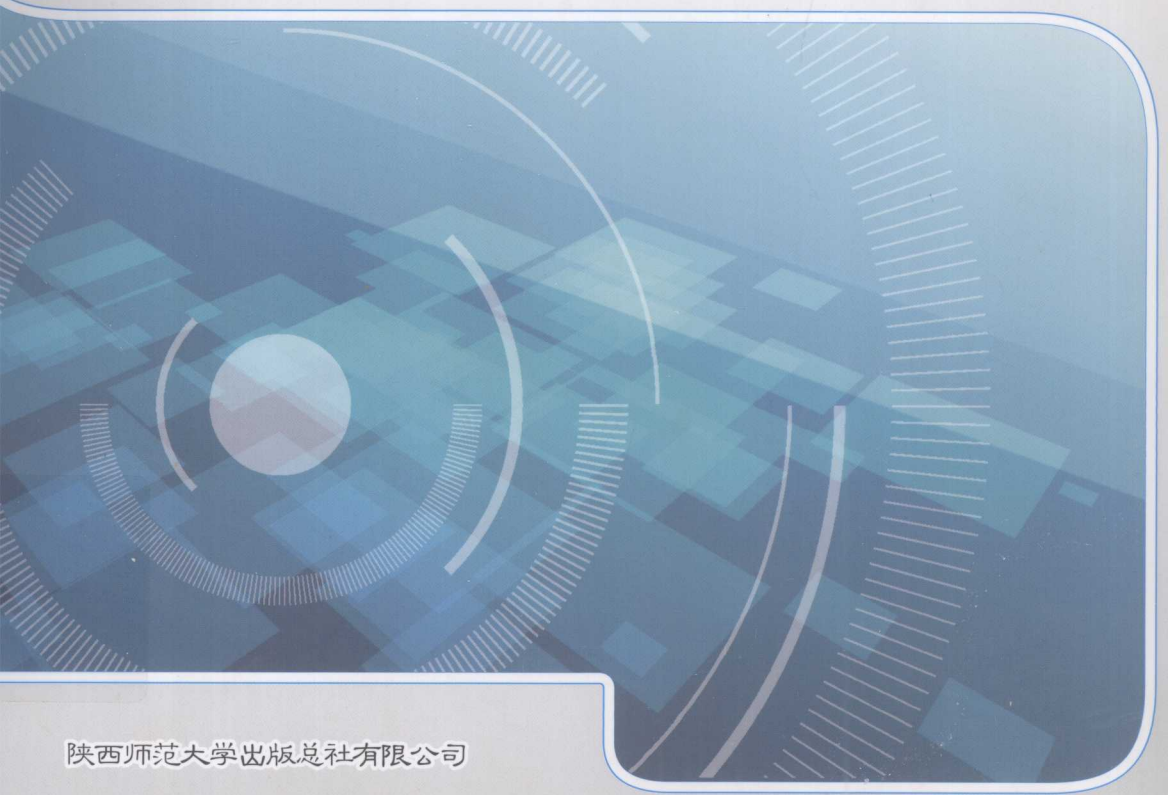
面向对象程序设计的思想和方法：用C++描述

张郭军　哈渭涛　著



陕西师范大学出版总社有限公司

陕西省教育厅自然科学研究计划项目(项目编号:12JK0745)
陕西省专业综合改革试点建设项目(项目名称:基于校企合作协同创新的实践应才培养模式创新实验区)
渭南师范学院学术著作出版基金资助出版

