高等院校软件工程专业规划教材

C++程序设计与实践

第2版

免费提供

电子教案

白忠建 编著

机械工业出版社
CHINA MACHINE PRESS

高等院校软件工程专业规划教材

C++程序设计与实践

第2版

白忠建　编著

机械工业出版社

本书采用逐层递进的方式，通过对案例的分析，将 C ++ 的语法及编程方法逐步展开。书中以 C ++ 1y 标准为基准，详细介绍了对象和面向对象技术的概念，并围绕案例的求解，深入浅出地介绍了面向对象技术的 4 个核心思想（数据封装、继承、多态和泛型编程）在 C ++ 中的概念、实现机制和语法、编程方法等，其中包括类和对象、运算符重载、继承和派生、虚函数和多态性、模板和泛型编程、流库、多继承、名字空间和异常处理，使读者能够循序渐进地掌握 C ++ 的语法以及面向对象程序设计的方法。

本着“能力为重”的理念，在每一章的重要知识点之后均穿插了适量的实践性题目，建议读者动手实践，以加深理解。

本书既可作为高等院校计算机专业相关课程的教材，也可作为 C ++ 程序员的参考书。

本书配套授课电子课件，需要的教师可登录 www. cmpedu. com 免费注册，审核通过后下载，或联系编辑索取（QQ：2850823885，电话：010 - 88379739）。

图书在版编目（CIP）数据

C ++ 程序设计与实践/白忠建编著 . —2 版 . —北京：机械工业出版社，2016. 8
高等院校软件工程专业规划教材
ISBN 978 - 7 - 111 - 54491 - 3

Ⅰ. ①C… Ⅱ. ①白… 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（2016）第 181294 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策划编辑：郝建伟 责任编辑：郝建伟
责任校对：张艳霞
河北鑫宏源印刷包装有限责任公司印刷

2016 年 10 月第 2 版 · 第 1 次印刷
184mm × 260mm · 20 印张 · 487 千字
0001 - 3000 册
标准书号：ISBN 978 - 7 - 111 - 54491 - 3
定价：49. 00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
服务咨询热线：（010）88379833
读者购书热线：（010）88379649

网络服务
机 工 官 网：www. cmpbook. com
机 工 官 博：weibo. com/cmp1952
教育服务网：www. cmpedu. com
金 书 网：www. golden - book. com

出版说明

计算机技术的发展极大促进了现代科学技术的发展，明显加快了社会发展的进程。因此，各国都非常重视计算机教育。

近年来，随着我国信息化建设的全面推进和高等教育的蓬勃发展，高等院校的计算机教育模式也在不断改革，计算机学科的课程体系和教学内容更加科学和合理，计算机教材建设逐渐成熟。“十五”以来，机械工业出版社组织出版了大量计算机教材，包括“21 世纪高等院校计算机教材系列”“21 世纪重点大学规划教材”“高等院校计算机科学与技术‘十五’规划教材”和“21 世纪高等院校应用型规划教材”等，均取得了可喜成果，其中多个品种的教材被评为国家级、省部级的精品教材。

为了进一步满足计算机教育的需求，机械工业出版社策划开发了“高等院校规划教材”。这套教材是在总结我社以往计算机教材出版经验的基础上策划的，同时借鉴了其他出版社同类教材的优点，对我社已有的计算机教材资源进行整合，旨在大幅提高教材质量。我们邀请多所高校的计算机专家、教师及教务部门针对此次计算机教材建设进行了充分的研讨，达成了许多共识，并由此形成“高等院校规划教材”的体系架构与编写原则，以保证本套教材与各高等院校的办学层次、学科设置和人才培养模式等相匹配，满足其计算机教学的需要。

本套教材包括计算机科学与技术、软件工程、网络工程、信息管理与信息系统、计算机应用技术以及计算机基础教育等教材系列。其中，计算机科学与技术系列、软件工程系列、网络工程系列和信息管理与信息系统系列是针对高校相应专业方向的课程设置而组织编写的，体系完整，讲解透彻；计算机应用技术系列是针对计算机应用类课程而组织编写的，着重培养学生利用计算机技术解决实际问题的能力；计算机基础教育系列是为大学公共基础课层面的计算机基础教学设计的，采用通俗易懂的方法讲解计算机的基础理论、常用技术及应用。

