C++面向对象程序设计

李素若 任正云 张牧 陈万华 编著

化学工业出版社
·北 京·

本书首先介绍了面向对象程序设计理论的基本概念，让读者从理论上理解面向对象程序设计与结构化程序设计的不同之处，然后介绍 C++对 C 语言的扩充，最后介绍了 C++面向对象程序设计的基本方法。本书直接介绍面向对象的程序设计，并贯穿始终，力求让读者尽快地建立起面向对象编程的思想。读者阅读本书后不仅学会一门程序设计语言，还能初步掌握面向对象的程序设计方法。

在本书的编写中，编者结合自己的教学和编程实践经验，力图用生动、通俗易懂的语言并结合编程实例来讲解各个知识点，便于读者理解和掌握。本书主要内容包括：面向对象程序设计的基本概念，C++对 C 语言在非面向对象方面的扩充，类和对象的定义和应用，以及使用过程中应注意问题，在 C++中怎样实现面向对象的三大特性：封装性，继承性和多态性，C++模板和 C++的输入和输出等。另外，为了方便学生上机实践，本书还专门设计了 10 套上机实验题，供读者上机练习，还讨论了 Visual C++集成开发环境的组成、编译和运行 C++程序的方法、程序调试技术。本书中的所有例子都已用 Visual C++6.0 调试通过。

本书是供普通高等院校计算机科学与技术专业本、专科生使用的教材，也可供从事计算机软件开发的科研人员使用。

图书在版编目(CIP)数据

C++面向对象程序设计 / 李素若等编著. —北京：化学工业出版社，2008.6

ISBN 978-7-122-02912-6

Ⅰ. C… Ⅱ. 李… Ⅲ. C 语言-程序设计-高等学校：技术学院-教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2008）第 070058 号

责任编辑：王听讲　　文字编辑：陈　元

责任校对：宋　玮　　装帧设计：刘丽华

出版发行：化学工业出版社（北京市东城区青年湖南街 13 号　邮政编码 100011）

印　　刷：北京永鑫印刷有限责任公司

装　　订：三河市万龙印装有限公司

787mm×1092mm　1/16　印张 18¾　字数　444　千字　　2008 年 7 月北京第 1 版第 1 次印刷

购书咨询：010-64518888（传真：010-64519686）　售后服务：010-64518899

网　　址：http: // www. cip. com. cn

凡购买本书，如有缺损质量问题，本社销售中心负责调换。

定　　价：29.00 元

前　言

随着面向程序设计方法的不断普及和应用，学习和掌握 C++语言已经成为许多计算机专业工作者和广大计算机应用人员的迫切需要。学好 C++，可以很容易地触类旁通其他语言，如 Java 和 C#等。C++架起了通向强大、易用、真正的软件开发应用的桥梁。

C++语言是在 C 语言基础上扩充了面向对象机制而形成的一种面向对象程序设计语言，它除了继承了 C 语言的全部优点和功能外，还支持面向对象程序设计。C++现在已成为介绍面向对象程序的首选语言。学习 C++不仅可以深刻理解和领会面向对象程序设计的特点和风格，掌握其方法和要领，而且可以使读者掌握一种十分流行和实用的程序设计语言。

本书首先介绍了面向对象程序设计理论的基本概念，让读者从理论上理解面向对象程序设计与结构化程序设计的不同之处，然后介绍 C++对 C 语言的扩充，最后介绍了 C++面向对象程序设计的基本方法。本书直接介绍面向对象的程序设计，并贯穿始终，力求让读者尽快地建立起面向对象编程的思想。读者阅读本书后不仅学会一门程序设计语言，还能初步掌握面向对象的程序设计方法。

