高等院校计算机应用技术规划教材

应用型教材系列

C#任务导引教程

谢书良　主编

C# RENWUDAOYINJIAOCHENG

清华大学出版社

高等院校计算机应用技术规划教材·应用型教材系列

C#任务导引教程

谢书良　编著

清华大学出版社
北京

内 容 简 介

本书是为从未学习过编程又期望能掌握基本编程的读者编写的入门教材。

全书共分 9 章，第 1～4 章介绍了结构化编程，第 5 和 6 章介绍了面向对象的编程；第 7～9 章介绍了可视化编程，内容涵盖了程序设计课程的主要内容。第 9 章是体现本书特色的一章，介绍了多数据表的“学生成绩管理系统”的设计过程和完整代码，为学生进行“课程实践”提供了一个完整的可视化程序设计的工程样例。

本书按任务导引教学方法进行编写，十分注重可读性和可用性。用任务的具体实践来加深基础知识的学习，既保持了知识的系统性，又使学习目的比较明确，学习效果容易检验，在激发读者学习程序设计应用知识和训练程序设计能力方面有较好的作用。本书还为教师提供了精心设计的配套电子课件、全部例题源代码、自测练习题答案和部分题目的源代码。

本书可作为高等院校相关专业开设“程序设计”课程或“程序设计工程实践”课程的教材，也可作为工程技术人员的参考用书和对程序设计有兴趣的初学者的自学用书。

图书在版编目(CIP)数据

C＃任务导引教程/谢书良编著. —北京：清华大学出版社，2012.1
(高等院校计算机应用技术规划教材. 应用型教材系列)
ISBN 978-7-302-27250-2

Ⅰ. ①C…　Ⅱ. ①谢…　Ⅲ. ①C 语言－程序设计－高等学校－教材　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2011)第 226817 号

责任编辑：张龙卿(sdzlq123@163.com)
责任校对：李　梅
责任印制：杨　艳
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**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
印 刷 者：北京四季青印刷厂
装 订 者：三河市兴旺装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：18　**字　数**：431 千字
版　　次：2012 年 1 月第 1 版　　**印　　次**：2012 年 1 月第1 次印刷
印　　数：1～3000
定　　价：38.00 元

产品编号：044688-01

前　言

当前，IT 技术迅猛发展，日新月异。在计算机应用日益广泛的形势下，软件的概念和程序设计的应用知识已逐渐成为人们渴求的新目标。如果说数学是“培养抽象思维的工具”，物理学是“培养逻辑思维的工具”，那么，程序设计则是“培养计算思维的工具”。有人预言，到 2050 年“计算思维”将成为全人类的主要思维方式，“计算思维”的精髓是“程序思维”，仅从此点而言，对于绝大多数理工类的高校学生来说，学一点程序设计基础和应用知识十分必要。

程序设计语言很多，本书之所以选择 C# 语言作为零起点的程序设计入门语言是因为：C# 是微软公司发布的由 C 和 C++ 衍生出来的运行于 .NET Framework 之上的一种安全、稳定、简单、规范、完全面向对象的编程语言。C# 凭借其强大的操作能力、优雅的语法风格、创新的语言特性和便捷的面向组件的编程，不仅成为 .NET 开发的首选，也将会成为教学程序设计语言课程的优选，关键在于教材的编写是否能真正体现零起点，真正能适合初学者作为入门教材使用。

编写本书就是基于这一初衷，期望能为只安排一个学期教学程序设计课程的有关本科、专科学生和对编程有兴趣的初学者提供一本真正零起点的入门用书。

本书采用“任务导引法”思路进行编写，即在教师的具体指导之下，引发学生的学习兴趣和学习动机，这样既有利于维护教学内容的体系，也便于检测教学进程的效果。

本书共分 9 章，内容分别为 C# 语言概述，运算符和表达式，程序控制结构，数组和方法，类的封装性，类的继承性和多态性，可视化编程基础，数据库操作和信息管理系统的设计。本书可用 64 课时或更少课时完成。如果课时有余量，可以安排“课程设计”环节，以达到“学以致用”的效果。