面向对象程序设计的思想和方法:用C++描述

张郭军　哈渭涛　著

陕西师范大学出版总社有限公司

图书代号　JC14N0322

图书在版编目(CIP)数据

面向对象程序设计的思想和方法：用C++描述／张郭军，哈渭涛著．—西安：陕西师范大学出版总社有限公司，2014.4
ISBN 978-7-5613-5994-5

Ⅰ．①面…　Ⅱ．①张…②哈…　Ⅲ．①C语言—程序设计　Ⅳ．①TP312

中国版本图书馆CIP数据核字(2014)第061506号

面向对象程序设计的思想和方法：用C++描述
张郭军　哈渭涛　著

责任编辑／钱　栩　李红凯
责任校对／杜世雄
封面设计／王　宇
出版发行／陕西师范大学出版总社有限公司
(西安市长安南路199号　邮编710062)
网　　址／http://www.snupg.com
经　　销／新华书店
印　　刷／西安永琛快速印务有限责任公司
开　　本／787mm×960mm　1/16
印　　张／12.75
字　　数／243千
版　　次／2014年4月第1版
印　　次／2014年4月第1次印刷
书　　号／ISBN 978-7-5613-5994-5
定　　价／25.50元

读者购书、书店添货或发现印刷装订问题，请与本社高教出版分社联系、调换。
电　话：(029)85303622(传真)　85307826

前　言

面向对象(Object Oriented,OO)是当前计算机界关心的重点,它是现行软件开发方法的主流。面向对象的概念和应用已超越了程序设计和软件开发,扩展到数据库系统、交互式界面、应用结构、应用平台、分布式系统、网络管理结构、CAD 技术、人工智能等广泛领域。面向对象技术主要包括面向对象分析和面向对象程序设计。所谓面向对象的程序设计,就是把面向对象的思想应用到软件工程中,并指导开发维护软件。面向对象程序设计技术的提出,主要是为了解决传统程序设计方法——结构化程序设计所不能解决的代码可重用问题。

面向对象的程序设计(OOP)的许多原始思想都来之于 20 世纪 60 年代的 Simula 语言,并在 Smalltalk 语言的完善和标准化过程中得到更多的扩展和对以前的思想的重新注解。而 OOP 的流行主要源自于 C++ 和 Java 这类语言的传播,使得 OOP 在现代程序设计中显得如此重要。从理论上讲,用面向对象的语言可以处理任何其他计算机语言所能完成的事情。然而当建立基于智能体的模型时,OOP 对于开始的程序设计和后来的程序使用者都表现出了强大的优势。

本书主要针对面向对象程序设计方法中继承、封装、多态性三大基本机制来展开阐述。继承是一种联结类的层次模型,并且允许和鼓励类的重用,对象的一个新类可以从现有的类中派生,新类继承了原始类的属性和行为。派生类可以从它的基类那里继承方法和实例变量,并且派生类可以修改或增加新的方法使之更适合特殊的需要。继承性很好地解决了软件的可重用性问题。封装是面向对象的特征之一,是对象和类概念的主要特性。封装是把过程和数据包装起来,对数据的访问只能通过已定义的接口。一旦定义了一个对象的特性,则有必要决定这些特性的可见性,即哪些特性对外部世界是可见的,哪些特性用于表示内部状态。信息的隐藏意味着应禁止直接访问一个对象的实际表示,而应通过操作接口访问对象。多态性是指允许不同类的对象对同一消息做出不同的响应,多态性语言具有灵活、抽象、行为共享、代码共享的优势。

在由封装、继承、多态所组成的环境中，程序员可以编写出比面向过程模型更健壮，更具扩展性的程序。经过仔细设计的类，层次结构是重用代码的基础，封装能让程序员不必修改公有接口的代码即可实现程序的移植，多态能使程序员开发出简洁易懂、易修改的代码。

本书包含大量的代码，几乎演示了 C++ 描述的 OOP 的所有特性。我们有意使用短小精炼的实例来突出重点，希望您在阅读本书过程中能获得乐趣，并对您关于 OOP 的思想和方法的理解提供帮助。

本书共分 6 章，第 1 章主要以“软件危机”产生的渊源为切入点，讨论面向对象程序设计的基本概念、基本机制和方法，以及对程序设计发展的历史变革和影响。第 2 章通过 C 向 C++ 的演变过程，提出了 C++ 中的一些新概念。第 3 章主要讨论了数据抽象和封装在 C++ 中的实现。第 4 章着重研究了 C++ 中的代码重用思想，也就是 C++ 中的继承问题。第 5 章以 C++ 中的重载、虚函数为例，研讨了面向对象程序设计中的多态性问题。第 6 章介绍了 C++ 中 I/O 流的思想。

