次印刷
书　　号：ISBN 978-7-81126-372-5
定　　价：42 .00 元

前 言

《高级程序设计语言Visual Basic》课程从2000年左右自工科专业开设以来，经历过几次教材选定的变动，变动主要有四个版本：① 同济大学龚培曾主编，高等教育出版社出版的《Visual Basic程序设计简明教程》；②贵州工业大学王力主编，贵州科技出版社出版的《Visual Basic程序设计教程》；③贵州大学李祥主编，贵州人民出版社出版的《Visual Basic语言教程》；④教育部考试中心编订，高等教育出版社出版的《全国计算机等级考试二级教程——Visual Basic语言程序设计》。各个版本的选定都遵循了与本校实际相结合的原则。经历一段时间的选用后，某些内容的选材上，逐渐与课程的发展不相适应。教师讲义内容的变化，与学生手中教材内容形成差异，降低了教材的核心职能。本着教材内容与讲授内容相适应的原则，学院组织有经验的任课教师进行了教材的改革与重新编订。

《高级程序设计语言Visual Basic》课程继承自80年代以来高校普遍开设的《高级程序设计语言BASIC》，BASIC作为程序入门级语言，是一种典型的结构化程序设计语言，而它的继承者Visual Basic刚开设之初，由于教师对面向对象设计方法的陌生，只能简单地把课程开成了窗口设计与Basic语法的简单相加，这使得很长时间内该课程更多地注重了窗口界面设计，偏离了程序设计课程的主旨，这个时期的教材也明显具有这方面的缺点。随着《全国计算机等级考试》的发展，市场上充斥了大量的等级考试方面的辅助教材，能力教育逐渐演变成了应试教育，对本科非计算机专业学生的程序设计能力培养形成了误导。回归程序设计课程的核心，进行程序设计能力培养，教材改革成为必然，新编教材在内容上进行了相应调整。

全书共分为13章，并配套有《Visual Basic程序设计基础实验指导与习题测试》的实验指导教材。由于本书所涉及的内容较多，加之时间仓促，难免有不足之处，敬请读者多提宝贵意见。

本书的编写内容系参照《全国计算机等级考试二级Visual Basic考试大纲》以及最新发布的《中国高等院校计算机基础教育课程体系2008》（蓝皮书）制定而成，保证了内容的时效性，同时可以作为参加全国计算机等级考试二级Visual Basic考试的读者的辅助教材。

郭剑 张萍

2010年1月5日

内容提要

本书以Visual Basic程序设计语言为媒介，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。同时，可视化界面设计作为操作方便的窗口界面设计方法，与程序设计语言两者有机结合，既可以提高学生学习积极性又可提高编程效率和教学效果，真正达到学以致用的目的。

本书编写过程中大量吸取了多年从事Visual Basic程序设计教学的一线教师的教学经验以及讲义，更有助于读者学会分析问题，掌握简单问题的编程。作为入门级的程序设计语言，方便读者自学使用。

本书的编写内容系参照《全国计算机等级考试二级Visual Basic考试大纲》以及最新发布的《中国高等院校计算机基础教育课程体系2008》（蓝皮书）制定而成，保证了内容的时效性，同时可以作为参加全国计算机等级考试二级Visual Basic考试的读者的辅助教材。

目 录

第 1 章 Visual Basic 程序设计入门 1
1.1 程序设计概述 1
1.2 程序设计方法 2
1.3 程序设计语言 4
1.4 第一个 Visual Basic 程序 6
1.5 Visual Basic 的运作原理 16
思考题 20
第 2 章 算法初步 21
2.1 算法的概念 21
2.2 简单算法举例 21
2.3 算法的特性 22
2.4 怎样表示一个算法 23
2.5 结构化程序设计方法 29
2.6 面向对象的程序设计 32
思考题 37
第 3 章 Visual Basic 编程基础 38
3.1 由一个简单的程序说起 38
3.2 变量与常量 39
3.3 数据类型 44
3.4 Visual Basic 运算符与表达式 51
3.5 Visual Basic 常用内部函数 55
3.6 基本语句 59
思考题 60
第 4 章 数据的输入输出 61
4.1 数据的输出 61
4.2 数据的输入 75
4.3 基本控件 78
思考题 89
第 5 章 基本程序结构 90
5.1 选择结构 90
5.2 循环结构 106
5.3 其他辅助控制语句 122
思考题 124
第 6 章 数 组 125
6.1 数组的概念 125
6.2 数组的声明与数组元素的使用 126
6.3 静态数组与全局数组 128
6.4 动态数组 129
6.5 数组元素的有关操作 130
6.6 数组在程序中的应用 135