“多思考，勤上机”是学好程序设计语言的重要条件，学习编程要细心、耐心并要有恒心，只有那些有志气、有毅力的人，才能饱尝编程带来的愉悦。

本书的编写是顺应程序设计语言发展历史潮流的一种新的尝试，虽经多次教学试用，但仍会存在不足之处，诚盼不吝指正，使其不断完善。

谢书良

2011 年夏于北京

目　录

第1章　C#语言概述

【教学要求】

1. 了解程序及程序设计的概念。
2. 了解面向对象程序设计的基本特点。
3. 掌握C#程序的基本结构和程序运行的流程。
4. 熟悉C#语言开发环境的简易使用。

1.1　基本概念

1. 程序

本书从如何计算两个数的平均值这样一个最简单的问题讲起。

如果这两个数是3和5，几乎可以不假思索地说出它们的平均值是4。

如果这两个数是23763965432和8456234445446456，它们的平均值是多少？那只能由计算机去完成。

不管怎么算，人和计算机的计算步骤都是：

(1) 要计算的是哪两个数。

(2) 先求出两个数之和。

(3) 再将此和除以2。

(4) 最后报告计算结果。

其实计算机自身并不会计算，必须由人来教会它。那么人们应该做什么呢？就一般的问题来说，人们要做的事应该是：针对要完成的任务，编排出正确的方法和步骤，并且用计算机能够接受的形式，把方法和步骤告诉它，指挥计算机完成任务。

解决问题的方法和步骤，以计算机能够理解的语言表达出来，就称为“程序”。程序是要计算机完成某项工作的代名词，是对计算机工作规则的描述。

计算机软件是用来指挥计算机硬件的，没有软件，计算机就什么事也做不了。而软件都是由各种程序构成的，程序是软件的灵魂。

2. 程序设计

人们要利用计算机解决实际问题，首先要按照人们的意愿，借助计算机语言，将解决问

题的方法、公式、步骤等编写成程序，然后将程序输入计算机中，由计算机执行这个程序，完成特定的任务，这个设计和书写程序的整个过程就是程序设计。简言之，为完成一项工作的规则的过程设计称为程序设计。从根本上说，程序设计是用人的智力来解决复杂的客观问题的过程。

程序设计是根据给出的具体任务，编制一个能正确完成该任务的计算机程序。计算机程序是有序指令的集合，或者说是能被计算机执行的具有一定结构的语句的集合。

图 1-1 所示的是一个简化了的计算机工作过程示意图，计算机的实际工作过程当然比这复杂得多，但它还是完整地体现了其基本工作原理，尤其体现“软件指挥硬件”这一根本思想。在整个过程中，如果没有软件程序，计算机什么也干不了，可见软件程序多么重要。如果软件程序编得好，计算机就运行得快而且结果正确；程序编得不好，则可能需要运行很久才出结果，且结果还不一定正确。程序是软件的灵魂，CPU、显示器等硬件必须由软件指挥，否则它们只是一堆没有灵性的工程塑料与金属的混合物。下面就是要教会读者怎样用编程语言又快又好地编写程序（软件）。

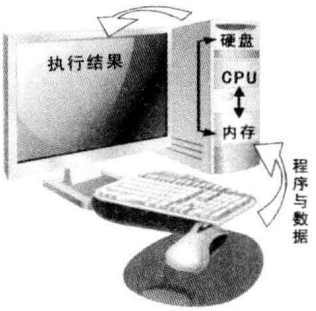

图 1-1

计算机直接能够读懂的语言是机器语言，也叫做机器代码，简称机器码。这是一种纯粹的二进制语言，用二进制代码来代表不同的指令。

下面这段程序是用我们通常使用的 x86 计算机的机器语言编写的，功能是计算 1+1。

```
10111000
00000001
00000000
00000101
00000001
00000000
```

这段程序看起来像“天书”，在用按钮开关和纸带打孔的方式向计算机输入程序的时代，程序员编写的都是这样的程序。很明显，这种程序编起来很费力气，很难读懂。从那时候起，让计算机能够直接懂得人的语言就成了计算机科学家们梦寐以求的目标。

