C# 面向对象程序设计

为了满足决策需要，将数据转换为信息是一个非常重要的过程，而程序存在的最主要价值，可以说就在于实现上述过程的计算机化，让整个操作得以高效进行；从这个角度看，如果没有数据处理的需求就几乎不会有程序设计的需求。因此，对于真实世界中的数据，如何在程序里加以表达与运用就成为了一个格外重要的课题！

全书以此为目的，以面向对象为经，以实现数据的程序化为纬，详细介绍了C#如何为信息的产生而竭尽全力！

黄聪明　编著

科学出版社
www.sciencep.com

图字：01-2003-7466 号

内 容 简 介

Visual C# .NET 是一套综合工具集，可以用来为 Microsoft Windows 和 Web 开发 XML Web 服务以及基于 Microsoft .NET 的应用程序。这个强劲的开发包使用面向组件的 C# 开发语言，为具备 C++或 Java 经验的初级和中级开发人员创建下一代软件提供了现代化的语言和环境。

本书的作者在多年教学和开发经验的基础之上，没有刻板地对 C# 的语法和编程技术进行逐条的介绍，而是以专业的眼光对 C# 语言的各项内容进行了总结和分类，并且提供了大量的代码示例供读者学习和参考，旨在使读者能够迅速掌握面向对象编程的实质，从而快速和高效地开发出具有专业水准的 Windows 应用程序和组件。

本书是 C# 语言的入门书籍，具有很强的实用性和指导性。语言通俗易懂，讲解细致深入，适合初学和自学计算机编程的各类人员学习，有经验的开发人员也可以使用本书作为参考。

图书在版编目（CIP）数据

C#面向对象程序设计：台版/黄聪明编著. —北京：科学出版社，2004
ISBN 7-03-012484-7

Ⅰ.C… Ⅱ.黄… Ⅲ.C 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2003）第 101538 号

策划编辑：李佩乾 / 责任编辑：朱凤成
责任印制：吕春珉 / 封面设计：一克米工作室

科学出版社 出版
北京东黄城根北街16号
邮政编码:100717
http://www.sciencep.com
新蕾印刷厂 印刷
科学出版社发行 各地新华书店经销
*
2004 年 1 月第 一 版 开本：787×1092 1/16
2004 年 1 月第一次印刷 印张：57 1/2
印数：1—4 000 字数：1360 000

定价：98.00 元（含光盘）

（如有印装质量问题，我社负责调换〈环伟〉）

序

编程语言（programming language）是任何信息系统形成的必要部分，而且也是计算机科学教学上所不可或缺者，甚至是一种用来精确描述个人思维的有效的工具之一，就像是人类思维的一种速记！

编程语言何其多也，而本书则选择 Microsoft 在 .NET Framework 中新创的 C#作为讨论编程语言基本观念的媒介，并且配合所谓的面向对象程序设计的思维模式来运用这样的一个思维速记工具。

学习使用编程语言来设计程序，对许多初学程序设计的人来说，可能都不是个愉快的经验，至少一开始的时候不是，这除了个人的因素外，再加上面对许多不熟悉的专有名词、过多的语法、过少的范例、及欠缺连贯性与逻辑架构的章节安排，都会对学习造成雪上加霜的效应；因此，我将 1985 年以来陆续接触 COBOL、dBase/Clipper、Pascal/Delphi、C/C++、Prolog、汇编语言、Visual FoxPro、JavaScript、ActionScript、Perl、Visual Basic/Visual Basic .NET 与 Java 等编程语言关于教学与自学所累积的点滴经验，通过不断地试着从一个初学者、教学者与研究者等不同的角度来思考编程语言及程序设计，希望能够写出一本真正能满足程序设计初学者需求的书，因此，在书中我将通过详细的分步讲解，将编程语言及程序设计的内容予以系统化、逻辑化，并尽量以图解的方式来做说明，希望本书终能成为初学程序设计者进入程序设计领域的垫脚石，我所著的《Java 2 程序设计彻底研究》一书基本上算是实现了上述愿望，而本书则是更进一步对该书略显不足的观念及范例进行了加强，如果说该书实现了 90%的愿望，那我希望本书能够趋近 100%。