6.7 控件数组与对象数组 145
思考题 152
第 7 章 过程与模块 153
7.1 过程的概念 153
7.2 过程的定义与使用 153
7.3 参数传递 160
7.4 标准模块 167
7.5 作用域 167
7.6 简单递归程序设计 170
思考题 172
第 8 章 键盘与鼠标事件过程 173
8.1 键盘事件 173
8.2 鼠标事件 180
思考题 191
第 9 章 文 件 192
9.1 文件访问模式 192
9.2 文件基本操作 193
9.3 文件系统的管理 202
9.4 文件系统控件 204
9.5 文件系统对象 207
思考题 215
第 10 章 高级界面设计 216
10.1 窗体 (Form) 与框架 (Frame) 216
10.2 菜单 (Menu) 229
10.3 工具栏 236
10.4 对话框 (DialogBox) 240
10.5 多重窗体与多文档界面 (MDI) 249
10.6 综合应用 253
思考题 267
第 11 章 图形程序设计 268
11.1 坐标系 268
11.2 绘图属性 272
11.3 图形控件 276
11.4 图形方法 279
思考题 286
第 12 章 数据库程序设计 287
12.1 数据库的概念 287
12.2 数据控件 289
12.3 使用 ADO 控件访问数据库 301
12.4 结构化查询语言 (SQL) 317
思考题 326
第 13 章 Visual Basic 程序调试环境 327
13.1 编译时期的错误 (Compile Errors) 327
13.2 执行时期的错误 (Run-Time Errors) 328
13.3 逻辑错误 329
13.4 调试基本过程 335
思考题 335

第1章 Visual Basic程序设计入门

程序设计课程经历过很大的发展变化——课程地位从早期的计算机专业的基础课发展到今天非计算机专业普遍开设的公共基础课；语言平台从面向过程的 Basic、Fortran、Pascal 到 C 语言，再到面向对象的 VC、VB、JAVA 等；实践教学从完全的“纸上谈兵”（只上理论课）到安排一定的实验课（但实验一般处于附属地位，目的还是为了学习语言）。直到 90 年代中期，程序设计课程才奠定了它做为高校一门重要的计算机基础课程的地位，其根本原因在于课程不限于学生学习计算机语言，更重要的是对学生程序设计能力和思想进行培养。

那么，什么是程序设计呢?

1.1 程序设计概述

关于程序设计，大家首先必须了解一个著名的公式：

程序设计 = 数据结构 + 算法

按照通常的定义，数据结构即非数值计算的程序设计问题中的计算机操作对象以及它们之间的关系和操作；算法是对特定问题求解步骤的一种描述，是一组指令的有序序列。

程序设计就像盖房子,数据结构就像砖瓦,而算法就是设计图纸。你若想盖房子首先必须有原料(数据结构)，但是这些原料不能自动地盖起你想要的房子，你必须按照设计图纸（算法）上的说明一砖一瓦地去砌，这样你才能拥有你想要的房子。程序设计也一样，你使用的编译工具（如 Java、C、Basic、Pascal 等）中有各种功能语句（如 Read、Write）或基本结构（如 Integer、Single、Boolean 等），它们不会自动排列成你要的程序代码。你得按照程序规定的功能去编写，而程序的功能是实现（算法）的具体体现。所以通俗地说，你必须按照特定的规则，把特定的功能语句和基本结构按照特定的顺序排列起来，形成一个有特定功能的程序，这就是：程序设计 = 数据结构 + 算法。

数据结构是程序设计这座大厦的基础，没有基础，无论设计有多么高明，这座大厦不可能建造起来的。算法则是程序设计的思想、灵魂！没有灵魂的程序不能叫程序，只是一堆杂乱无章的符号而已。

数据结构内容有系统自带的几个基本结构（顺序结构、分支结构、循环结构、函数过程）、数据类型（整型、实型、布尔、字符等）和用户定义的更高级的数据结构 [数组、集合、文件、指针（队列、栈、树、图等）……]。可是算法却不同了，它是多种多样的！它可以让数据以你想要的方式排列（当然要符合语法和功能要求）。打个比方，数据结构是人体的各种组织、器官，算法则是人的思想。你可以用你的思想去支配你身体的各个可以运动的器官随意运动。如果你想去取一个苹果，你可以走过去，也可以跑过去，只要你想，你甚至可以爬过去。但是无论如何，你的器官还是你的器官（没有变），目的还是同一个目的（取苹果），而方式却是多种多样的！这就是算法的灵活性、不固定性。因此可以这

样说：数据结构是死的，而算法是活的！

理解了数据结构的在程序设计中重要性，也就明白努力学习好数据结构是学好程序设计的第一步，也是关键的一步。

算法是程序设计的重点，也是难点。难在学会了数据结构，似乎也知道该怎么样用，可是到了编程时，还是不知道该怎样用！其实，这就是缺少算法思想的一个具体表现。并不是学好了数据结构就会编程了。算法是灵活多变的，它不像数据结构那样有固定的形式。算法的不固定性就决定了它的特点，你不可能把所有的算法都学会，因为它是无穷尽的。但是你要是能把自己学的有限的算法融会贯通了，那就很了不起了。就好象你只学会了 1、2、3 这三个数，但是你可以组成 12、23、32、123……。

怎样学好算法呢?

1.2 程序设计方法

学习程序设计的方法不少，在此详细的给大家举几个例子。

首先，你要养成一个良好的程序设计风格和习惯。这也分许多种，其中最重要的是养成自上向下、逐步求精的程序设计思想和方法。怎样才是自上向下逐步求精的程序设计呢？举个例子。