本套教材的内容源自致力于教学与科研一线的骨干教师与资深专家的实践经验和研究成果，融合了先进的教学理念，涵盖了计算机领域的核心理论和最新应用技术，真正在教材体系、内容和方法上做到了创新。同时，本套教材根据实际需要配有电子教案、实验指导或多媒体光盘等教学资源，实现了教材的“立体化”建设。本套教材将随着计算机技术的进步和计算机应用领域的扩展及时改版，并及时吸纳新兴课程和特色课程的教材。我们将努力把这套教材打造成为国家级或省部级精品教材，为高等院校的计算机教育提供更好的服务。

对于本套教材的组织出版工作，希望计算机教育界的专家和老师能提出宝贵的意见和建议。衷心感谢计算机教育工作者和广大读者的支持与帮助！

机械工业出版社

前　言

C ++是一门非常优秀的面向对象程序设计语言，它不仅继承了C语言的全部优点，同时实现了面向对象技术的所有核心概念，使它成为很多程序员，尤其是从C转过来的程序员开发大型复杂软件的首选语言。

自从贝尔实验室的Bjarne Stroustrup博士研发出C ++后，这门混合型的语言一直在沿着C ++ 98、C ++ 0x和C ++ 1y（C ++ 11和14的合称）的路线进化，最新的C ++ 17也正在被ISO讨论中。这些标准的应用使得C ++变得更加精致，也更容易使用。

C ++是一种混合编程语言。Bjarne Stroustrup这样描述：C ++的强项恰恰在于它支持多种有效的编程风格（多种思维模型）及它们之间的相互组合。最优雅、最有效也最容易维护的解决方案常常涉及不止一种风格（编程模型）。如果一定要用吸引人的字眼，可以说，C ++是一种多思维模型的语言。在软件开发的庞大领域中，需求千变万化，需要至少一种支持多种编程风格的通用语言，而且很可能需要一种以上。

C ++被业界诟病的问题是C ++ STL的效率问题。的确，在C ++ 11之前，STL因为存在大量的内存复制操作而使系统效率低下。而当C ++ 11推出之后，尤其是标准中的右值引用和move语义的应用，使原来的问题不复存在。

目前，C ++ 1y标准已被业界广泛接受，很多程序员都采用此标准编写应用程序。基于此，本书中引入了最常用的C ++ 1y标准，并对它们的语法和应用情况做了详细介绍。为了能使读者分辨标准的差异，书中凡是涉及C ++ 11标准的内容都被做上了标记。

本书的内容分为以下3部分。

第1部分：1 ～4章，主要讲解C ++的基础语法。

第2部分：5 ～10章，主要讲解面向对象技术在C ++中的实现。

第3部分：11 ～13章，主要讲解面向对象技术中的高级话题。

作为讲解面向对象技术的教材，本书重点强调了面向对象技术的4个核心概念：数据封装、继承、多态和泛型编程，而对基础语法部分的介绍并不多。这就要求阅读本书的读者应该已系统学习过C语言，或者具有相当的C语言编程经验。如果这对读者阅读本书造成了困扰，敬请谅解。

本书在第1版的基础上有了较大的改动。除了对章节结构做了一些微调外，笔者还做了大量的文字修订工作，力图使行文更加流畅，更容易阅读和理解，并且重新设计了大部分案例，使读者在阅读和实践时更容易理解和模仿。

本书从第5章起，在每一章讲解C ++的各项知识点时，基本上都采用了一横一纵两个案例来展开。每一章章首的案例贯穿全章，通过对该案例的解析串联起本章的所有知识点；章尾的案例贯穿全书，用以串接所有的知识点。当然，有些案例会在不同的章节中交叉出现。如果这给读者的阅读带来了困扰，敬请原谅。