有人想出了这样的办法，编一个可以把人类的语言翻译成计算机语言的程序，这样计算机就能读懂人类语言了。这说起来容易，做起来难。就拿计算 1+1 来说，人们可以用“1+1 等于几”、“算一下 1+1 的结果”、“1+1 得多少”等多种说法，再加上英语、法语、日语、韩语、俄语等来描述。如果想把这些都自动转换成上面的机器码，是可望而不可即的事。所以人们退后一步，打算设计一种中间语言，它还是一种程序设计语言，但比较容易翻译成机器代码，且容易被人学会和读懂，于是诞生了“汇编语言”。

用汇编语言编写计算 1+1 的程序如下所示：

```
MOV AX , 1
ADD AX , 1
```

这个程序的功能是什么呢？从程序中 ADD 和 1 的字样，或许我们能猜个大概。没错，它还是计算 1+1 的。这个程序经过编译器（也是一个程序，它能把 CPU 不能识别的语言翻

译成CPU能直接识别的机器语言)编译,就会自动生成前面的程序。这已经是很大的进步了,但并不理想。这里面的MOV是什么含义?好像是Move的缩写。这里的AX又代表什么?这是一个纯粹的计算机概念。从这个小程序我们能看出,汇编语言虽然已经开始贴近人类的语言,但还全然不像我们所期望的那样,里面还有很多计算机固有的东西必须要学习。它与机器语言的距离很近,每行程序都直接对应上例的三行代码。以后如果有机会学习、使用汇编语言,到那时你将学到更多有关计算机内部的知识。

程序设计语言要无限地接近自然语言,所以它注定要不停地发展。此时出现了一道分水岭,人们把机器语言和汇编语言称为低级语言,把以后发展起来的语言称为高级语言。低级语言并不比高级语言"低级",而是说它与计算机(硬件)的距离较近因而级别比较低。高级语言高级到什么程度呢?我们先介绍一下很著名的BASIC语言,看它是怎样完成1+1计算的。

用BASIC语言计算并显示1+1的内容如下:

```
PRINT  1+1;
```

英文PRINT的中文意思是打印。比起前两个例子,它确实简单了不少,而且功能很强。前两个例子的计算结果只保存在CPU内,并没有输出给用户。这个例子直接把计算结果显示在屏幕上,它才是真正功能完备的程序,相信从这个例子你已经体会到了高级语言的魅力。

【阅读材料】 目前较流行的几种高级语言简介

因为高级语言易学、易用、强大,所以它发展很快,其种类之多完全可以用"百花争艳"来形容。据一位民间人士耗时多年的不完全统计,目前已经有超过2500种计算机语言,其中绝大多数都是高级语言。

BYTE杂志创刊20周年特刊里一篇题为"程序设计语言发展简史"的文章中,列举了从1946年起到1995年为止,在社会上产生一定影响的程序设计语言约有40多种,实际上目前广泛流行的只有几种。图1-2是日本计算机教育家三田典玄先生提供的一幅这几种程序设计语言与软件、硬件、系统、用户四个方面的关系图。

由图中不难看出,BASIC语言正好处于中间位置,它兼顾了软件、硬件、系统和用户四个方面的要求,几乎成了其他任何一种语言都无法替代的语言。几种常用的高级语言,在漫长的发展过程中互相渗透、互相借鉴,逐步形成了你中有我、我中有你的局面,其中变化最大的当属BASIC语言,它最初是从FORTRAN语言脱胎而来,以后又陆续吸收了其他语言的诸多特点。国外一些计算机教育家早已注意到了这一点,他们发现已掌握BASIC的人可以很快地学会任何一门(哪怕是很晦涩的)程序设计语言。都公认BASIC语言是易学、易用、发展快、变化大、覆盖面广的初学者的入门语言。美国、日本等发达国家都选择BASIC作为程序设计课程的入门语言,不无道理。我国在20世纪80年代以前,也几乎都是选择BASIC语言作为程序设计课程的入门语言。

到了20世纪90年代,BASIC逐步被QBASIC所替代。QBASIC是纯粹的面向过程的解释型语言,内容又过于烦琐,特别是它不能编译,从而不能脱离系统运行,使用十分不便。当前,在我国几乎清一色地选择了C语言作为程序设计课程的入门语言。C语言是面向过程的编译型语言,从图1-2中不难看出,C语言所处的位置是偏向硬件和系统的,由于C语