【例 1.1】已知三角形的两边及夹角，求第三边及面积。

现在，当你看到这个问题时，你不要立即下手去编写，而是要思考一下怎样才能求解（算法），然后把它写下来以免忘记。于是就写了：

一级算法：

（1）输入两边的长度 a、b 以及夹角 α；

（2）计算第三边 c 和面积 s；

（3）输出 c 以及 s；

当你写完之后，你若觉得第 2 步不太清晰，还不能写出代码，那就继续求精。

二级求精：

（2.1）将 α 的角度转换为弧度；

（2.2）写出计算第三边的公式：$c=\sqrt{a^2+b^2+2ab\cos(\alpha)}$；

（2.3）写出计算机面积的公式：$s=\frac{1}{2}ab\sin(\alpha)$

有了二级求精，问题的解决已经很明显了，接着就可以轻松地编代码了。再仔细看看上面的步骤：先写出一级算法，如果觉得由此还是得不到源代码，或者是其中有一步还有很多步骤要写，那么就继续求精，直到你认为可以很清晰地写出代码为止。但是求精可以因人而异，像上面的题目，若对一个编过很多程序的程序员来说，可能不用什么求精，代码就可以直接写出来了；但要是一个初学者，就可能要写到二级求精。

可能有人认为这个求精是不是太慢，抑或根本就是在浪费时间。根本不是！这可是众多的编程先驱、编程高手们推荐的基本方法。你知道编程中，主要时间用在了哪里吗？不是编写代码，而是用在了求解算法和调试程序上。算法的求精就是缩短这些时间的最好的方法。它可以使你不用再很费力的苦思冥想到底该怎样构造算法使这个过程更条理、更简明；它也很可能使你不用再为了一个小小 BUG 而调试半天，因为，它给出的算法已经趋于完美。其实，列一个小提纲（算法）就可以解决很多的问题，

生活中不也是如此吗？所以，第一，求精算法这个最重要的习惯一定要养成！

还有一个好的习惯就是要做到程序源代码的清晰度与可读性一定要高。

这是为什么呢？在源代码中，每行代码前面空格的有无与多少并不影响程序的运行，但是它影响人的阅读。你想，当你看到一篇代码，格式杂乱无章，没有什么主次内外之分时，你一定会头痛不已，绝对看了第一遍就不想再看第二遍了。而且当你调试这样的代码时，也会被一个小错误弄得找不到北。如果格式很清晰的话，代码就非常易读，结构很清晰，begin…end 都看得清清楚楚。我想这样的代码阅读和调试起来一定很顺手的。

还有就是在有些重要的代码块后加入注释，也可以提高程序的可读性。很多原来的代码不加注释，我们当时对其结构很清楚，可是过了几个星期之后再看代码简直就像看另外一个人写的一样，要费不少工夫才能弄懂，代码越多越如此。自己还很难理解自己的代码，更不用说别人了。因此费点力气加上注释还是值得的。

其次，就是多加练习和多多与别人交流心得，研究和欣赏别人优秀的程序代码（上面说的清晰度与可读性的重要之处就体现在这里了）。无论干任何事情，只有经常练习才能熟练的掌握与应用。它不仅能帮你提高对算法的认知程度，还可以使你更好地理解数据结构；而交流心得与欣赏别人的代码则是要你学习别人优秀独特的见解与逻辑思维，许多好的程序员就是在研究优秀代码中慢慢成长起来的。但研究和欣赏并不是意味着让你全盘照搬，它是说既不要拘泥于自己一成不变的思考方式，也不要成为别人思想奔驰的高速路，而是让你取长补短，取其精华，去其糟粕。另外，在编译一个程序时，要多想几种不同的算法，反复比较，看看究竟哪个更适合这个程序。这样，不仅有助于提高你程序的运行效率，更重要的是有助于提高你对算法的认识和思维的广度。

通过以上介绍，你会发现程序设计的这半扇门似乎并不是想象的那么容易打开；但是如果你能耐住寂寞与枯燥的考验，按部就班、循序渐进地练习、思考，不知不觉中你的水平已经提高了很多。

除了上面说的求精算法和数据结构外，数学方法和逻辑思想对你的程序设计也会有一定的影响（这也属于算法），这也是这门课程需要培养的基本能力。

先说数学方法，它是指导你学习的一个重要助手。比如你要实现一个看似很复杂的函数，而且你编写了一大堆代码，什么结构和类型都用上了，可是还是实现不了。这时候先别着急，也许在你做这个函数前就要静下心来，仔细思考，看看是否有数学上学习的特别公式能够实现这个问题？这种思索的过程正如解数学题，浸淫其中，方知个中趣味。

【例 1.2】将 8 个“车”放在 8×8 的国际象棋棋盘上，如果它们两两均不能互吃，那么称 8 个“车”处于安全状态。问总共有多少种不同安全状态？

对于这样的题一般的解法是用“回溯法”，即先有顺序的固定一个棋子，然后将剩下的逐行试探，如果可以则摆放下一个，要是不行就把该棋子换一个位置，直到所有的方法都用一遍（遍历）。这样看来这个方法真的很麻烦，不仅需要编写大量代码，而且还要用到“递归”（编程的一个重要的算法）。这使程序的效率大大降低。但是我们可以想想有没有更好的数学方法。肯定有！你看在第一行中，棋子可以在 8 个空格内随意摆放而不会受到攻击；而第二行，棋子则只有在 7 个空格内再挑选一个了；第三行就只有 6 个……依次类推第 8 行就只剩下一个空格了。所以这个方法一共有 8×7×6×5×4×3×2×1 即 8！种解法，这不就是数学中的排列与组合的问题吗？若不进一步思考，盲目地逐个枚举方案，其笨拙是可想而知的了。所以在设计算法时，一定要仔细思考有没有更好的数学公式或思想与之相关联，这样往往会收到意想不到的好效果，这个过程我们称之为数学模型的建立。