由于C ++是一门实践性很强的语言，因此笔者强烈建议读者在学习时多上机实践，这样才能将面向对象技术（而不仅仅是C ++的语法特性）掌握得更加牢固。

虽然笔者在高校从事了多年的C ++教学及C ++应用研发，但对这门与时俱进的语言仍然有不能把握的地方。如果读者在书中发现谬误，敬请指正，不胜感激。

编　者

目　录

第1章 引 论

本章要点

- 对象的概念。对象是一个主动的实体，拥有若干属性和行为。
- 面向过程技术与面向对象技术的异同。面向过程技术以过程为中心；而面向对象技术以对象为中心。
- 面向对象的核心概念。数据封装、继承和多态是每一种面向对象的程序设计语言必须实现的核心概念。现代的观点认为泛型编程也是核心之一。
- C++程序的概貌。以类为中心的编码是C++程序的基本特征。
- 贯穿全书的案例。从第4章起，每章的最后一节都讨论了这一案例的局部解决方案。

C++语言是一门面向对象的程序设计语言，它所支持的面向对象的概念容易将问题空间直接映射到程序空间，为程序员提供了一种与传统的结构化程序设计十分不同的思维方式。

虽然C++一直被认为是C语言的后继者，本书也假定读者已掌握了C语言的基础知识，但C++仍然是一门崭新的语言，所以，学习C++与学习C语言相比，要有不同的思维和学习方式。学习C++面临的两个主要问题如下。

- 如何建立面向对象的思维方式？
- 如何用C++语言实现面向对象程序设计？

为了初步回答这两个问题，本章试图从对象的概念、面向对象的核心概念和面向对象程序设计等几个方面，给读者以面向对象概念的总体印象。同时抛出一个贯穿全书的案例，并在其后各章给出该案例的局部解决方案，用以串起C++的关键知识点，并使读者有一个做小型工程的实践机会。

1.1 对象的概念

学习一种计算机程序设计语言，其目的是为了能够使用它来解决实际问题。解决问题的步骤一般如下。

1）对实际问题进行抽象，根据此抽象建立起一个可行可解的现实模型。

2）将现实模型转换为计算机模型。

3）根据计算机模型，用某种程序设计语言编码实现。

4）测试和修改。

5）运维。

从以上步骤中可以看到，问题总是来自于现实，它的解决方案也用之于现实。因此，在引入对象的概念之前，先来考察现实世界，以期获得一些概念性的认知。

1.1.1 现实世界中的对象

在现实的世界中，人们时时刻刻都在面对一些客观实体（客体），例如，一位名叫 Ken 的同事、一栋大楼、一台计算机和一盆植物等。这些客体都拥有不同的特性及独特的行为，构成了人们所认识的外部世界。而作为这些客体中的一员，人们会与其他客体进行交流，或者请求别的客体提供帮助。这里，将这些不依赖于人类意识而存在的客体称为“**对象(Object)**”。

单个的对象是独立存在的，然而却又不是孤立的。很难想象一个由孤立对象构成的世界，那将是一片死寂。实际上，对象之间存在着一张复杂的关系网，而网中的对象随时随地都在进行信息交流，它们之间互相构成了服务与被服务的关系。可以这么说，现实中的对象加上它们之间的关系就构成了现实世界。

正如上面提到的那样，现实对象不是孤立的，而是以群体的方式出现的，正是所谓的“人以群居，物以类聚”。尽管群体当中的对象具有鲜明的个性，但同一个群体中的所有对象都具有相似的特性和行为模式。分类学上会根据对象特性和行为的相似性而将这些对象划分在一个分类当中，然后用一个抽象的概念描述这个分类。例如，世界上所有的张三、李四、Linda 和 Peter 等就构成了“人”这个群体。“人”这个抽象概念，是在总结了大量同类对象的特性和行为模式，并提炼出其中的共同特性的基础上得到的。可以看出，抽象描述了共性，属于这个抽象类别的具象（个体）无条件地拥有这些共性，而具象同时还拥有各自不同的个性。