著者中张郭军主要负责第 1、2、6 章的编著工作，哈渭涛完成第 3、4、5 章的编著工作，全书由张郭军统稿并定稿。

本书的编著过程中得到了渭南师范学院学术专著出版基金的资助和陕西师范大学出版社的大力支持与合作，在此一并感谢。

本书虽然是作者多年从事面向对象程序设计工作和研究的心得，但是限于作者的水平，在编写的本书中，仍难免出现错误和不当之处，恳请您的批评指正。

著　者

2013 年 6 月

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第1章　面向对象导论

当今，人们对于计算机的认识不仅仅局限在它是一台冰冷机器，仿佛更多的是一个富有表现力的媒体。例如，借助计算机这个工具，能完成写作、绘画、动画制作、特效电影制作等富有挑战与创造的工作，实际上是把计算机作为我们大脑中的有机组成部分，正如 Steve Jobs 喜欢称它为“智力的自行车”一样，我们称计算机为“电脑”。计算机之所以能成为富有“智力”的机器，是因为在其底层要运行按照我们意图而设计的程序（软件），而面向对象程序设计（OOP）的思想和方法，可以说是程序设计系统工程具有革命意义的变革，一直主导和影响着程序设计历史发展的进程。

本章以“软件危机”产生的渊源为切入点，讨论面向对象程序设计的基本概念、基本机制和方法，以及对程序设计发展的历史变革和影响。

1.1　OOP 思想和方法的革命

20 世纪 60 年代以前，软件设计往往只是为了一个特定的应用而在指定的计算机上设计和编制，采用密切依赖于计算机的机器代码或汇编语言，软件的规模比较小，文档资料通常也不存在，很少使用系统化的开发方法。设计软件往往等同于编制程序，基本上是个人设计、个人使用、个人操作、自给自足的私人化的软件生产方式。

60 年代中期，大容量、高速度计算机的出现，使计算机的应用范围迅速扩大，软件开发急剧增长。高级语言开始出现；操作系统的发展引起了计算机应用方式的变化；大量数据处理导致第一代数据库管理系统的诞生。软件系统的规模越来越大，复杂程度越来越高，软件可靠性问题也越来越显得突出。原来的个人设计、个人使用的方式不再能满足要求，迫切需要改变软件生产方式，提高软件生产率，软件危机开始爆发。其主要表现在：软件开发费用和进度失控；软件的可靠性差；生产出来的软件难以维护；用户对“已完成”的系统不满意现象经常发生。随着微电子学技术的进步和硬件生产自动化程度不断提高，硬件成本逐年下降，性能和产量也迅速提高，但软件危机不仅没有消失，而且还有加剧之势。主要表现在：软件成本在计算机系统的总成本中所占比例居高不下，且逐年呈上升趋势；软件开发生产率提高的速度远远跟不上计算机应用迅速普及深入的需要，软件产品供不应求的状况使得人类不能充分利用现代计算机硬件所能提供的巨大潜力。

软件工程诞生于 20 世纪 60 年代末期，它作为一个新兴的工程学科，主要研究软件生产的客观规律性，建立与系统化软件生产有关的概念、原则、方法、技术和工具，指导和支持软件系统的生产活动，以期达到降低软件生产成本、改进软件产品

质量、提高软件生产率水平的目标。软件工程学从硬件工程和其他人类工程中吸收了许多成功的经验,明确提出了软件生命周期的模型,发展了许多软件开发与维护阶段适用的技术和方法,并应用于软件工程实践,取得了良好的效果。

在软件开发过程中,人们开始研制和使用软件工具,用以辅助进行软件项目管理与技术生产,人们还将软件生命周期各阶段使用的软件工具有机地集合成为一个整体,形成能够连续支持软件开发与维护全过程的集成化软件支援环境,以期从管理和技术两方面解决软件危机问题。