本书的内容及组织

本书内容分成四个主要的部分：

Part 1 程序设计的基本观念与环境建立（第 1 章至第 2 章）

第 1 章，针对程序设计做一个简要的说明，让各位在运用 C#编程语言来开发程序前，能够知道究竟程序设计是什么，程序设计的基本结构及程序设计的思维方式，特别是针对面向对象程序设计思想的说明。

第 2 章则说明建立 C# 开发环境的一些必要的软硬件需求及其安装。

Part 2 C# 基本编程语言规范（第 3 章至第 7 章）

编程语言的规范及如何利用 C#来设计 Windows 应用程序都会在这个部分加以说明。

经过这些章节的洗礼，各位将会对组成语言规范的变量、数据运算及思想流程的运作有充分的了解。

Part 3 C# 面向对象程序设计思想（第 8 章至第 15 章）

到底 C#编程语言如何实现面向对象思想？这就是这个部分所要说明的：包

括类的设计、继承、多态及对象响应外界环境变化的面向事件能力。

除了说明面向对象思维的具体实现方法之外，本部分着重于如何借助 C#编程语言来使用 .NET Framework 的数据表达和运算类库，包括数组、字符串、时间与数值数据。

Part 4 Web Application 程序设计（第 16 章）

本书的最后一个部分则与网页应用程序有关。这个部分可视为 C#可视化程序设计的延伸。

这一章会简单地带领各位看一下 ASP（Active Server Pages）的下一代开发方式：ASP.NET，以及将程序代码复用的观念通过网络予以强化的网络服务（Web Services）。

特别注意

◆ 类库及其定义的引用

由于利用 C#开发的程序势必会使用到 Microsoft .NET Framework，因此一定会引用大量的官方资料，而且这些资料又非引用不可，例如 String 对象及其方法的定义及功能的说明，所有的引用纯为讲述 C#及 .NET Framework 所必须，这应该和所谓的抄袭不同！用别人的技术却不引用别人的规范是不可能的！特别是一本以这些规范作为教学目的的书！

◆ 个人见解

书中会穿插类似下面的文字，用来对文中的说明进行一个个人观点的自问自答，各位可以参考，这样的文字段落算是头脑风暴吧！

各位觉得 C#规定 named constant 必须要在进行定义的同时赋给数据常量以内容；但是变量则可以在事后定义它的值，是否合理？

个人的见解是：变量本来就是表示数据内容会变的数，既是如此，声明时是否定义就不是那么重要，但是 named constant 应该是事前已知其值，而且可望在未来不应变动，因此声明时就赋予应有的值，以免声明后不小心定义了不希望的内容！

◆ 比喻及类推的使用

B.R. Hergenhahn 与 Matthew H. Olson 二位作者在 An Introduction to Theories of Learning 一书第 2 章 2.3 节“模型的使用”中提到：

“在 Random House 英语字典中，把类推（analogy）一词定义为：两件相似东西之间的部分类似性，根据这样的类似性即可据以进行比较之用。在科学上，发现两件类似的东西经常是有用的，特别是当其中的一件东西为大家所熟悉，而另一件则不为大家所熟知时尤甚。如此一来，我们便可使用大家熟悉的对象作为解释另外较不为人知的东西的模型。例如，注意到机械式抽水机（这是为大家所较为熟悉的东西）的功能与人类的心脏（一个较不为人知的东西）功能之间的相似性，可为心脏研究提供有用的指导。”

因此，比喻的使用通常只是拿其中的一个方面来类推，但不表示说明中的两个东西或是观念应该完全相同，例如，我在第 4 章用容器来比喻变量时，取的角度只是两者都有装载的功能，却不表示变量的行为应该完全与容器相同，切记！

◆ 书中范例

书中范例并非以单个程序为单一文档存在，而是将各章的范例及练习以一个 Word 文档予以储存，这些范例的目的在于让各位参考而已，希望各位都能自己亲自去编写每个范例程序，而不是直接就拿现成的范例予以编译及执行而已！切记，程序是写出来的，不是看出来的！