抽象和具象之间有直观的辩证关系。例如，当听到“人”这个词时，脑海中会立即浮现出由头、躯干和四肢组成的一种直立行走的生物，并且能说会道，有自己的思想，这些正是“人”这种物种的外部特征。此时，“人”这个概念是抽象的，无法详细描述“人”的具体特征，所以，“人”这个概念还不是一个对象；但将“人”这个概念具体到某个熟知的个体——例如张三时，他的容貌声音、行为举止就立刻附着在那个模型人身上，此时“人”这个概念被具体化了，形成了一个独特的对象。从这个例子可以看到，抽象是所有对象的模板，而对象是抽象的一个具体实例。

在现实中，用**属性**和**行为**这一静一动两个术语来描述抽象的特性。例如，人拥有姓名、年龄和性别等静态属性，拥有思维、行走和说话等动态行为。而对于抽象概念“人”，这些属性和行为还都没有具体的值或方式。只有在描述一个特定的对象时，人的属性和行为才会具体化。例如，对象张三，年龄 22，性别男，按某种非常具有张三特色的方式思维、行走和说话。这样一来，对象就真正活起来了。图 1-1 示意了对象的抽象与具象之间的关系。

一个对象是一个主动的实体，它能够主动发起动作，从而引起其内部状态的改变。一个显而易见的例子就是，张三是用他自己的双腿靠自己的意识支配行走的。一旦走了几步后，他所处的位置就发生了改变。

对象和其他对象之间是有联系的，它们之间要产生互动，从而驱动问题向能够解决的方向发展。例如，在张三与李四的交流过程中，张三在讲，李四在听、在想。可以看到，一个对象发起一个动作，将刺激另一个对象发起响应动作，这样一来一往，双方的交流就得以顺利进行。

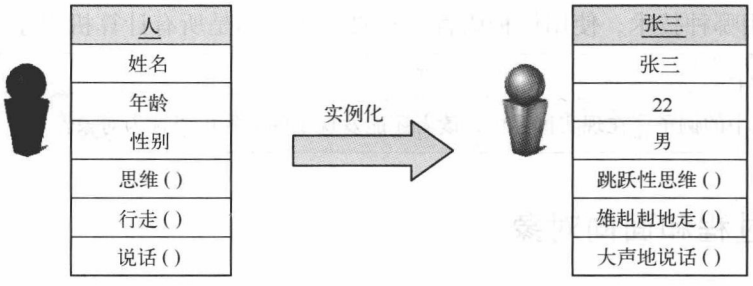

图 1-1　抽象和具象（对象）的关系

现实世界中充满了对象，并且一切皆可成为对象。因此，在认识客观世界时，首先接触到的都是一些独立的对象，然后从这个基点出发，去认识更多的与之关联的对象。随着关联网络的扩大，会将原来相对独立的知识串联起来，最后拼接出一幅较完整的画卷。从对这个画卷的阅读中，能够掌握更全面的知识。可以说，这是一种从局部出发，最后到获得全局观的认知过程。

1.2.1　计算机中的对象

在理解了现实世界中的对象后，下面再来观察在计算机世界中是如何实现对象这一概念的。

众所周知，现代的计算（Computing）实际上不再仅仅是数学上的数值计算，而是一种对事务处理的仿真，而仿真的第一步是建模（Modeling）。一般地，会为问题建立现实和计算机两种模型，后者是对前者的仿真。由于现实模型主要由对象构成，因此计算机模型也应该用某种技术去刻画对象。换句话说，就是在计算机中，对象应该有一种具体的表达方式。

现实的对象是一种存在，那么这个对象在计算机中的映像也必须是一种存在。换句话说，就是在模拟现实运作的计算机程序中，对象必须以代码的形式存在。由于程序代码是被装入到内存当中得以运行的，因此，用计算机技术描述的对象是存在于内存中的。基于此，可以简单地为计算机处理的对象下一个定义：**一段带有特定类型的内存**。这个定义告诉了人们以下 3 个事实。