此外,人工智能与软件工程的结合成为 20 世纪 80 年代末期活跃的研究领域。基于程序变换、自动生成和可重用软件等软件新技术研究也已取得一定的进展,把程序设计自动化的进程向前推进了一步。在软件工程理论的指导下,发达国家已经建立起较为完备的软件工业化生产体系,形成了强大的软件生产能力。软件标准化与可重用性得到了工业界的高度重视,在避免重用劳动、缓解软件危机方面起到了重要作用。

可以说,OOP 的思想和方法对软件工程体系的架构起到了革命性的作用。

1.2 抽象的过程

人们解决问题的复杂性直接与抽象的类型和质量有关,所有的程序设计语言都提供了抽象这一人类解决问题的基本思想和方法。汇编语言是对底层机器的有限度抽象,其后发展起来的“面向过程”的语言,如 FORTRAN、BASIC 和 C,都是对汇编语言的抽象。“面向过程”语言的抽象虽然使得程序更具结构化、易读性,但其解决问题的思想架构在计算机解题实现的步骤中,主要是对具体问题的解题过程的抽象,程序设计者必须在计算机的机器模型(“解空间”)和实际要解决问题的解模型(“问题空间”)之间建立联系。而“解空间”与“问题空间”之间对应关系的建立,实际上是程序设计语言之外的任务。

OOP 的抽象思想是对要解决的问题直接进行建模,而与“解空间”没有关系。这种抽象思想为程序设计者提供了在“问题空间”中表示各种事物元素的通用方法,与要解决问题的类型无关。OOP 把“问题空间”中的事物和它们在“解空间”中的表示统一抽象为对象,其思想是允许程序通过构建新的对象类型而使程序本身能够根据问题的解决诉求进行调整。每个对象有属性(状态),有行为(可以执行的运算),当给面向的对象发送消息时,它就会激活对象的行为特征,就会得到抽象类型下某个问题解的实例。这种高度抽象的思想和解决问题的方法,和人类认识客观世界方法论的思维习惯和定势取得了高度一致。

OOP 抽象过程的思想与方法,主要体现在以下几个方面:

(1)任何问题都可封装为对象。对象是其本身的数据属性及施加在其运算行为的封装体,对其发送要执行什么样操作的消息,对象就能执行什么样的运算,从而表现出特有的行为特征。

(2)程序实质上是一组对象,对象之间可以通过发送消息(事件)驱动(触发)相应代码的执行。

(3)对象有自己的数据类型。采用 OOP 的术语,所谓对象就是类的实例(变量),类最重要的突出特征就是发送消息的机制。

(4)特定类型的所有对象都能接收相同的消息,但可能会产生不同的行为特征,从而表现 OOP 中的多态性。

(5)对象成员的机制使得程序设计者可以构造出复杂的程序,而这种复杂性隐藏在对象的抽象中,从而提供了复杂性简明化的有效途径。

1.3 对象的接口

第一位引入类型(type)概念的人可能是亚里士多德,他较早提出了"鱼类和鸟类"的概念。所有对象都是一类对象中的一员,它们拥有共同的属性和行为。这一思想在 OOP 中得到了直接的应用,C++ 语言用关键字"class"在程序中引入了新的构造类型。

创建抽象数据类型是 OOP 的基本思想。抽象数据类型在 OOP 的实现中具体化为用户自定义的数据类型,用它创建的类型变量,称之对象或类的实例,通过向对象发送消息或请求来操纵对象。可见,类是描述了一组具有相同属性和相同行为的对象,实际上就是数据类型,区别在于类数据类型是和具体问题相适应,程序设计者可以根据需要来对程序设计语言进行扩展。

对象的行为由消息驱动,而消息发出的请求是由接口(interface)定义的,而接口由类型确定。接口定义了能向特定的对象发出什么样的消息,由相应的代码响应消息的请求,加上数据的隐藏,就组成了基于 OOP 程序的实现。从面向过程程序设计的观点来看,消息的发送对应函数的调用,对象根据消息确定所调用的函数,最终驱动了代码的执行。