黄聪明

2003 年 5 月

目　录

第 1 章　程序设计概论 1
1.1　什么是程序 1
1.2　程序设计思维 7
1.3　实现程序化的工具——程序语言概论 15
1.4　程序语言的选择——思维模式的搭配 19
第 2 章　开发环境的建立 20
2.1　系统需求 20
2.2　开始安装 21
第 3 章　程序设计初步体验 35
3.1　程序的实体组成单元 35
3.1.1　构成程序的符号 35
3.1.2　一系列的字符 37
3.1.3　空格符 37
3.1.4　注释 38
3.1.5　标识符 38
3.1.6　保留字 39
3.1.7　数据常量 40
3.1.8　分隔符 41
3.1.9　运算符 41
3.2　C#程序的种类 41
3.3　C#程序的开发 42
3.3.1　编写阶段 42
3.3.2　编译阶段 45
3.3.3　执行阶段 45
3.4　范例 46
3.4.1　范例一 46
3.4.2　范例二 57
3.4.3　范例三 63
3.4.4　范例四 66
3.4.5　范例五 71
3.4.6　范例六 77
3.4.7　范例七 82
3.4.8　范例八 85
3.4.9　范例九 88
3.4.10　范例十 93

3.5 符号组合后的语意 94
3.6 错误总结 97
3.7 用户自定义名称的命名习惯 102
第 4 章 可视化程序设计 I Windows Application 103
4.1 Visual Studio .NET 主要环境简介 103
4.2 控制台应用程序 106
4.3 Windows Form 可视化程序设计 120
4.3.1 范例一 120
4.3.2 范例二 127
4.4 Windows 应用程序的安装与部署 137
4.5 实例研究——过程模块的思考 146
4.5.1 没有笨的用户，只有差劲的程序设计人员 146
4.5.2 找出基本逻辑后加以调整 162
第 5 章 数据的表达 167
5.1 数据类型概论 167
5.1.1 类型的重要性 167
5.1.2 类型 168
5.2 程序中如何表示一份数据 173
5.2.1 变量的意义 174
5.2.2 变量的声明 179
5.2.3 “=”的意义（赋值运算符） 183
5.2.4 内存与内存的内容 184
5.2.5 数据类型总论 186
5.2.6 选用数据类型应特别注意的事项 189
5.3 基本数据类型 190
5.3.1 整数数据类型 190
5.3.2 浮点数值数据类型 195
5.3.3 真/假值数据类型 199
5.3.4 字符数据类型 200
5.3.5 不同基本数据类型的数据如何转换 204
5.4 具名的数据常量 206
5.5 自定义数据类型 209
5.5.1 枚举数据类型 209
5.5.2 structue 数据类型 215
5.6 负责数据类型转换的类 218
第 6 章 数据的运算 220
6.1 概论 220
6.1.1 数据的基本能力 220
6.1.2 运算能力 222

6.2 运算符 224
6.2.1 改变运算符优先级的运算符 224
6.2.2 明确类型转换与 cast 运算符 224
6.3 算术运算符 230
6.3.1 四则运算 230
6.3.2 除法的余数运算 232
6.3.3 增减 1 的递增 / 减运算 239
6.3.4 正 / 负号的运算 240
6.3.5 Op 运算赋值符号的运算 241
6.3.6 算术运算符的运算顺序 242
6.4 比较运算符 243
6.4.1 相等运算 243
6.4.2 关系运算 245
6.4.3 比较运算符的运算顺序 246
6.5 逻辑运算符 246
6.5.1 逻辑运算符的真值表 247
6.5.2 逻辑乘 248
6.5.3 逻辑和 249
6.5.4 逻辑非 252
6.5.5 Short-circuit evaluation，简化方式 255
6.5.6 Op 运算赋值符号的运算 258
6.5.7 逻辑运算符的运算顺序 259
6.6 位运算符 260
6.6.1 &的用途 263
6.6.2 | 的用途 265
6.7 条件运算符 265
6.8 类型兼容判断符号 266
6.9 副作用 266
6.10 与算术运算相关的类库 267
6.10.1 System.Math 267
6.10.2 System.Random 284
6.11 数据的应用范围 288
6.12 重载标识符 292
第 7 章 流程控制 295
7.1 概述 295
7.1.1 顺序流程控制 295
7.1.2 选择性执行的流程控制 296
7.1.3 选择性重复执行的流程控制 296
7.1.4 执行流程的组合方式 298