1）对象是程序运行时用到的数据，它们要占据一定大小的内存。

2）对象属于某种（数据）类型。类型是一种重要信息，它描述了对象要占据多大的内存、对象的细节在这段内存中是如何分布的，以及对象能够参与的运算。

3）对象是可操控的。

根据以上描述，读者可能很容易地联想到程序设计中常用到的“变量（Variables）”，因为变量实实在在拥有上述特征。基于此，可以说变量就是对象，只不过，由于变量缺乏主动的行为，因此还只能是一种简单对象，而真正意义上的对象要比变量复杂得多。

建立了对象的概念之后，解决问题的计算机程序就会围绕对象来进行设计。不过，不同的计算机程序设计技术对对象这个概念会有不同的看法，因而设计模式也会有所不同。目前，通常采用的设计技术有两种：面向过程的和面向对象的设计技术，这两种技术用两类对应的程序设计语言实现。

但无论采用哪种技术，使用何种语言，解决问题永远是所有计算机程序唯一的目标。

习题 1.1

除了书中的例子，在现实世界中，读者还能发现哪些事物可以称为对象？

1.2 面向过程和面向对象

通常，人们利用计算机来解决实际问题，实际问题的解决步骤会映射到一种程序设计语言中。而这些程序设计语言的编码一般都围绕以下两类事物展开。

- 数据。
- 计算方法（算法）。

对以上两者关系的处理方式衍生了多种程序设计方法，而其中最流行的就是面向过程和面向对象的方法。

1.2.1 面向过程方法

面向过程的观点以计算方法为重。这显然是一种注重过程的观点，也是这种程序设计方法得名的缘由。

在早期的程序设计技术中，对过程的仿真成为最主要的关注点。这种观点在很多早期的程序设计语言中（如 Pascal、FORTRAN 等）得以充分体现。在这些语言的程序中，最显著的、占主导地位的语法成分就是“**过程/子例程（Procedure/Subroutine）**”，在一些语言（如 C 语言）中又称为“**函数（Function）**”，它们是程序的主宰。与此相对应的，待处理的数据并没有仿真实际对象，而是退化成仅包含基本属性的最小数据封包，而没有包含对象应有的行为。严格地说，对象的行为还是存在的，只不过它们从对象中分离了出来，成为一个个独立的过程。当对象要发起一个动作时，一般是通过“**过程（函数）调用（Procedure/Function Call）**”来完成的，对象是这个过程的一个“**参数（Parameter）**”。通俗地说，就是动作作用于对象之上。这样一来，对象从一个主动的实体“沦落”为一个被动的附属品，用通俗的话来讲，就是人“行走”变成了“被行走”。显然，这种观点或多或少与人的自然思维相悖。

过程和函数最直观的区别在于：函数一般要计算出一个（或一些）结果，并且要将结果返回到调用这个函数的地方；而过程往往不会返回结果。

面向过程技术将对象及其行为截然分开的特性有其合理性，但也存在着一些弊端。

1）描述对象特性的数据包没有任何或者只有很弱的保护措施。也就是说，任何人可以直接访问数据封包中的元素而不需要任何特殊手段。而现实的情况是：对象的有些属性是应该被隐蔽的、外部不可见的。