再说哲学方法，哲学是所有学科的总和，当然也包括程序设计这门新兴的学科。哲学中的一小部分，

就是逻辑学。一个人若没有严格的逻辑思维，他（她）的程序一定有不少虫子(bug)。然而对于这些虫子，他（她）可能不知道，也很有可能知道了，但是就不知道哪里出错了。这就是因为他（她）没有把问题出现的情况（包括出错的情况）考虑全面而造成的。虽然理论上所有的程序都有一定的漏洞（尤其是代码越多的程序越是如此），但是我们可以尽我们的全力把问题考虑全面，使错误发生的几率降低到最小，以确保程序的稳定性。

比如有这么一个小程序，其功能是输入你的生日，然后再输入某天的日期，最后计算出你到这一天一共活了多少天。从算法而言很简单，但请别急着完成，写完后请测试如下数据：

生日：1989 1 28
某日：1899 1 28（也许你想输入的是 1999 1 28）

结果不言而喻，很多初学者都会遇到类似的问题。所以说，提高逻辑思维能力也是提高编程效率的很有效的方法。

世界上所有的学科都有联系。当数学与逻辑学擦出“爱情”的火花时，程序设计就有了雏形。其实还有其他的学科也会影响程序设计，比如英语，地球人都知道，现在绝大多数程序源代码是用英文编写的，而且很多开发工具、编译工具还有一些用来参考的小程序几乎都是英文界面。学好英语对代码的分析、对数据结构的了解以及对开发程序或软件的效率提高都有帮助。

综上，我们不妨将前面那个著名的公式作一个扩展：

程序设计 = 数据结构 + 算法 + 程序设计方法 + 语言工具和环境

难道语言工具和环境的选择和程序设计也有关系吗？

1.3 程序设计语言

自 1946 年世界上第一台电子计算机问世以来，计算机系统都是由硬件系统和软件系统两大部分构成的，硬件是物质基础，而软件可以说是计算机的灵魂，没有软件，计算机是一台“裸机”，是什么也不能干的，有了软件，才能灵动起来，成为一台真正的“电脑”。所有的软件，都是用计算机语言编写的。计算机程序设计语言的发展，经历了从机器语言、汇编语言到高级语言的历程。

1. 机器语言

电子计算机所使用的是由“0”和“1”组成的二进制数，二进制是计算机的语言的基础。计算机发明之初，人们只能降贵纡尊，用计算机的语言去命令计算机干这干那，一句话，就是写出一串串由“0”和“1”组成的指令序列交由计算机执行，这种语言，就是机器语言。使用机器语言是十分痛苦的，特别是在程序有错需要修改时，更是如此。而且，由于每台计算机的指令系统往往各不相同，所以，在一台计算机上执行的程序，要想在另一台计算机上执行，必须另编程序，造成了重复工作。但由于使用的是针对特定型号计算机的语言，故而运算效率是所有语言中最高的。机器语言，是第一代计算机语言。

2. 汇编语言

为了减轻使用机器语言编程的痛苦，人们进行了一种有益的改进：用一些简洁的英文字母、符号串来替代一个特定指令的二进制串，比如，用“ADD”代表加法，“MOV”代表数据传递等等，这样一来，人们很轻易读懂并理解程序在干什么，纠错及维护都变得方便了，这种程序设计语言就称为汇编语言，

即第二代计算机语言。然而计算机是不熟悉这些符号的，这就需要一个专门的程序，专门负责将这些符号翻译成二进制数的机器语言，这种翻译程序被称为汇编程序。汇编语言同样十分依靠于机器硬件，移植性不好，但效率仍十分高，针对计算机特定硬件而编制的汇编语言程序，能准确发挥计算机硬件的功能和特长，程序精炼而质量高，所以至今仍是一种常用而强有力的软件开发工具。

3. 高级语言

从最初与计算机交流的痛苦经历中，人们意识到，应该设计一种这样的语言，这种语言接近于数学语言或人的自然语言，同时又不依靠于计算机硬件，编出的程序能在所有机器上通用。经过努力，1954 年，第一个完全脱离机器硬件的高级语言—FORTRAN 问世了，40 多年来，共有几百种高级语言出现，有重要意义的有几十种，影响较大、使用较普遍的有 FORTRAN、ALGOL、COBOL、BASIC、LISP、SNOBOL、PL/1、PASCAL、C、PROLOG、Ada、C++、VC、VB、DEPHI、JAVA、C# 等。高级语言的发展也经历了从早期语言到结构化程序设计语言，从面向过程到非过程化程序语言的过程。相应地，软件的开发也由最初的个体手工作坊式的封闭式生产，发展为产业化、流水线式的工业化生产。60 年代中后期，软件越来越多，规模越来越大，而软件的生产基本上是各自为政，缺乏科学规范的系统规划与测试、评估标准，其恶果是大批耗费巨资建立起来的软件系统，由于含有错误而无法使用，甚至带来巨大损失，软件给人的感觉是越来越不可靠，以致几乎没有不出错的软件。这一切，极大地震动了计算机界，史称“软件危机”。人们认识到：大型程序的编制不同于写小程序，它应该是一项新的技术，应该像处理工程一样处理软件研制的全过程。程序的设计应易于保证正确性，也便于验证正确性。1969 年，提出了结构化程序设计方法，1970 年，第一个结构化程序设计语言—Pascal 语言出现，标志着结构化程序设计时期的开始。 80 年代初开始，在软件设计思想上，又产生了一次革命，其成果就是面向对象的程序设计。在此之前的高级语言，几乎都是面向过程的，程序的执行是流水线似的，在一个模块被执行完成前，人们不能干别的事，也无法动态地改变程序的执行方向。这和人们日常处理事物的方式是不一致的，对人而言是希望发生一件事就处理一件事，也就是说，不能面向过程，而应是面向具体的应用功能，也就是对象（Object）。其方法就是软件的集成化，如同硬件的集成电路一样，生产一些通用的、封装紧密的功能模块，称之为软件集成块，它与具体应用无关，但能相互组合，完成具体的应用功能，同时又能重复使用。对使用者来说，只关心它的接口（输入量、输出量）及能实现的功能，至于如何实现的，那是它内部的事，使用者完全不用关心，C++、VB、Dephi 就是典型代表。高级语言的下一个发展目标是面向应用，也就是说：只需要告诉程序你要干什么，程序就能自动生成算法，自动进行处理，这就是非过程化的程序语言。