在本书的编写中，编者结合自己的教学和编程实践经验，力图用生动、通俗易懂的语言并结合编程实例来讲解各个知识点，便于读者理解和掌握。本书主要内容包括：面向对象程序设计的基本概念，C++对 C 语言在非面向对象方面的扩充，类和对象的定义和应用，以及使用过程中应注意问题，在 C++中怎样实现面向对象的三大特性：封装性、继承性和多态性，C++模板和 C++的输入和输出等。另外，为了方便学生上机实践，本书还专门设计了 10 套上机实验题，供读者上机练习，还讨论了 Visual C++集成开发环境的组成、编译和运行 C++程序的方法、程序调试技术。本书中的所有例子都已用 Visual C++6.0 调试通过。

本书主要由李素若主编并统稿，第 1、2、5、10、11 章及附录由李素若编写，第 3、4 章由任正云编写，第 6、7 章由张牧编写，第 8、9 章由陈万华编写。

由于编者水平有限，加之时间仓促，书中难免有不足之处，敬请广大读者批评指正，以使本书质量得到进一步提高。

编　者

2008 年 3 月

目　　录

第 1 章　面向对象程序设计概述

1.1　什么是面向对象程序设计

1.1.1　新的程序设计范型

面向对象程序设计是一种新的程序设计范型（Paradigm）。程序设计范型是指设计程序的规范、模型和风格，它是一类程序设计语言的基础。一种程序设计范型体现了一类语言的主要特征，这些特征能用以支持应用领域所希望的设计风格。不同的设计范型有不同的程序设计技术和方法学。

面向过程程序设计范型是使用较广泛的程序设计范型，这种范型的主要特征是，程序由过程定义和过程调用组成，即程序=过程+调用。基于面向过程程序设计范型的语言称为面向过程性语言，如 C、PASCAL、Ada 等都是典型的面向过程性语言。函数式程序设计范型也是较为流行的程序设计范型，它的主要特征是，程序被看作“描述输入与输出之间关系”的数学函数。LISP 是支持这种范型的典型语言。除了面向过程程序设计范型和函数式程序设计范型外，还有许多其他的程序设计范型，如模块程序设计范型（典型语言是 Modula）、逻辑式程序设计范型（典型的语言是 PROLOG）、进程式程序设计范型、类型系统程序设计范型、事件程序设计范型、数据流程程序设计范型等。

面向对象程序设计是一种新型的程序设计范型。这种范型的主要特征是：

程序=对象+消息

面向对象程序的基本元素是对象，面向对象程序的主要结构特点是：第一，程序一般由类的定义和类的使用两部分组成，在程序中定义各对象并规定它们之间传递消息的规律。第二，程序中的一切操作都是通过向对象发送消息来实现的，对象接收到消息后，启动有关方法来完成相应得操作。一个程序中涉及的类，可以由程序设计者自己定义，也可以使用现成的类（包括类库中为用户提供的类和他人已构建好的）。尽量使用现成的类，是面向对象程序设计范型所倡导的程序设计风格。需要说明的是，某一种程序设计语言不一定与一种程序设计范型相对应。实际上存在有具备两种范型或多种范型的程序设计语言，即混合型语言。例如 C++就不是纯粹的面向对象程序设计范型，而是面向过程程序设计范型和面向对象程序设计范型的混合型程序设计语言。

1.1.2　面向对象程序设计基本概念

1．对象

首先需要搞清楚的第一问题是：什么是对象？对象具有两方面含义，即在现实生活中的含义和在计算机世界中的含义。

在我们所生活的现实世界中，“对象”无处不在。在我们身边存在的一切事物都是对象，例如一粒米、一本书、一个人、一所学校，甚至地球，这些都是对象。除去这些可以触及的事物是对象外，还有一些无法整体触及的抽象事件，例如一次演出、一场球赛、一次借书，也都是对象。

一个对象既可以非常简单，又可以非常复杂，复杂的对象往往可以是由若干个简单对象组合而成的。

所有这些对象，除去它们都是现实世界中所存在的事物之外，它们都还是有各自不同的特征。例如一粒米，它首先是一粒米这样一个客观存在。再例如一个人，首先它是一个客观实体，具有一个名字来标识，其次它具有性别、年龄、身高、体重等这些体现他自身状态的特征；再其次他还具有一些技能，例如会说英语、会修电器等。