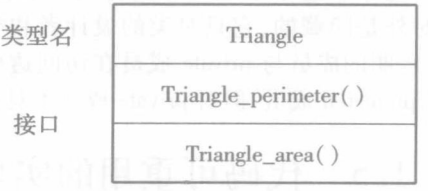


图 1-1 Triangle 对象示例

图 1-1 是按统一建模语言(Unified Modeling Language,UML)的规范书写的,每个类由一个方框表示,方框的顶部标有类型名,中间部分列出所关注的数据成员,底部是成员函数。通常,只有类的名字和公有成员函数在 UML 设计图中表示。

Triangle Tri; //创建一 Triangle 对象。

Tri. Triangle_perimeter( ) ; //发送 Triangle_perimeter( )消息,执行 Triangle_perimeter( )代码,产生行为。

此为试读,需要完整PDF请访问: www.ertongbook.com

Tri. Triangle_area( ) ; //发送 Triangle_area( )消息,执行 Triangle_area( )代码,产生行为。

1.4 数据的封装与隐藏

OOP 通过封装与隐藏以实现数据抽象,这种封装最终表现为称之为类的用户自定义数据类型的形式。这样,可以把要处理的问题直接在程序中表示为类的实例,亦即对象,且可用标识符进行标识。

OOP 实现抽象通过封装机制将问题要处理的数据及实施在处理数据上的操作封装成一个整体,即在类所封装的成员包括数据成员(属性)和成员函数(行为)。这样,从程序设计者的角度,可以将程序员分为类的创建者和应用类的客户程序员。类的创建者的目标是通过类的设计完成对问题解的抽象,而客户程序员仅关心类的接口,创建类的实例,通过对象之间发送消息,完成具体问题的解决。

OOP 在抽象过程中另外一个重要机制,是对封装的数据进行了访问权限的控制,即数据的隐藏机制。

对封装的成员进行访问权限控制的原因之一,是出于数据安全的考虑,也就是说,对于数据类型内部问题解决方案的保护实际上给客户程序员提供了更好的服务,使得客户程序员在类的使用过程中明确了哪些是他们关心的,哪些对他们来说是重要的,哪些是他们可以忽略的。

对封装的成员进行访问权限控制的原因之二,是由于类内部工作方式的改变只允许由类的设计者完成,而不必担心这种改变对类的使用者带来影响,这种类接口部分的开放性及类实现部分对外的隐藏性,使得类的设计者对类的内部工作方式重新设计而不会对类的使用带来任何影响。

在 C++ 语言中,使用三个明确的关键字来设置类成员的访问权限。用 public 关键字所声明的成员,对外是透明的,对所有的访问者来说是可见的。用 private 关键字所声明的成员,对外是隐藏的,它只对类的设计者和类的成员函数是可见的。而用 protected 关键字声明的成员与 private 成员在访问透明度上基本相似,区别在于继承的类可以访问 protected 成员,但对 private 成员不具备访问权限。

1.5 代码可重用的实现

代码的可重用性是 OOP 的核心思想,也是基于 OOP 程序设计技术的优势之一。

实现代码可重用最直接的方法,就是在类的设计中用已有类的对象作为新类的成员,也就是创建一个对象成员,以实现对象成员中已有的成员方法的重用。

实现代码可重用另外一条途径,可以用任何数量和类型的其他对象组合来设计新类,通过组合得到新类所期望的功能。组合具有很大的灵活性,新类的成员对象通常是私有的,使用这个类的客户程序员不具有访问权限。其特点是允许改变

这些成员而不会干扰已存在的客户代码。在程序运行时可以改变这些对象成员，动态地改变程序的行为。

类的继承性也是实现代码可重用的一条重要途径，但继承缺乏组合的灵活性，因为在用继承的方法设计新类时，必须考虑父类成员的继承方式，而继承方式对父类代码的可重用访问加入了限制。