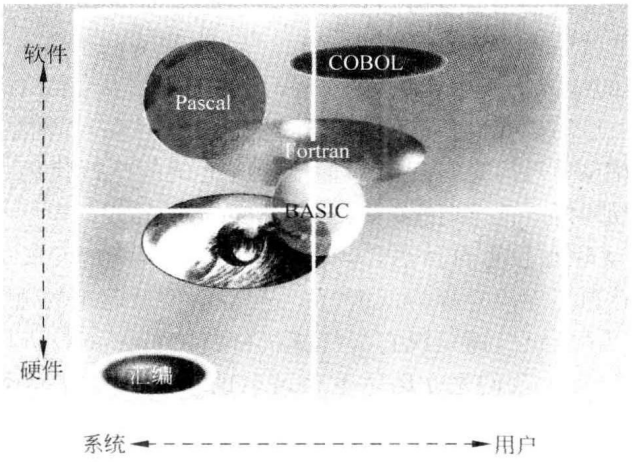

图 1-2

言距离机器语言比其他高级语言更近，因此也有人把C语言看成唯一的介于低级语言和高级语言之间的中级语言。它距离自然语言比较远，其思维还是顺应计算机而不是顺应人的，所以学起来比较费劲，用起来也不简便，虽然很适合程序员使用，但对于缺乏计算机语言基础的程序设计课程的初学者来说，其难度之大可想而知。

在当今世界，C语言除其固守的系统软件开发阵地和历史延续下来的大型软件外，几乎仅在小型的且追求运行效率的软件和嵌入式软件开发方面还有一定空间。况且，就这一点点空间也正在逐渐缩小。取而代之的是C++，Java和C#以及一些很专门化的语言。

C++从名字上看就知道其与C语言有着很深的渊源。C++几乎完全兼容C语言的语法，且所有流行的C++编译器，都能编译C程序，所以很多人认为C++就是C语言的升级。在很多场合，它们也被放在一起，称为C/C++。这个“++”里第二个加号加上的，便是大名鼎鼎的“面向对象”。

在面向对象被广泛接受之前，流行的是结构化程序设计思想。C、Pascal等语言都是结构化的。面向对象的程序设计思想对程序设计的影响可谓翻天覆地，整个计算机界的思维都因其而改变，程序设计更加贴近现实世界，更加符合人类固有的习惯。C++借助C语言的影响力，携面向对象的威力，其份额迅速增大。在20世纪90年代，曾有民间组织统计当年发布的软件中，有90%以上都采用C++语言开发。

C++兼有C语言的能力，也是一种“什么都能干”的高级语言，完整地学习和使用起来也不简单。Java语言就是在这种情况下诞生的。Sun公司推出的Java语言，以面向对象、与平台无关和易学易用而著称。它全面照搬了C++语法，并去掉了其不常用和不成功的部分，化繁为简，迅速博得了程序员们的认可，获得了越来越多的支持。Java语言的市场做得很不错，它已经被广泛接受，以至于很多客户都指名道姓地要求开发商必须用Java语言为其开发软件，尽管他们并不十分明白Java语言究竟如何。

Java语言的成功，让软件大鳄微软公司(Microsoft)坐不住了。它及时推出了C#(读作CSharp)语言。“#”这个符号就是四个加号围坐一圈，所以有人戏称其为C++++。从名字可以看出它与C/C++的渊源。不错，它也吸收了C/C++的语法。

C#语言诞生在Java语言之后，由原设计Java语言团队的Borland公司的Anders Hejlaberg转到微软公司后担任C#的首席设计师，所以他能把Java语言的成功之处吸收进来，把不成功之处抛弃，打造了一个似Java而非Java，并超越Java的语言。C#语言于2002年诞生(Java语言诞生于1995年)，由于其简洁、安全、高效，功能强大，兼容性好，完全面向对象，其前途不可估量。C#是专门为与微软公司的.NET Framework一起使用而设计的编程语言之一，有数据统计，在.NET中各种语言的使用比例，J#占1%，VC++占4%，VB占5%，而C#占90%，可见其应用之广泛，地位之重要。