通过上面的这些举例可以对“对象”下一个定义，即对象是现实世界中的一个实体，它具有如下特性：

- 有一个名字以区别于其他对象；
- 有一个状态用来描述它的某些特征；
- 有一组操作，每一个操作决定对象的一种功能或行为；
- 对象的操作可分为两类：一类是自身承受的操作，一类是施加于其他对象的操作。

在面向对象程序设计中，对象是描述其属性的数据以及对这些数据施加的一组操作封装在一起构成的统一体。对象可以认为是：数据+操作。对象所能完成的操作表示它的动态行为，通常也把操作称为方法。

为了帮助读者理解对象的概念，图 1-1 形象地描述了具有 3 个操作的对象。

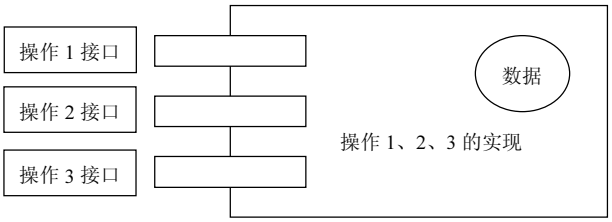

图 1-1　3 个操作对象

下面用一台录音机比喻一个对象，通俗地说明对象的某些特点。

录音机上有若干按键，如 Play（播放）、Rec（录音）、Stop（停止）、Rew（倒带）等，当人们使用录音机时，只要根据自己的需要如放音、录音、停止、倒带等按下与之对应的键，录音机就会完成相应的工作。这些按键安装在录音机的表面，人们通过它们与录音机交互。我们无法操作录音机的内部电路，因为它们被装在机壳里，录音机的内部情况对于用户来说是隐蔽的，不可见的。

一个对象很像一台录音机，当在软件中使用一个对象的时候，只能通过对象与外界的接口操作它。对象与外界的接口也就是该对象向公众开放操作。使用对象向公众开放的操作就好像使用录音机的按键，只需知道该操作的名字（如录音机的键名）和所需要的参数（用于提供附加信息或设置状态，好像听录音前先装录音带并将录音带转到指定位置），根本无需知道实现这些操作的方法。事实上，实现对象操作的代码和数据是隐藏在对象内部

的，一个对象好像是一个黑盒子，表示它内部的数据和实现各个操作的代码，都被封装在这个黑盒子内部，在外面是看不见的，更不能从外面去访问或修改这些数据或代码。

使用对象时只需知道它向外界提供的接口形式而无需知道它的内部实现算法，不仅使得对象的使用变得非常简单、方便，而且具有很高的安全性和可靠性。可见面向对象程序设计中的对象来源于现实世界，更接近人们的思维。

2．类

在现实世界中，“类”是一组相同属性和行为的对象的抽象。例如，张三、李四、王五，虽然每个人的性格、爱好、职业、特长等各不同，但是他们的基本特征是相似的，都具有相同的生理构造，都能吃饭、说话、走路等，于是把他们统称为“人”，而具体的每一个人是人类的一个实例，也就是一个对象。

类和对象之间的关系是抽象和具体的关系。类是多个对象进行综合抽象的结果，一个对象是类的一个实例。例如“狗”是一个类，它是由千千万万个具体不同的狗抽象而来的一般概念。同理，鸡、鸭、牛、羊等都是类。

类在现实世界中并不真正存在。例如，在地球上并没有抽象的“人”，只有一个个具体的人，如张三、李四、王五。同样，世界上没有抽象的“学生”，只有一个个具体的学生。

面向对象程序设计中，“类”就是具有相同的数据和相同的操作的一组对象的集合，即类是对具有相同数据结构和相同操作的一类对象描述。例如，“学生”类可以由学号、姓名、性别、成绩等表示其属性的数据项和对这些数据的录入、修改和显示等操作组成。在C++语言中把类中的数据称为数据成员。