虽然继承是 OOP 中很重要的思想，但从代码可重用的角度考量，首先应考虑组合，因为它更简单和灵活，采用组合的方法，会使设计变得清晰。

1.6 继承性特征

在抽象的机制中，类型不仅仅描述同类对象的约束，更重要的是描述与其他类型之间的层次关系，类的继承提供了实现类型层次之间关系描述的机制和方法。

继承性是子类自动共享父类之间数据和方法的机制。它由类的派生功能体现。一个类直接继承其他类的全部描述，同时可修改和扩充。继承具有传递性。继承分为单继承（一个子类只有一个父类）和多重继承（一个类有多个父类）。类的对象是各自封闭的，如果没有继承性机制，则类对象中数据、方法就会出现大量重复。继承不仅支持系统的可重用性，而且还促进系统的可扩充性。

使得派生类区别于原始基类的最直接、最简单的方法，就是在派生类中添加新的数据成员和成员函数，即添加新的属性和行为，这些成员不是基类接口的一部分。继承这种对原始基类简单运用和发展的思想，往往使得问题能得到完美、高效的解决。

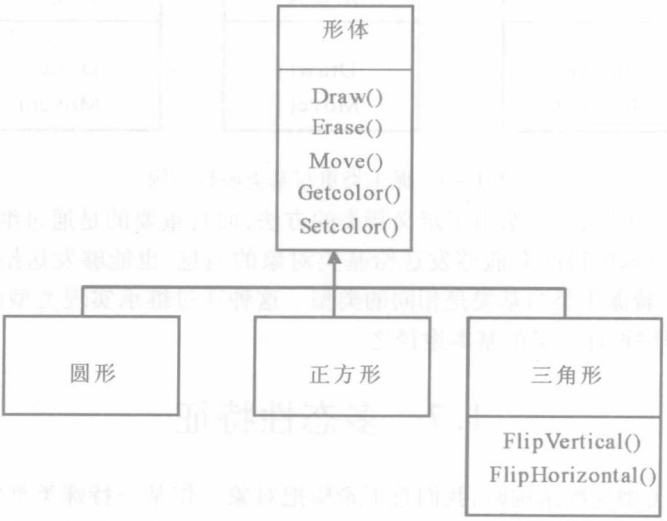


图 1-2 类继承示例

图 1-2 是经典类继承的范例。在这个范例中,基类型是“形体”,每个形体有大小、颜色、位置等属性,具有被绘制、擦除、移动、着色等行为。由“形体”这个基类可以派生出特殊类型的形体,如圆形、正方形、三角形等,派生出的子类除了具备基类的一般属性和行为外,还可具备各自的属性和行为。这种用与问题相同的术语描述问题的解是 OOP 的重要思想,该思想摒弃了问题描述与解描述之间的中间模型,可从问题的系统描述直接进入代码的系统描述。

类的继承使派生类有别于基类的第二种更重要的方法是,它允许改变基类函数的行为,即允许重写(overriding)基类的函数。但使用的是和基类函数相同的同一语义的接口,或者说发送的是同一消息。如图 1-3 所示的 Move()和 Draw()方法,在圆形、正方形和三角形中,重写了各自不同的实现,但其操作语义是相同的。同一个接口函数,在新的派生类中表现出了不同的行为特征。