7.2 流程控制 300
7.2.1 选择性执行的流程控制 300
7.2.2 选择性重复执行的流程控制 320
7.2.3 执行转移的流程控制 345
7.3 try/catch/finally 语句 352
7.4 预处理（preprocessor）的流程控制 368
第 8 章 类与面向对象程序设计 370
8.1 基本观念 370
8.1.1 对象 370
8.1.2 类 377
8.2 程序实现 384
8.2.1 类的声明和定义 385
8.2.2 建立一个对象 387
8.2.3 对象的生与死 395
8.2.4 抽象类 425
8.2.5 sealed 类 427
8.3 接口 427
8.3.1 接口，一个象征性的符号 427
8.3.2 定义规范接口 428
8.3.3 规范接口 430
8.3.4 规范接口的赋值运算 433
8.3.5 规范接口的重载标识符 435
8.3.6 规范接口的扩展 435
8.3.7 接口的声明格式 440
8.3.8 .NET Framework 中的基本接口 440
第 9 章 数组数据的表达与运算 452
9.1 数组是什么 453
9.2 多维数组 465
9.3 注意事项 473
9.4 System.Array 478
9.4.1 统计信息 480
9.4.2 管理数组元素 483
9.4.3 数组操作 490
9.4.4 数组元素的操作 504
9.5 应用 508
9.5.1 冒泡排序 508
9.5.2 二分法搜索 511
9.5.3 魔术方阵 515
9.5.4 矩阵操作 529

第 10 章　字符串数据的表达与运算.....538
10.1　字符串.....538
10.2　String 的应用.....565
10.3　StringBuilder.....568
第 11 章　日期与数值数据的表达与运算.....572
11.1　日期数据类型.....572
11.1.1　TimeSpan 结构.....572
11.1.2　DateTime 结构.....582
11.1.3　Calendar.....599
11.2　再谈数值类型.....600
11.2.1　Boolean 结构.....602
11.2.2　Char 结构.....603
11.2.3　整数结构.....605
11.2.4　含小数值的结构.....614
11.2.5　Decimal 结构.....615
第 12 章　设计类的成员(I).....627
12.1　Field：字段的数据成员.....627
12.1.1　访问设定：考虑封装层次.....628
12.1.2　实例数据成员的设计.....630
12.1.3　静态数据成员的设计.....633
12.1.4　常量数据成员的设计.....637
12.1.5　只读数据成员的设计.....641
12.1.6　枚举值的设计.....643
12.1.7　属性的默认值.....644
12.1.8　Shadowing（遮蔽效应）.....645
12.2　方法：成员函数.....647
12.2.1　方法的名称.....652
12.2.2　返回值类型.....652
12.2.3　方法返回数据的管道.....654
12.2.4　返回值与类型转换.....657
12.2.5　方法重载.....658
12.2.6　参数与参数的传递.....662
12.2.7　静态方法成员的设计.....680
12.2.8　sealed 方法.....684
12.2.9　再论数据的范围与生命期.....685
12.2.10　再论 this.....694
12.2.11　递归算法.....695
12.2.12　abstract 方法.....711

第 13 章 关系 713
13.1 类与类的关系 713
13.1.1 什么是关系 713
13.1.2 什么是相依性关系 714
13.1.3 什么是一般化的关系 715
13.1.4 什么是整体与部分的关系 719
13.1.5 一个称作“Object”的类 720
13.2 一般化关系的程序实现 723
13.2.1 基本原则 723
13.2.2 构造函数顺序的相关性 725
13.2.3 功能 730
13.2.4 有多少遗产 734
13.2.5 赋值运算 736
13.2.6 我是谁 743
13.2.7 龙生龙，凤生凤 746
13.2.8 同名的处理方式：shadow 和 override 748
13.2.9 保留字 base 与 this 在继承关系下所扮演的角色 758
13.2.10 保留字 abstract 在继承关系下所扮演的角色 775
13.2.11 保留字 sealed 在继承的关系下所扮演的角色 779
13.2.12 保留字 protected 在继承关系下所扮演的角色 783
13.2.13 再论继承 785
13.2.14 多态 786
13.3 集合关系 799
第 14 章 设计类的成员（II） 801
14.1 运算符重载 801
14.1.1 ++/－－ 802
14.1.2 true/false! 803
14.1.3 +、－ 805
14.1.4 关系运算符号 808
14.1.5 casting 812
14.2 属性 814
14.3 Indexer － 索引运算符 824
第 15 章 设计类的成员（II）：委托与事件 827
15.1 委托 831
15.1.1 程序的基本架构 831
15.1.2 多播（multicasting） 845
15.2 事件 849
15.2.1 程序的基本架构 849
15.2.2 .NET 事件的语法规范 851