目前，C/C++、Java和C#语言即将形成三足鼎立之势，其他语言仅能在其专属领域里得以发挥作用。同时，C++、Java和C#语言也在不断地完善、扩充自身，极力挤压其他语言的空间。

每一种语言都有其内涵，那是它独特的神韵。已经完全陶醉于一种语言的人，很难领会、接受另一种语言的神韵，这就造成了学习新语言的障碍。其实，学习语法并不重要，领会其神韵才是最重要的。各种语言的语法都大同小异，就算互不抄袭，也基本难以逾越几条普遍规则。只要熟练掌握一种语言的语法，再学任何其他语言的语法就简单多了。但要真正掌握一种编程语言的神韵，并不能一蹴而就。

在介绍了目前流行的程序设计语言之后，仍旧回到对初学者来说究竟选择哪种语言作为入门语言更好的问题。从现实情况来看，由于在学习高级语言程序设计入门课程之前，绝大多数人都缺乏计算机语言基础，不少学生学了一个学期的C语言后，脑子里仍稀里糊涂，教学效果普遍很差。这里既有学生自身的基础问题，也有C语言作为入门语言自身的固有缺陷问题。作为入门语言应该比较规范，而C语言过于灵活，存在语言检查不够规范、严密等问题。

计算机技术发展很快，面向对象的事件驱动程序的开发平台早已广泛应用，而C语言仅仅是面向过程的结构化语言，当学生今后要进一步学习VC++等可视化程序设计技术时，又要重新学习VC++编程技术所使用的语言C++，这显然是十分不妥的。与C++比较，C#更简单、易用、安全，它是纯粹面向对象的，因而更现代。为此，本书编写时采用了仿照C—C++—VC++的主要内容，从零起点的角度用C#语言进行编写。期望能通过形象化、具体化的途径，达到易学好用的目的。让读者较轻松地通过程序设计语言的学习，提升编程思维能力，为进一步从事程序设计打好较扎实的语言和应用基础。

任务1-1 为什么编程要从过程化发展到对象化？对象化编程为什么要使用“类”？

我们不妨先介绍一下第一次软件危机的情况吧。

20世纪60年代末到70年代初，出现了大型软件系统，如操作系统、数据库，这给程序设计带来了新的问题。大型系统的研制需要花费大量的资金和人力，可是研制出来的产品却可靠性差、错误多、不易维护和修改。一个大型操作系统有时需要每年几千人的工作量，而获得的系统又常常会隐藏几百甚至几千个错误。也就是说，在大型程序设计中，投入大量的人力、物力、财力和时间开发出来的软件，其成本、效率、质量等方面都处于失控状态，尤其是软件的维护异常困难。造成这种情况的原因很多，而直接原因是出自面向过程程序设计本身的重大缺陷。这就是人们常说的历史上的第一次“软件危机”。它的问题何在呢？

面向过程程序设计存在的问题有：

(1) 数据默认公有，易被修改；

(2) 数据和数据处理分离,管理不便,结构繁杂;

(3) 代码无法重用。

由于数据与数据处理分离,代码的重用性差,所以程序的维护十分困难。当程序规模较大时,编程人员必然会显得力不从心。

针对面向过程结构化程序设计的种种缺陷,人们提出了面向对象的程序设计方法。

面向对象程序设计是软件系统设计与实现的新方法,这种方法是通过增加软件的可扩充性和可重用性来提高程序设计者的生产能力,控制软件的复杂性,降低软件维护的开销。因此,它的应用使软件开发的难度和费用大幅度降低,已为世界软件产业带来了革命性的突破。

在客观世界中,人们处理问题都是面向对象的,对象是构成系统的基本单位。在实际社会生活中,人们都是在不同的对象中活动。

① 一个具体的杯子是一个对象,它的属性有:口径、型号和材质等,对它的操作(或者说它的行为)是盛水等;

② 一部具体的汽车也是一个对象,它的属性有:品牌、型号和排量等,对它的操作(或者说它的行为)是开动和转弯等;