在面向对象中，总是先声明类，再由类生成对象。类是建立对象的“模板”，按照这个模板所建立的一个个具体的对象，就是类的实际例子，通常称为实例。比如，手工制作月饼时，先雕刻一个有凹下图案的木模，然后在木模上抹油，接着将事先揉好的面塞进木模里，用力挤压后，将木模反扣在桌上，一个漂亮的图案就会出现在月饼上了。这样一个接着一个的，就可以制造出外形一模一样的月饼。这个木模就好比是“类”，制造出来的糕点好比是“对象”。

3．消息

现实世界中的对象不是孤立存在的实体，它们之间存在着各种各样的联系，正是它们之间的相互作用、联系和连接，才构成了世间各种不同的系统。同样，在面向对象程序设计中，对象之间也需要联系，称为对象的交互。面向对象程序设计必须提供一种机制允许一个对象与另一个对象的交互。这种机制叫消息传递。即对象之间进行通信的结构叫做消息。在对象的操作中，当一个消息发送给某个对象时，消息包含接收对象去执行某种操作的信息。发送一条消息至少要包括说明接受消息的对象名、发送给该对象的消息名（即对象名、方法名）。一般还要对参数加以说明，参数可以是认识该消息的对象所知道的变量名，或者是所有对象都知道的全局变量名。

在面向对象程序设计中的消息传递实际是对现实世界中的信息传递的直接模拟。以实际生活为例，我们每一个人可以为他人服务，也可以要求他人为自己服务。当我们需要别人为自己服务时，必须告诉他们我们需要的是什么服务，也就是说，要向其他对象提出请求，其他对象接到请求后，才会提供相应的服务。

一般情况下，称发送消息的对象为发送者或请求者，接收消息的对象为接收者或目标

对象。对象中的联系只能通过消息传递来进行。接收对象只有在接收到消息时，才能被激活，被激活的对象会根据消息的要求完成相应的功能。

消息具有三个性质：

- 同一对象可接收不同形式的多个消息，产生不同的响应；
- 相同形式的消息可以送给不同对象，所做出的响应可以是截然不同的；
- 消息的发送可以不考虑具体的接收者，对象可以响应消息，也可以对消息不予理会，对消息的响应并不是必须的。

在面向对象程序设计中，消息分为两类，即公有消息和私有消息。到底哪些消息是公有消息，哪些消息是私有消息，需要有一个明确的规定。若有一批消息同属于一个对象，其中有一部分是由外界对象直接向它发送的，称为公有（public）消息；还有一部分则是它自己向本身发送的，这些消息是不对外开放的，外界不必了解它，称为私有（private）消息。

4．方法

在面向对象程序设计中，要求某一个对象做操作时，就向该对象发送一个相应的消息，当对象接收到发向它的消息时，就调用有关方法，执行相应的操作。方法就是对象所能执行的操作。方法包括界面和方法体两部分。方法的界面也就是消息的模式，它给出方法的调用协议；方法体则是实现某种操作的一系列计算步骤，也就是一段程序。消息和方法的关系是：对象根据接收到消息，调用相应的方法；反过来，有了方法，对象才能响应相应的消息。所以消息模式与方法界面应该是一致的。同时，只要方法界面保持不变，方法体的改动不会影响方法的调用。在 C++语言中方法是通过函数来实现的，称为成员函数。

1.1.3 面向对象程序设计的基本特征

面向对象程序设计方法模拟人类习惯的解题方法，代表了计算机程序设计的新颖的思维方法。这种方法的提出是对软件开发方法的一场革命，是目前解决软件开发面临困难的最有希望、最有前途的方法之一。本节介绍面向对象程序设计的 4 个基本特征。

1．抽象性

抽象是人们认识问题的最基本的手段之一。例如，常见的名词“人”，就是一种抽象。因为世界上只有具体的人，如张三、李四、王五。把所有国籍为中国人的人归纳为一类，称为“中国人”，这就是一种“抽象”。再把中国人、美国人、日本人等所有国家的人抽象为“人”。一般抽象过程中忽略了主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象是对复杂世界的简单表示，抽象强调感兴趣的信息，忽略了不重要的信息。例如，在设计一个学籍管理程序的过程中，考察某个学生对象时，只关心他的姓名、学号、成绩等信息，而对他的身高、体重等信息就可以忽略。