第 16 章　可视化程序设计 ASP .NET 与 Web Service 856
16.1　ASP .NET 程序的基本架构及环境设置 856
16.1.1　欢迎来到 ASP .NET 856
16.1.2　设置 867
16.2　Visual Studio .NET 与 ASP .NET 871
16.3　ASP .NET 服务器控件 890
16.4　Web Service 897
16.4.1　定义 Web Service 的内容 898
16.4.2　测试 Web Service 902
16.4.3　Visual Studio .NET 与 Web Service 903

第 1 章　程序设计概论

1.1　什么是程序

COBUILD 英文字典对程序（program）一词的说明如下：

n. A program is a set of instructions that a computer follows in order to perform a particular task.

[译]所谓的程序是指一组指令，计算机遵循这些指令能够完成一些特定工作。

v. When you program a computer, you give it a set of instructions to make it able to perform a particular task.

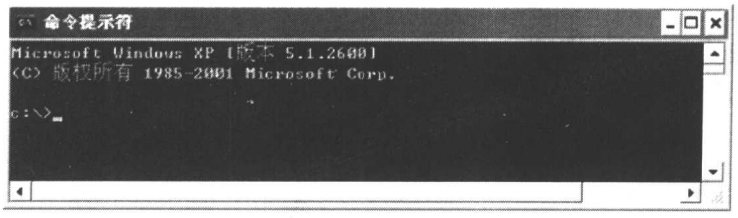
[译]当你为计算机设计程序时，是给计算机提供一组指令，使其能够完成特定工作。

由此可知，当各位开始练习写程序时，有两点需谨记在心：

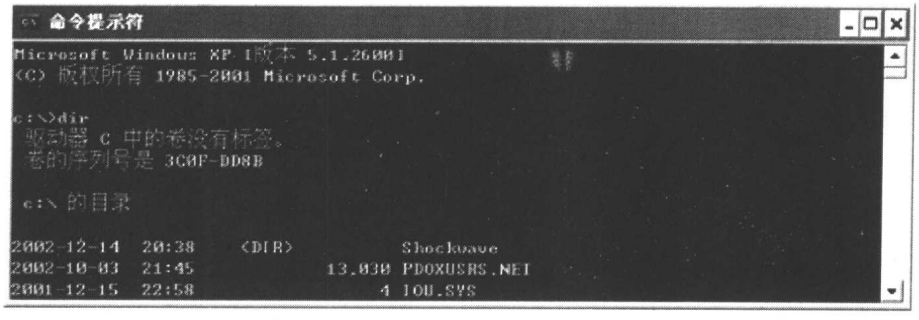
任何一个程序的存在都是为了解决特定的问题，因此开始撰写程序前就必须先针对问题进行分析、规划和设计，然后利用工具来设计程序。

设计出来的程序是用来指挥计算机运作的各项指令的集合，是程序用户与计算机之间开展沟通的语言，因此这些指令的使用必须让计算机能够看得懂，如果从程序语言的角度来看，就是这些指令都必须是符合语法规范的指令。

下面是执行 Windows 的命令提示符这个程序时的画面：

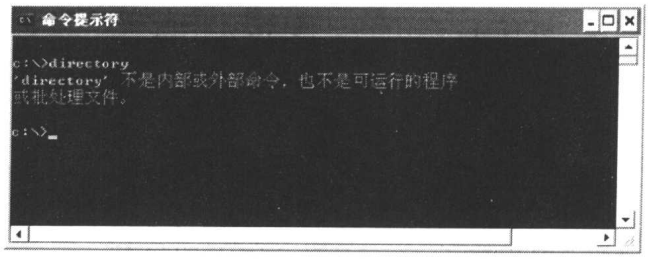


在 Windows 尚未出现前，大部分在 PC 上执行的程序，都是通过这样的环境来完成的。用户想要做什么事，都需要自己直接下指令，也就是利用指令来与这个环境进行沟通，例如，想要知道当前目录的内容，正确的命令是 Dir：