2）对象的属性和行为之间的联系非常松散，这降低了客户程序员（即那些使用数据封包的程序员，他们不是该数据封包的创建者）对整体逻辑的可理解程度。

3）如果说现实模型和面向过程的计算机模型之间存在着映射关系，那么这个映射关系是有些“扭曲”的，正如图 1-2 描述的那样。

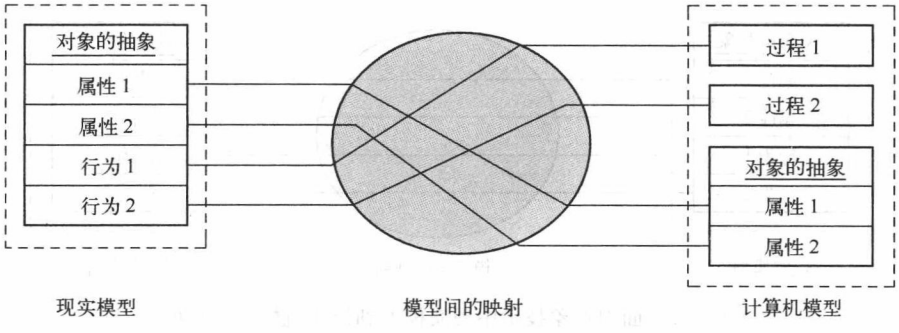

图 1-2　面向过程技术中现实模型到计算机模型的映射

在程序设计方面，面向过程程序设计采用了“自顶向下，逐步求精”的方式，模块化（或结构化）成为其最重要的思维方法，其具体设计步骤如下。

1）整个软件系统功能逐步细化为多个小的功能。

2）每个小的功能由一个模块（如函数、过程、分程序和子程序等）来实现。

3）多个模块合作完成较大的功能，所有模块的合作完成整个软件系统的功能。

当程序规模不是很大时，面向过程的方法会体现出一种优势，因为程序的流程很清楚，按着模块与函数的方法可以很好地组织整个程序结构。然而，随着软件规模的扩大，软件结构变得越来越复杂，软件开发的困难也越来越大，其中两个最被关注的问题如下。

1）怎样克服软件的复杂性。

2）怎样将现实世界模型在计算机中自然地表示出来。

面向过程设计方法的先天缺陷使之完成这些任务显得非常艰巨。因此，换一种思维方式去解决问题是非常必要的。

1.2.2　面向对象方法

随着软硬件技术的进步，以及人类认知能力的提高，面向对象（Object - Oriented）技术应运而生。

依据面向对象技术的观点，客观世界是由大量对象构成的，每一个对象都有自己的运动规律和内部状态，不同对象之间的相互作用和互相通信构成了完整的客观世界。因此，从思维模型的角度来看，面向对象很自然地与客观世界相对应。

计算是一种仿真。如果每个被仿真的对象都由一个特定的数据结构来表示，并且将相关的操作信息封装进去，那么就可以更方便地刻画对象的内部状态和运动规律，因此仿真将被简化。面向对象就是这样一种适用于直观模型化的设计方法。这意味着系统设计者从现实世界所得到的图像，或设计者头脑中形成的模型里所出现的物理图像与构成系统的一组对象之间有近乎一对一的对应关系（见图 1-3）。这一思想非常利于实现大型的软件系统。

相信读者已经注意到图 1-3 与图 1-2 有一个显著的不同：在图 1-3 中，对象的行为与属性一样，是封装于对象内部的，它们是明确地属于而非独立于对象的；而图 1-2 中不存在这样的直接关系。而前者描述了面向对象技术的一个非常重要的特征。

作为克服软件复杂性的手段，在面向对象技术中，利用了对象的如下性质。

1）将密切相关的数据和过程封装起来定义为一个实体。

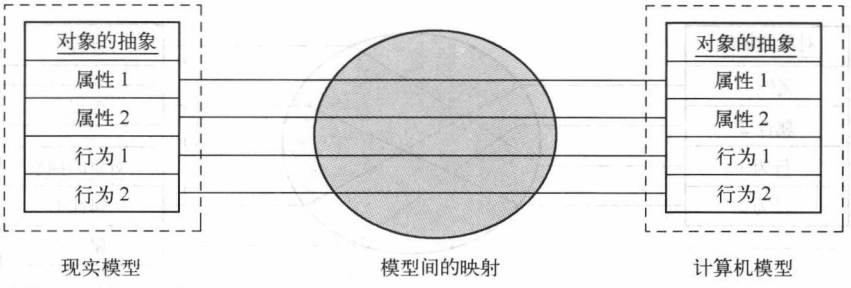

图 1-3　面向对象技术中现实模型到计算机模型的映射

2）定义了一个实体后，即使不知道此实体的功能是怎样实现的，也能使用它们。

这相当于软件工程和程序设计方法论中的抽象化、抽象数据类型和信息隐藏等概念所具有的性质。它把系统中的所有资源，如数据、模块及系统都看作对象；对象把一组数据和一组过程封装在一起，这组过程对这组数据进行处理。使用这一方法，设计人员可以依照自己的意图创建自己的对象，并将问题映射到该对象上。该方法直接、自然，可以使设计人员把主要精力放在系统一级，而较少关心细节问题。以常用的计算机作为例子，作为对象的计算机将其内部复杂结构封装在了机箱中，外部保留了显示、输入等公共接口，操作人员可以通过这些接口方便地使用计算机，而并不需要知道计算机的内部组成及接口是如何实现的。