在众多的面向对象的软件开发工具中，选择一门适合于初学者学习程序设计的语言工具是很重要的。根据对计算机专业知识需求的不同，一般按专业和非专业划分，针对计算机专业，普遍选择 C++ 系列、Java 等作为学习和开发工具，而非计算机专业通常选用 VB、VF 等作为学习和开发工具。其实，作为同一时期的各种语言开发工具（如 VC、VB、Dephi、PB、VF、Java 等），并没有能力高下的定论，各有优缺点，只有适合不适合的选择。本书选择 VB，正是是根据 VB 的特点作出的。

VB 作为第一个可视化（Visual）的集成开发环境（IDE），以其快节奏高效率的程序建立吸引着用户，进行 Windows 编程可以像搭积木一样方便。作为初学者可以把时间更多地用在程序核心功能的算法设计上面，这也是大多数非计算机专业人员在建立自己的应用程序时的首选。

另外，VB 支持大量的第三方控件，通过对这些控件方便地使用，完成一个商业化的程序并不是一件困难的事。而通常这些控件的编写都是由专业的计算机软件人员完成的，术业有专攻，一般的非计算机专业人员没必要也没时间精力去做这些偏底层的工作，用好已有的优秀控件更具现实意义。

最后，本课程是程序设计的入门学习，VB 仅是众多语言工具中的一种，掌握程序设计思想才是关键，此理通，对于其他语言工具的学习也将事半功倍，使用其它工具进行编程也能很快上手。

那么，VB 究竟拥有怎样一个开发环境呢？我们真的能用该工具快速完成我们的第一个程序吗？下一节我们将通过实际的例子，带你一步步操作，进入 VB 的编程世界。

1.4 第一个Visual Basic程序

所谓 VB，就是 Visual Basic 的简称。首先，它是基于 Basic 语言的；其次，作为开发工具，它具有 Visual——可视化的特点。什么是可视化？我们将从 Visual Basic 的外貌谈起。

1.4.1 Visual Basic 外貌

首先参照附录安装好本书使用的 Viusal Basic 6.0。然后如图 1.4.1 所示，从“开始”菜单启动 Microsoft Visual Basic 6.0。