这条命令就是这个程序的程序设计人员设定的用来与计算机进行沟通的格式，如果

我们不遵守，例如将 Dir 改成 Directory：



这时候，因为程序不懂这道指令，计算机就无法顺利执行！

我举这个例子，就是要告诉各位：

① 未来我们利用程序语言来与计算机沟通时，同样要遵守那个程序语言的格式，否则是无法沟通的，一旦无法沟通，当然也就没有办法命令计算机来完成特定的任务了！

② 指令的格式早就制定好了，只能遵守，毫无讨价还价的空间！

前面提到，程序的目的是为了完成特定的任务，换句话说是要建立程序来帮助我们解决现实世界中的问题，通常就是将原本是人工操作的工作改由计算机来完成。但是人工操作下的工作不下数千数万种，那么要将这样的工作写成程序让计算机得以执行岂不是“不可能完成的任务”吗？到底有没有什么特定的方向可供遵循以方便程序的开发？也就是说写程序时有没有什么基本的模式可供参考呢？

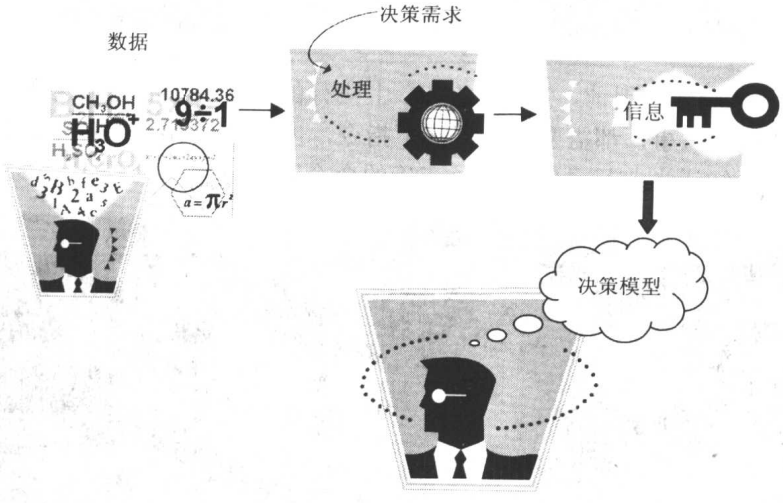
人工操作的工作真的是不下数千数万种，但基本上人工操作的处理流程，通常只是三个基本模块的组合罢了，这三个基本的模块就是：

输入模块——取得数据。

处理模块——将数据处理成符合特定决策需求的信息。

输出模块——输出信息。

这三个单元的关系如下图所示：



因此设计程序时，必然也需要包含相关的指令让计算机遵循：

◆ 数据的输入

此阶段工作的重点在于如何利用计算机所提供的工具将人工操作处理流程中的数据在计算机中实现出来，为了方便数据的输入，通常需搭配相关的用户界面，以方便数据的在线输入或从储存媒体中读取。设计用户界面，消极的目的是为了获取数据，积极的目的是希望通过此界面能帮助用户有效输入数据以方便后续处理。

◆ 数据的处理

利用对已经取得并且通过计算机实现的数据加以组织、整理以产生所需的信息。

◆ 数据的输出

利用计算机提供的工具将处理而得的信息加以输出。

换言之，不管程序如何复杂，都可分解成输入、处理及输出三个部分，因此学习任何程序设计时都必须对与这三个模块有关的指令有所了解。

若觉得不够具体、不够清楚，下面我用两个例子来说明这三个基本模块的使用情况。

Q: 假设你要写一个程序来判断某个通过用户界面取得的编号到底是不是公司的已注册会员时，怎么办?