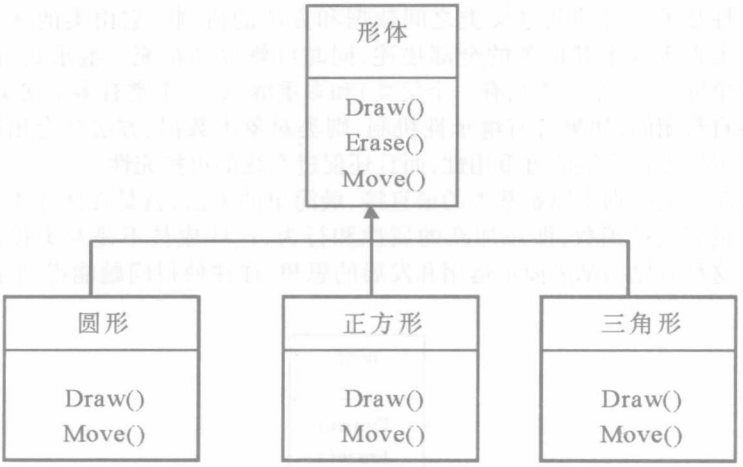


图 1-3 派生类重写基类函数示例

类的继承机制,不仅给出了定义新类的方法,而且重要的是通过继承复制了基类的接口。继承使得所有能够发送给基类对象的消息,也能够发送给派生类的对象,这就意味着派生类与基类是相同的类型。这种通过继承实现类型的等价性,是理解 OOP 程序设计含义的基本途径之一。

1.7 多态性特征

当处理类型层次结构时,我们总不希望把对象当作某一特殊类型的对象,而是把它当作基本类型的对象,这样就允许程序员编写不依赖于特殊类型的接口程序。如在“形体”的范例中,函数可以对一般形体进行操作,而不必关心它们是圆形、正

方形还是三角形。所有的形体都能被绘制、擦除和移动，所以这些函数能简单地发送消息给一个形体对象，而不必考虑这个对象如何处理这个消息。接口程序不受新增类型的影响，而且增添新类型是扩展 OOP 程序来处理新情况最普通的方法。也就是通过派生新的子类型，可以很容易扩展程序，这个思想对于软件设计具有革命性的意义。

然而，如果把派生类型的对象当作一般的基类型的对象，存在的问题是：当发送消息时，程序员并不想知道将执行哪个具体的代码。在“形体”的范例中，绘图函数接口能等同地应用于圆形、正方形或三角形，对象根据它的特殊类型来执行合适的代码。如果增加一个新的子类型，不用修改函数调用接口，它就可以执行不同的代码。这种将同一消息发送给不同对象产生的不同行为是 OOP 中重要的多态性特征，多态性特征是实现类型等价性求解的重要技术支撑。

在 OOP 中，实现继承层次下的多态是基于动态（晚期）联编（绑定）的思想。动态联编是在程序运行的过程中绑定被调用的程序代码，而不是像静态（早期）联编是在编译阶段对被调用的函数进行代码绑定。在 C++ 中，实现动态联编需要虚函数的技术支持，同时派生类型与基类型之间的派生关系是“is_a”关系，也就是派生类型是基类型的子类型，或是基类型的子类型化，或者说是基类型的类型适应。

1.8 对象的创建和释放

从技术角度分析，OOP 的思想和方法就是抽象数据类型、继承和多态性。实现 OOP 的思想和方法的重要载体就是对象。

关于对象创建和释放的方法是至关重要的。对象的数据存放在什么地方？对象的生命周期如何进行控制？不同的程序设计语言提供了不同的处理机制。

C++ 解决对象创建和释放，其机制是基于效率控制的考虑，它为程序设计者提供了一个选择。为了使运行速度最大化，通过将对象存放在栈区或静态存储区域中，存储和生命周期可以在编写程序时确定。栈是内存中的一个区域，可以直接由处理器在程序执行期间对其进行操作。来自于栈中的对象通常被称为自动对象或局部对象。静态存储区域是内存中的一个固定块，在程序开始执行之前进行分配。使用栈或静态存储区，可以快速地分配和释放对象，这对提高程序的运行效率是有益的。然而，要创建来自于栈区和静态存储区的对象，程序设计者在编写程序时就要确切知道对象的数量、生命周期和类型，这对于解决更一般的问题来讲，就受到了限制。

对于在程序运行过程中要确定创建对象的数量、生命周期，C++ 给我们提供了来自于堆内存区域动态对象的创建和释放的方法。程序设计者在程序运行时还不能确定需要的对象数量、对象的生命周期及对象的类型时，使用这种方法使动态对

象的创建和释放成为可能,更适应于一般问题的解决。动态对象的创建,可以使用 C++ 语言成分 new 运算符在堆内存上生成动态对象。动态对象的释放,可以使用 C++ 语言成分 delete 运算符完成释放。