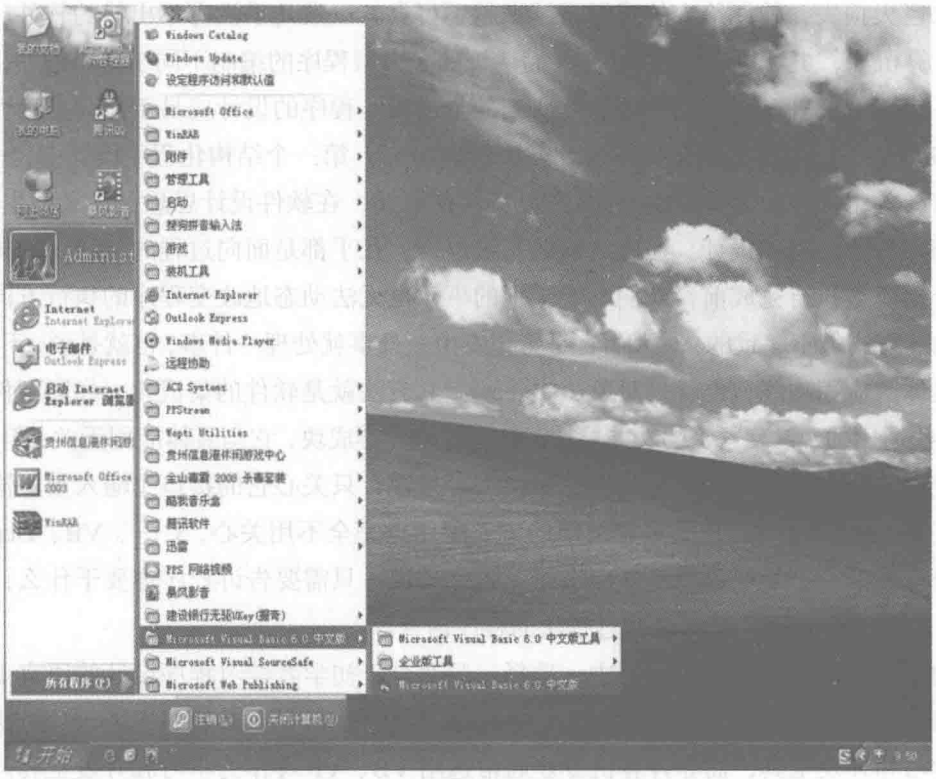

图 1.4.1 启动 Visual Basic

Visual Basic 启动后，首先映入眼帘的，将是个名为“新建工程”的对话框，如图 1.4.2。

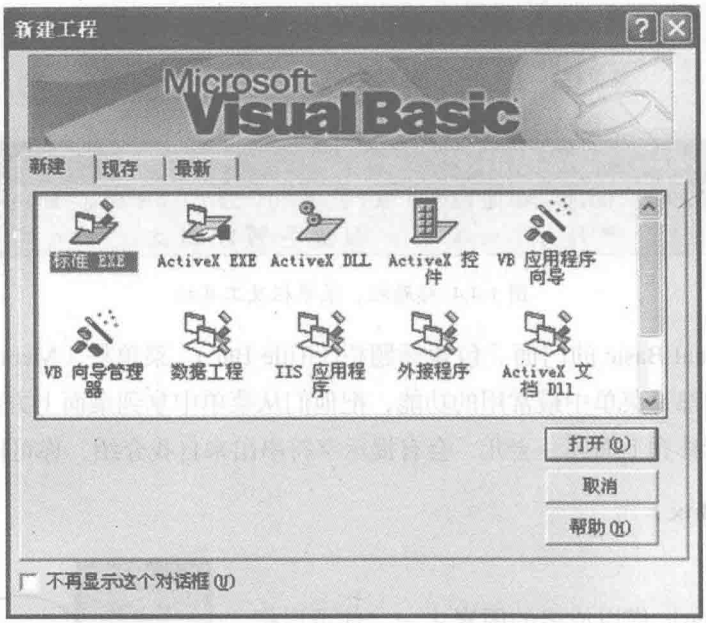

图 1.4.2“新建工程”对话框

每一个 Visual Basic 的程序都有一个动听的名字，叫“工程（Project）”。我们可以使用 Visual Basic 来开发各种不同类型的工程，每种工程依照特性不同，写程序的方法也有些许的差异，这个对话框会依程序员的回答，准备一些必须或可能需要的文件。作为初学者，先不管其它选择，直接选第一项——标准 EXE。

按下右下角的“打开“之后，我们将进入 Visual Basic 的集成开发环境（IDE）。如图 1.4.3。

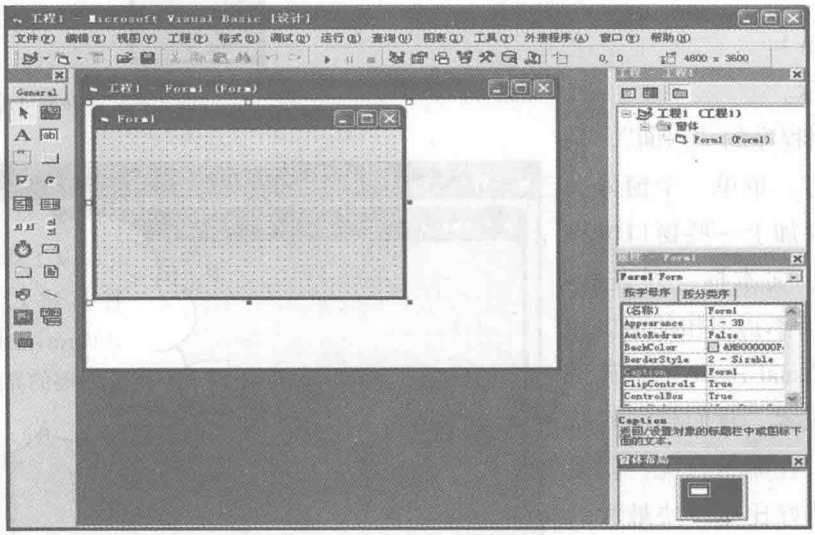

图 1.4.3 Visual Basic 的集成开发环境

IDE 里面有很多有用的工具，该界面下我们可以看到主要的五大基本模块。学习 Visual Basic 的集成开发环境，必须首先弄清楚这五大基本模块的作用。

1. 标题栏、菜单栏及工具栏

如图 1.4.4 所示。

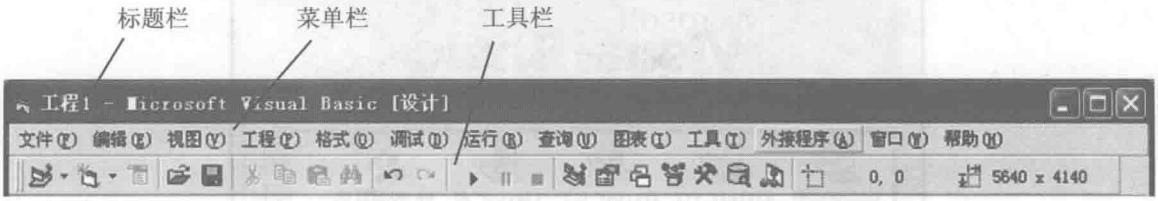

图 1.4.4 标题栏、菜单栏及工具栏

该模块可称为 Visual Basic 的门面。包含标题栏（Title Bar）、菜单栏（Menu Bar）与工具栏（Tool Bar）。其中工具栏通常都是菜单中最常用的功能，把他们从菜单中拿到桌面上方便使用的。工具栏有很多浮动式按钮，把鼠标移到上面停一会儿，会有提示字符串出来自我介绍，你可以大致了解其功能。