A: 根据前面三个模块分述如下:

输出单元: 显示出是“是会员”或“不是会员”的信息。

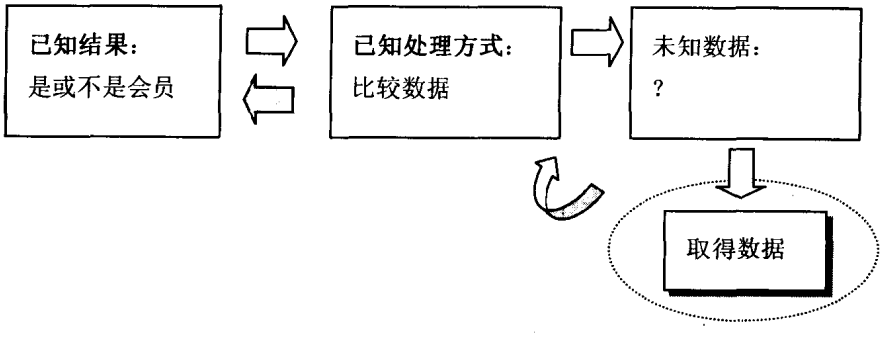
处理单元: 将该编号与现有会员数据比较。但去哪里拿到那个待比较的编号数据呢? 当然是从输入单元来的!

输入单元: 取得编号数据。但操作计算机的人员怎么知道要把编号数据的内容填入? 计算机又怎么会知道应该开始判断是不是会员数据? 喔! 那就做个画面让计算机操作人员输入数据吧! 这个画面就是用户界面。

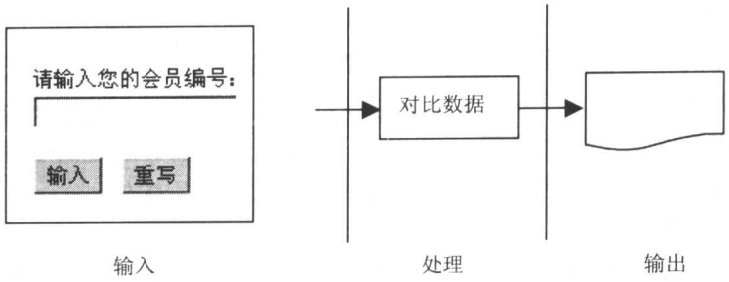
各位是不是注意到了，设计的流程好像与数据的流程相反。

我知道我要的结果，我也知道怎么处理来达到这样的结果，但是我没有数据。既然没有数据，那么我就向别人要，然后根据我知道的处理方式加以处理，最后将我要的答案求解出来。

思考逻辑如下图所示:



执行流程则为:



Q: 如何写一个程序来将摄氏温度转换为华氏温度？

A: 依据前面三个模块分述如下：

输出单元：显示出是“华氏温度”。

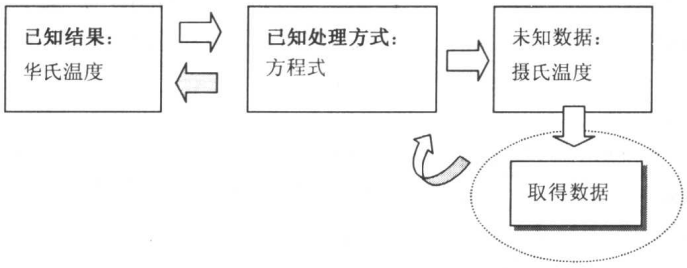
处理单元：从数学课本里，我们知道，华氏温度等于摄氏温度乘以 9 除以 5，最后再加上 32，换成数学里常用的方程式，其方程式如下：

$$\text{华氏温度}=\frac{9}{5}\times\text{摄氏温度}+32$$

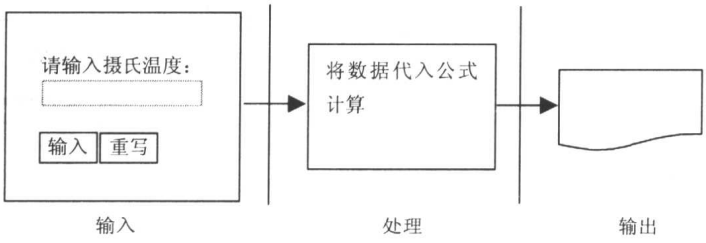
其中华氏温度显然就是输出的结果，而方程式右边的就是我所谓的处理模块，不过这里有个未知数：摄氏温度，因此处理模块也无法算出结果，为了达成目的，势必要拿到这个表示摄氏温度的值！当然这个数据处理模块无法自行产生，最后还是要向执行这个程序的用户索取。