一般来讲，抽象是通过特定的实例（对象）抽取共同性质以后形成概念的过程。抽象是对系统简化描述或规范说明，它强调了系统中的一部分细节和特性，而忽略了其他部分。抽象包括两个方面：数据抽象和代码抽象。前者描述某类对象的属性或状况，也就是此类对象区别于彼类对象的特征物理量；后者描述了某类对象的共同行为特征或具有的共同操作。

抽象在系统分析、系统设计以及程序设计的发展中一直起着重要的作用。在面向对象

程序设计方法中，对一个具体问题抽象分析的结果，是通过类来描述和实现的。

2．封装性

从字面上可以理解，封装就是将某事物包围起来，使外界不知道其实际内容。

在程序设计中，封装是指将一个数据和与这个数据有关的操作集合放在一起，形成一个能动的实体——对象，用户不必知道对象行为的实现细节，只需根据对象提供的外部特征性接口访问对象即可。因此，从用户的观点来看，这些对象的细节就像包含在一个“黑盒子”里，是隐蔽的、看不见的。

从上面的叙述看出，封装应该具有下面几个条件：

- 具有一个清楚的边界，对象的所有私有数据、成员函数细节都被固定在这个边界内；
- 具有一个接口，这个接口描述了对象之间的相互作用、请求和响应，它就是消息；
- 对象内部的实现代码受到封装壳保护，其他对象不能修改对象所拥有的数据和代码。

面向对象系统的封装是一种信息隐蔽技术，它使系统设计员能够清楚地标明他们所提供的服务界面，用户和应用程序员则只看见对象提供的操作功能，看不到其中的数据或操作代码细节。从用户和应用程序员的角度看，对象提供了一组服务，而服务的具体实现即对象内部却被屏蔽封装着。

对象的这一封装机制，可以将对象的使用者与设计者分开，使用者不必知道对象行为实现细节，只需使用设计者提供的接口让对象去做。封装的结果实际上隐蔽了复杂性，并提供了代码重用性，从而减轻了开发一个软件系统的难度。

3．继承性

继承在现实生活中是一个很容易理解的概念。例如：我们每一个人都从我们父母继承了一些特征，如皮肤、毛发、眼睛的颜色等，我们身上的特性来自我们的父母，换句话说，父母是我们具有的属性的基础。图 1-2 说明了两个对象的相互关系，箭头的方向指向基对象。

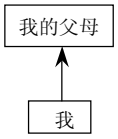

图 1-2　两个对象相互关系示意图

图 1-3 展示了交通工具的类层次。最顶部的类称为基类，是交通工具类。这个基类有汽车子类。这样，交通工具类就是汽车类的父类。可以从交通工具类派生出其他类，比如飞机类、火车类和轮船类。每个类都只有交通工具类作为其父类。汽车子类还有三个子类：小汽车类、旅行车类和卡车类，每个类都以汽车类作为父类，交通工具类可称为它们的祖先类。另外，小汽车类是轿车类、轿车类和面包车类这些派生类的父类。图中展示了小型四层次的类，它用继承来派生子类。每个类有且仅有一个父类，所有子类都是一种父类。例如，小汽车是一种汽车，轿车是一种汽车，汽车是一种交通工具，小汽车也是一种交通工具。这样，每个子类代表父类的特定版本。