2. 工具箱（Tool Box）

如图 1.4.5 所示。

该模块是 Visual Basic 借以成名的因素之一。你可以把它称作积木盒，因为它让我们写程序跟堆积木一样，比如我们希望程序有一个按钮，你只需要到工具箱里找到按钮这个部件，取出来摆上去就行了。工具箱里的“积木”——控件，都是功能完整的部件，程序员可以利用他们“拼出”程序界面。写程序好比是根据程序员的创意拿积木盖房子，再依靠对属性的设置，可以帮助我们设计出许多美观实用的窗口（Windows）程序。

图 1.4.5 工具箱（Tool Box）

3. 窗体（Form）

如图 1.4.6 所示。

该模块是我们程序的“门面”，是与用户沟通的地方。单单一个窗体没有什么，可是如果加上一些窗口世界的部件（如按钮、文本框、列表框、标签等），它就成了不折不扣的窗口程序了。如果说在 Visual Basic 的世界里写程序好比是堆积木，我们到工具箱里找积木来组合，在哪组合呢？就是放到窗体里。窗体好比是一块最大的积木，是其它积木的底。当我们把一些按钮、列表框、文本框“放”在窗体里“组合”起来后，就形成了标准的 Windows 对话框。

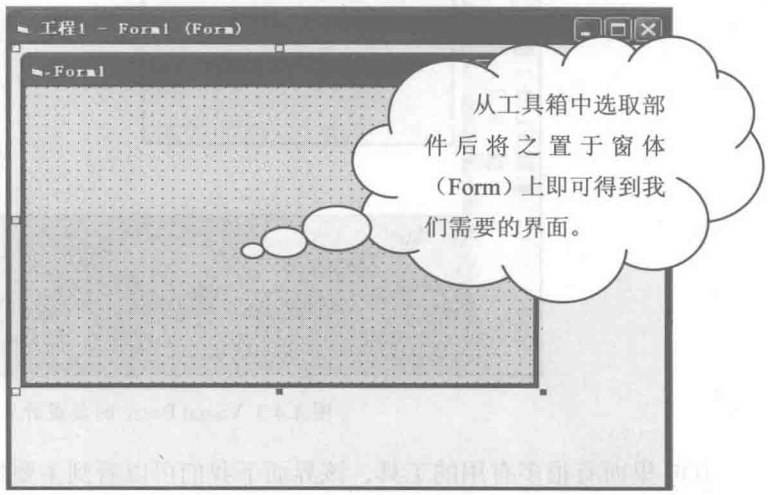

图 1.4.6 最开始的窗体

4. 属性窗口（Property Window）

如图 1.4.7 所示。

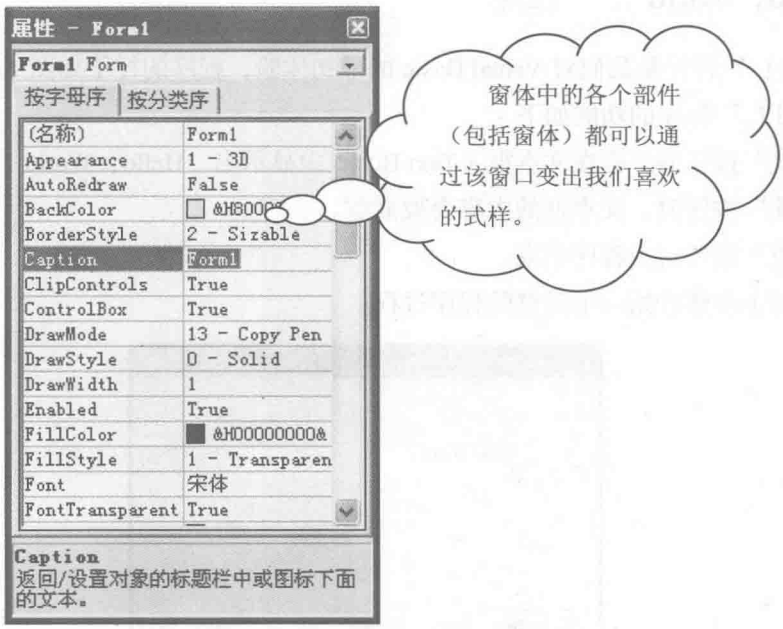

图 1.4.7 属性窗口

该模块可以使我们的窗口程序界面更加丰富多彩。当我们把工具箱里的积木摆放到窗体上后，基本的程序界面已经形成，但如果积木的样子我们不喜欢怎么办？（比如它原来是灰色的，而我喜欢绿色）这时候，就是属性窗口进行工作的时刻了。我们可以通过它设置积木的样式、功能及运作方式。换句话说，它可以为我们把积木变成我们喜欢的样子，并加上我们喜欢的功能。

细心的读者会注意到，在属性窗口的下方，也就是开发环境的右下角还有一个称为“窗体布局”的功能模块，我们可以通过该模块调整窗口在屏幕上的位置，我们将在后面的实例中为大家介绍。