输入单元：取得摄氏温度。但计算机的操作人员怎么知道要把摄氏温度的内容填入？计算机又怎么会知道开始将这个摄氏温度转换为华氏温度然后显示出来呢？喔！那就做个界面让计算机操作人员输入数据吧！

思考逻辑如下图所示：



执行流程则为：



综上所述，可知整个运行过程中，想要达到某一种结果时，需要找到一个适当的处理模块，通过这个模块才能知道需要哪些数据才能满足程序的输出，也才能据此设计用

户界面来让使用程序的人把处理模块需要的数据提供给程序，好让程序使用这个数据去处理以求得结果。

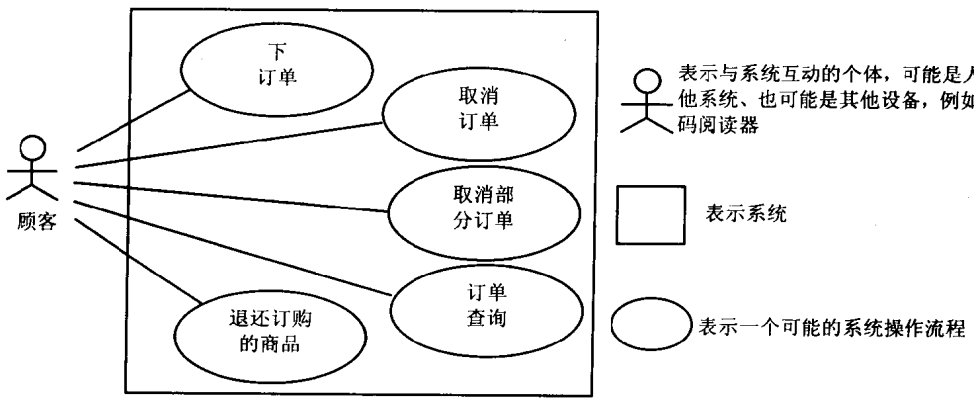
显然在这三个模块里，处理模块最为重要，程序中的处理模块位于一个承前启后的位置：承接用户先前输入的数据，开启输出的结果。我们可以这么说，如果无法知道应该如何处理，就无法知道需要哪些数据；无法知道需要哪些数据，就无法产生想要的结果。

既然处理模块是这么的重要，那么应该怎么取得特定问题领域的处理模块呢？个人认为，程序设计者应该具有下面两项能力：

① 具有某种领域的知识。比如说，设计会计信息系统时，应具备有会计及账务处理的基本认识，如此才能对处理何种交易数据，输出何种报表，相关输入应该是什么，以及三者间的关系如何等等问题了然于胸。当然，并非所有的人工操作流程皆为程序设计人员所了解，因此，这里所指的是当程序设计人员真正在写程序时，必须对他正准备计算机化的人工操作流程有所了解，通常必须在需求分析阶段获得所需的知识。

② 具有不错的知识组织能力，也可以说是“思维模式”，因为这牵涉到如果程序设计人员对该人工操作流程并非本行时，如何对程序设计人员所不熟悉的人工操作流程知识进行有效和正确的整理，并将其转换成计算机能够遵循的指令。当程序设计人员并非该人工操作的专家时，此时他对人工操作流程的了解势必取材于访谈及事后寻找相关的数据，若没有不错的知识组织能力，就算有不错的编程能力也无济于事。通常，程序设计人员会在系统分析阶段采用一些所谓的 CASE 工具（Computer Aided Software Engineering，计算机辅助软件工程）来帮助自己将所获得的知识组织起来。从目前来看，用于面向对象思维的分析工具大都采用 UML（Unified Modeling Language，统一建模语言）。UML 中的“使用分类图”可对人工操作流程进行静态描述，而“操作图”则是动态的说明。

例如，在一个目录邮购的处理系统里，程序设计人员根据所做的需求分析与知识组织，利用“使用分类图”来描述该系统：



以下我用 4 个说明来描述这样的观点，如果这个例子你看不懂的话，不用紧张，这只表示截至目前为止，你尚无能力将此例子写成程序而已，但不会影响各位对本书的学习，这个例子主要是让你了解，能否找出处理模块会在你写程序时决定人工操作计算机