继承所表达的就是一种对象类之间的相关关系。这种关系使得某类对象可以继承另外一类对象的特征和能力。

若类间具有继承关系，则它们之间应具有下列几个特征：

- 类间具有共享特征（包括数据和程序代码的共享）；
- 类间具有细微的差别或新增部分（包括非共享的程序代码和数据）；
- 类间具有层次结构。

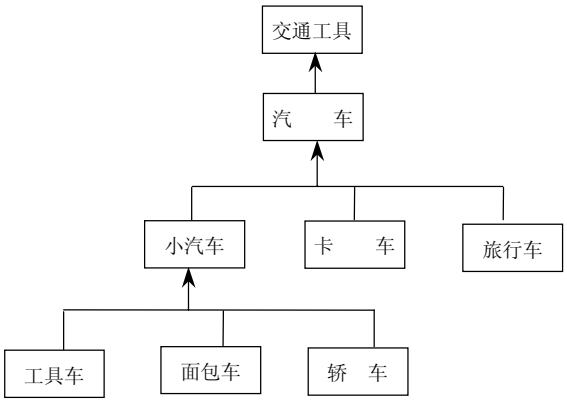

图 1-3　继承的类层次

假设有两个类 A 和 B，若类 B 继承类 A，则类 B 包含了类 A 的特征（包括数据和操作），同时也可以加入自己所特有的新特征。这时，称被继承类 A 为基类或父类或超类；而称继承类 B 为 A 的派生类或子类。同时，还可以说，类 B 是从类 A 中派生出来的。

如果有类 B 是从类 A 派生出来，而类 C 又是从类 B 派生出来的，就构成了类的层次。这样，又有了直接基类和间接基类的概念。类 A 是类 B 的直接基类，是类 C 的间接基类。类 C 不但继承它的直接基类的所有特征，还继承它的所有间接基类的特征。

面向对象程序设计可以让你声明一个新类作为另一个类的派生。派生类(也称子类)继承它父类的属性和操作。子类也声明了新的属性和新的操作，剔除了那些不适合于其用途的继承下来的操作。这样，继承可让你重用父类的代码，专注于为子类编写新代码。

那些父类已经存在，在新的应用中，你无须修改父类。所要做的就是派生子类，在子类中做一些增加与修改。这种机制，使重用成为可能。

那么，面向对象程序设计为什么要提供继承机制，也就是说，继承的作用是什么。继承的作用有两个：其一是避免公用代码的重复开发，减少代码和数据冗余；其二是通过增强一致性来减少模块间的接口和界面。如果没有继承机制，每次软件开发都要从“一无所有”开始，类的开发者在构造类时各自为政，使类与类之间没有什么联系，分别是一个个独立的实体。继承使程序不再是毫无关系的类的堆砌，而是具有良好的结构。

从继承源上分，继承分为单继承和多继承。

单继承是指每个派生类只直接继承了一个基类的特征。前面介绍的交通工具的分类，就是一个单继承的实例。图 1-4 表示了一种单继承关系。它表示 Windows 操作系统的菜单之间的继承关系。

单继承并不能解决继承中所有问题，例如，消防车即继承了灭火工具的一些特性，还继承了汽车的一些特性，如图 1-5 所示。

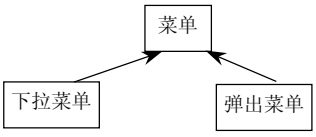

图 1-4　单继承关系图

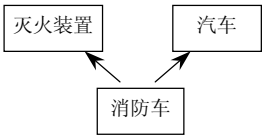

图 1-5　多继承关系图

多继承是指多个基类派生出一个派生类的继承关系。多继承的派生类直接继承了不止

一个基类的特征。

4．多态性

多态性（polymorphism）是面向对象程序设计的一个重要特征。如果一种语言只支持类而不支持多态，是不能称为面向对象语言的，只能说是基于对象的，如 Ada、VB 就属此类。C++支持多态性，在 C++程序设计中能够实现多态性。利用多态性可以设计和实现一个易于扩展的系统。

从字面理解，多态的意思是一个事物有多种形态。多态性的英文单词 polymorphism 来源于希腊词根 poly（意为“很多”）和 morph（意为“形态”）。在 C++程序设计中，多态性是指具有不同功能的函数可以用同一个函数名，这样就可以用一个函数名调用不同内容的函数。在面向对象方法中一般是这样表述多态性的：向不同的对象发送同一个消息，不同的对象在接收时会产生不同的行为（方法）。也就是说，每个对象可以用自己的方式去响应共同的消息。