从程序运行的效率上进行分析,动态对象在堆内存分配存储比在栈上创建对象需要更长的时间开销。但动态方法做出了一般性的逻辑假设,对于解决一般性的程序设计问题,更具灵活性,效率的优越性和处理问题的一般性与灵活性相比,显得无关紧要。

关于对象生命周期的存储持续,取决于对象创建的方式。如果在栈区或静态存储区域创建对象,编译器就能决定对象的生命存储持续并自动完成释放。如果在堆内存区创建对象,编译器不能确定对象的生命存储持续,程序设计者必须编程决定何时释放所创建的动态对象,使用 delete 运算符执行动态对象的释放任务。

1.9 OOP 的程序设计范型

面向对象程序设计是一种新的程序设计范型。这种范型的主要特征是:

程序 = 对象 + 消息

面向对象程序设计的基本元素是对象,面向对象程序的主要结构特点是:程序架构主要由类的定义和类的使用两部分组成,通过向对象发送消息以实现程序中的操作。由抽象、数据的封装与隐藏、消息与方法、代码的可重用、代码的多态性构成面向对象程序设计的基本特征。

1.10 小 结

本章的讨论希望能使读者对 OOP 程序设计的思想和方法有一定的感性认识,包括 OOP 的基本特征和程序设计的范型。OOP 程序设计思想的核心是描述问题空间概念的抽象和发送给对象的消息,这些消息描述了问题空间的活动。而 OOP 程序设计的高效性在于许多问题都能通过重用库代码得到很好的解决。

第2章　从C到C++的变革

讨论从C到C++的变革是必要的，由于标准(ANSI)C是C++的子集，许多标准C程序同时也是C++程序。但C++中更为有效的新技术，如引用、const引用、内联函数、设置函数默认参数、函数重载等机制对OPP思想的实现提供了技术支撑，并广泛地使用C++中的面向对象特性。

2.1　C++中的数据处理

数据处理(data processing)是对数据的采集、存储、检索、加工、变换和传输，数据是对事实、概念或指令的一种表达形式。“C++中的数据处理”意味着要解决程序设计中所涉及的最基本问题，即如何在C++的程序中完成对数据处理过程的描述。针对这个问题，冯·诺依曼“存储程序”的思想明确指出，数据处理的前提是存储，即处理器是从内存单元中取得处理数据的对象，在对数据处理过程中所产生的一些中间结果和最终结果同样也是存储在内存单元中。所以，数据处理要解决的一个问题就是内存空间的分配、标识及对其的读写方式。在数据处理中，要解决的另一个问题是程序中对数据的描述以及对数据运算与加工的描述。因此，内存单元的分配、数据在程序中的描述及数据的加工描述是程序设计中要解决的三个最基本的问题，数据处理的概念和方法在高级语言中具有通用性的原理。

数据类型是高级程序设计语言中的一个基本概念，其实质是讨论数据在处理过程中所涉及数据最基本的两个属性：数据的存储属性和数据的操作属性。计算机在对数据的处理过程中，将数据进行分类。不同的数据在内存中的存储形式是不同的，如整数与实型数据，在内存中有不同的存储方式。不同的数据，其所允许操作的集合也是不同的，如整型数与实型数所允许的运算集合是不同的。数据类型的概念正是基于数据不同的存储属性及操作集合(属性)将数据进行分类，从而形成不同类型的数据集合，这便是数据类型的概念，或者说这是数据类型所讨论的主要课题。而数据的存储属性与操作属性的讨论，具体涉及的就是数据在程序中的描述、数据在内存中的存储方式及内存单元的分配、在程序设计中如何对数据的加工进行描述等。需要强调的是，任何高级语言认为数据都是有类型的，这同样具有通用性的原理。

在C++中，将数据类型分为基本数据类型、导出的数据类型和抽象的数据类型。基本数据类型的讨论是所有数据类型讨论的基础。

此为试读，需要完整PDF请访问：www.ertongbook.com