③ 一个具体的人同样也是一个对象,人的属性有性别、身高和体重等,人的行为(也可称为操作)有走路、吃饭等。

类则是一个抽象的概念,用来描述某一类对象所共有的、本质的属性和类的操作、行为。对象则是类的一个具体实现,又称为实例。

以杯子为例,它是描述这类对象共有的、本质的属性和操作、行为的抽象体,而大杯子和小杯子则是杯子类的某个实例,或者说是杯子类的具体对象。

对象的基本特征分为静态特征和动态特征,我们不妨举两个具体的例子。

【例 1-1】 学生在一个班级中上课、开会、开展社团活动和文体活动等。

这里的对象是班级,它的静态特征是:所属系、专业、学生人数、所在教室等;它的动态特征有:上课、开会、开展社团活动和文体活动等。

【例 1-2】 我们所熟悉的计算机也是一个对象,它的静态特征(或者说属性)有 CPU、内存、硬盘、主板、显卡、声卡、键盘、鼠标、光驱等,它的动态特征(或者说行为)有打字、上网、游戏、编程、处理图像、听音乐、欣赏影视节目等。可以说,计算机的组成部件和计算机所做的各种事情共同描述了一部计算机。

类是具有共同特征的对象的抽象,例如:

① 教师。肩负传道、授业、解惑重任的一类人;

② 学生。接受思想教育、道德教育、专业教育、人文教育的一类人。

教师和学生同属于人类,他们是人类中两个属性和行为各不相同的对象(也可称为实例)。

对类的成员的访问级别可分为:公有、私有和保护三级。而且若没有申明,类默认为私有,我们知道结构体默认的是公有。

类具有抽象性、隐蔽性和封装性的特征。类的隐蔽性就体现在外界不能直接访问私有成员。封装性使对象的数据得到了保护,所以说封装性是"面向对象"程序设计的重要特征。类是一个封装体,在其中封装了该对象的属性和操作。通过限制对属性和操作的访问权限,

可以将属性“隐藏”在类的内部，公有方法作为对外的接口，在对象之外只能通过这一接口借助于对象对类的私有性成员进行具体的操作。

C#就是通过建立数据类型——类来支持封装和数据隐藏。封装性增加了对象的独立性，从而保证了数据的可靠性。一个定义完好的类可以作为独立模块使用。

对象的属性和行为总是紧密联系在一起的，属性用数据（即变量）来描述，行为则是对数据的处理，要通过方法（即函数）来实现。数据和对数据的处理在面向过程的程序设计中是分离的，而在面向对象的程序设计中两者是合一的，都封装在类体中。

封装性就是指将数据（变量）和数据处理（方法）都封装在类体内。我们可以理解为是把变量和相关的方法集中在一个有孔的容器中，只有在孔的边缘处的数据与方法才能与外界相通（这便是指所有的公有的成员），而其余的（指私有和保护的成员）均不受外界的影响，这个容器就是类。

在程序设计与实现中，程序设计方法正在从面向过程走向面向对象，使得编程语言与自然语言之间以及程序设计方法与实际解决问题方式之间的距离越来越近。这就意味着软件开发人员可以用更接近自然的思维方式，用更少的精力去完成同样的工作。

概括起来说，面向对象程序设计有如下优点：

- 与人类习惯的思维方式一致。
- 可重用性好。
- 可维护性好。

正因为面向对象程序设计有众多的优点，所以今天程序设计方法逐步由面向过程程序设计发展为面向对象程序设计。

类是一种数据结构，它可以封装数据成员、方法成员和其他的类。类又是创建对象的模板，C#所有的语句都必须包含在类内。因此，类是C#语言的核心和基本构成模块。C#支持自定义的类，使用C#编程，实质上是编写自己的类来描述实际需要解决的问题。

使用任何新的类之前都必须声明它，一个类一旦被声明，就可以当作一种新的类型来使用。在C#中通过用class关键字来声明类，声明形式如下：

```
类修饰词 class 类名
{
    类体
}
```

1.2 几个重要概念

1.2.1 命名空间

C#程序是利用命名空间组织起来的，即对具有相关功能的类在逻辑上进行分组，类似于存放相关物品的容器。例如一所大学中的人员模型可能包括以下几种类型：个人、学员、职员、教师、学生等。命名空间既用作程序的“内部”组织系统，也用作向外部其他程序公开自己拥有的程序元素的方法。