更深入一点，使用对象将信息局部化，并使程序结构与设计结构相吻合的优点，有利于在完善和维护阶段对软件进行修改，也有利于其他人（非设计人员）来清除软件错误。程序员容易确定程序的哪些部分依赖于正要修改的片断，而且正在修改的部分对其他部分影响很小。这对大型、复杂软件的维护和改进是很重要的。

面向对象设计非常注重设计方法，因为它要产生一种与现实具有自然关系的软件系统，而现实就是一种模型。实际上，用面向对象方法编程的关键是模型化。程序员的责任是构造现实的软件模型。此时，计算机的观点是不重要的，而现实生活的观点才是最重要的。

因此，对试图利用计算机进行问题求解和信息处理的领域，尽量使用对象的概念，将问题空间中的现实模型映射到程序空间，由此所得到的自然性可克服软件系统的复杂性，从而得到问题求解和信息处理的更高性能。

习题 1.2

除了 C 语言，读者还用过或者知道哪些程序设计语言可以归类为面向过程的语言？与 C 语言相比，它们都有哪些异同？

习题 1.3

除了本书介绍的 C++，读者还用过或者知道哪些程序设计语言可以归类为面向对象的语言？它们各自拥有什么鲜明的特点？

1.3　面向对象技术的核心概念

在问题空间中，客观存在的实体称为“**对象**”，不同对象之间的相互作用和互相通信构

成了完整的客观世界。如何将问题空间的这一思维模型直接映射到程序空间，也就是说，面向对象语言提供什么概念来支持这一思维模型？纵观诸多面向对象的程序设计语言，最核心的概念可以概括为：**数据封装（抽象）**（Data Encapsulation）、**继承**（Inheritance）、**多态性**（Poly－morphism）和**泛型编程**（Generic Programming）。

1.3.1 数据封装

使用传统设计方法的程序员往往有着共同的弱点。用一个较为夸张的例子来说，当人们问到有些程序员时间的时候，他们会详细地解释钟表的工作原理。这是由他们使用计算机的经验所致。这里，必须搞清楚的概念是，告诉别人时间和让他人理解时间的概念是两个不同的知识领域。例如，某人有一块手表，并常常告诉别人现在是几点了，这并不等于说他必须知道这块表的内部工作原理。手表是一个对象，使用者只需知道这个对象可以提供当前的时间，而无需了解它如何运转。数据封装的概念反映的就是这一简单原理。

从上述例子可以推论出数据封装的特点如下。

1）不需要被外部了解的信息被封装隐藏了。

2）这些隐藏的信息是被保护的。如果没有授权，外部是不能直接访问这些信息的。

3）封装刻画了内部信息之间的联系。

4）封装对外提供了访问接口。使用这个封装系统的人员不会也不必去关心该系统的组成和工作原理；通过使用接口就可以获取所需的功能。

那么，数据封装在 C ++ 中是如何实现的呢？

这里举一个例子来说明。当需要使用“时间”这个数据时，可以设计一个 C 风格的结构体来表示时间信息，同时设计一些与之相关的操纵函数（这里称为**接口**）。这些程序元素一般会放在一个头文件中，这里将其命名为 mytime. h。以下是时间结构体的 C 代码。

```
struct time
{
    int hour,minute,second;
};
extern int getHour(time *theTime);
extern time* addHour(time *theTime,int h);
//其他时间操纵函数
```

可以看到，结构体 time 在一定程度上完成了数据封装：结构体描述了时、分、秒数据之间的内在联系，接口获得的将是关于时间的整体描述；反之，用零散变量表示的时、分、秒数据则没有展示出它们之间的联系。

当其他程序员（如 David）需要使用上述结构体的时候，将会在他们的源代码中包含头文件。

```
#include "mytime.h"
```

然后就可以在代码中自由地使用关于 time 的一切元素。