5. 工程管理窗口（Project Window）

如图 1.4.8 所示。

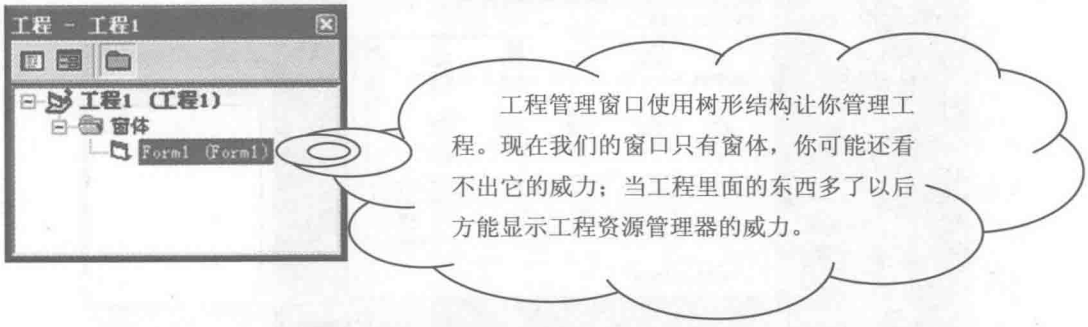

图 1.4.8 工程管理窗口

该模块是我们程序的“管家”。我们已经知道，每个 Visual Basic 程序都叫做一个工程，这是因为窗口程序本身相当复杂，一个工程中可能包含了我们进行程序编写时所需的各种功能、各式各样的文件，如若干的窗体、模块文件等，工程管理窗口能帮助程序员方便地管理，同时方便程序员在各种文件间进行切换。

下面我们就学习用 Visual Basic 的集成开发环境（IDE）写出第一个窗口程序——Hello，World！

1.4.2 “Hello，World！”程序

“Hello，World！”程序是我们对 Visual Basic 的最初体验，程序的样子如图 1.4.9 所示。

“Hello，World！”程序的功能如下：

（1）按下“问候”按钮时，会在文本框（Text Box）中显示出“Hello，World！”。

（2）按下“清除”按钮时，文本框的内容会被清空。

（3）按下“结束”按钮时，程序结束。

下面我们按照如下步骤开始一个完整的程序过程。

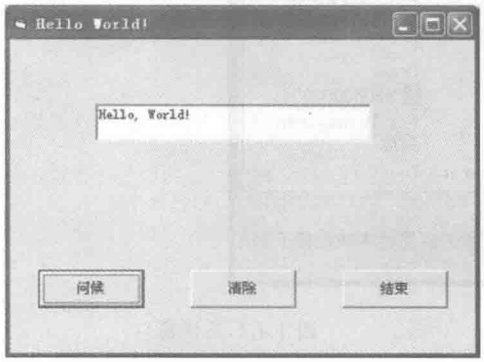

图 1.4.9 “Hello，World！”程序

1. 建立一个新的工程（Project）

（1）从菜单栏（Menu Bar）中选择“文件 / 新建工程”。Visual Basic 会先问你：“想建立什么样的工程呢？”如图 1.4.10 所示。

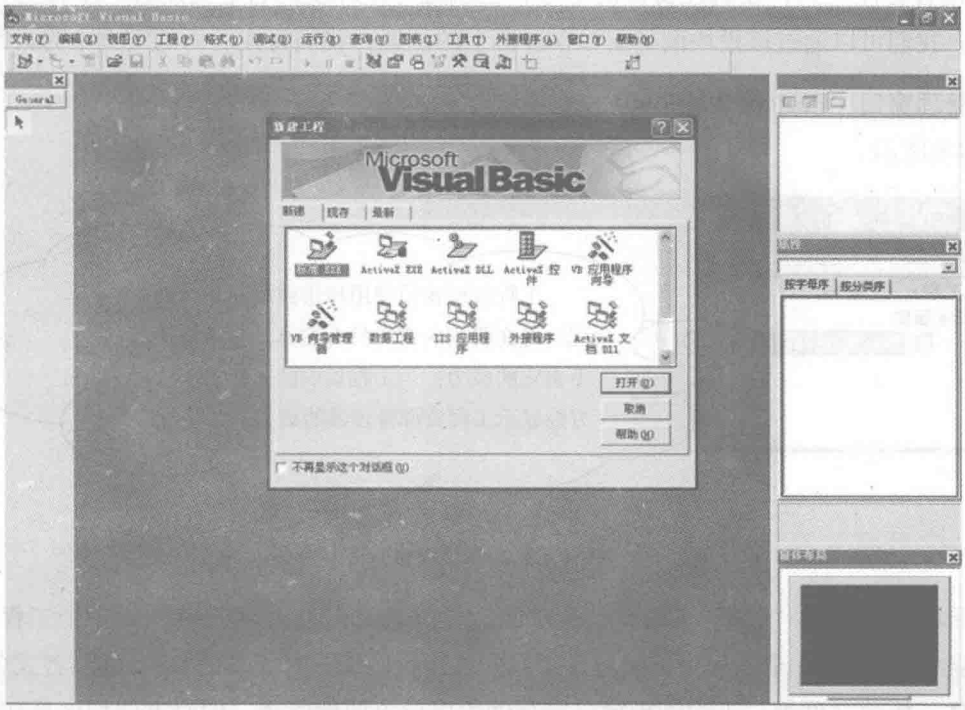

图 1.4.10 新建工程