在现实生活中可以看到许多多态性的例子。如学校校长向社会发布一个消息：9 月 1 日开学。不同的对象会作出不同的响应：学生要准备好课本准时到校上课；家长要筹集学费；教师要备好课；后勤部门要准备好教室、宿舍和食堂……由于事先对各种人的任务已作了规定。因此，在得到同一消息时，各种人都知道该怎么做，这就是多态性。

C++语言支持两种多态性，即编译时的多态性和运行时多态性。编译时多态性是通过重载来实现的，运行时多态性是通过虚函数来实现的。

重载是指用同一个名字命名不同的函数或操作符。函数重载是 C++对一般程序设计语言中操作符重载机制的扩充，它可使具有相同或相近含义的函数用相同的名字，只要其参数的个数、次序或类型不一样即可。例如：

```
int min(int x, int y);              /*求 2 个整数的最小数*/
int min(int x, int y, int z);       /*求 3 个整数的最小数*/
int min(int n, int a[]);            /*求 n 个整数的最小数*/
```

当用户要求增加比较 2 个字符串大小的功能时，只需增加：

```
char*min(char*，char*);
```

而原来如何使用这组函数的逻辑不需改变，min 的功能扩充很容易，也就是说维护比较容易，同时也提高了程序的可理解性，min 表示求最小值的函数。

由于虚函数的概念较为复杂，并且涉及 C++的语法细节，将在后面章节再进一步的讨论。

多态性增强了软件的灵活性和重用性，为了软件的开发与维护提供了极大的便利。尤其是采用了虚函数和动态联编机制后，允许用户以更为明确、易懂的方式去建立通用的软件。

1.2　为什么要使用面向对象程序设计

1.2.1　传统程序设计方法的局限性

当今社会是信息社会，信息社会的灵魂是作为“信息处理机”的电子计算机，从 1946

年第一台计算机 ENIAC 问世到今天的“深蓝”，电子计算机的硬件得到突飞猛进的发展，程序设计的方法也随之不断的进步。20 世纪 70 年代以前，程序设计方法主要采用流程图，结构化设计(Structure Programming，SP)也趋成熟，整个 20 世纪 80 年代 SP 是主要的程序设计方法。然而，随着信息系统的加速发展，应用程序日趋复杂化和大型化。传统的软件开发技术难以满足发展的新要求。

1．传统程序设计开发软件的生产效率低下

早期的计算机存储器容量非常小，人们设计程序时首先考虑的问题是如何减少存储器开销，硬件的限制不容许人们考虑如何组织数据与逻辑，程序本身短小，逻辑简单，也无需人们考虑程序设计方法问题。与其说程序设计是一项工作，倒不如说它是程序员的个人技艺。但是，随着大容量存储器的出现及计算机技术的广泛应用，程序编写越来越困难，程序的大小以算术基数递增，而程序的逻辑控制难度则以几何基数递增，人们不得不考虑程序设计的方法。相对于硬件的快速发展，软件的生产能力还比较低下，开发周期长、效率低、费用不断上升，以致出现了所谓的“软件危机”。

然而，尽管传统的程序设计语言经历了第一代语言、第二代语言以及第三代语言的发展过程，但是其编制程序的主要工作还是围绕着设计解题过程来进行的，故称为面向过程的程序设计，传统的程序设计语言为过程性语言。这种传统程序设计的生产方式仍是采用较原始的方式进行，程序设计基本上还是从语句一级开始。软件的生产缺乏大粒度、可重用的构件，软件的重用问题没有得到很好解决，从而导致软件生产的工程化和自动化屡屡受阻。

影响软件生产效率低的另一个原因是问题的复杂性。随着计算机技术的大规模推广，软件的应用范围越来越广，软件的规模越来越大，要解决的问题越来越复杂。传统程序设计的特点是数据与其操作分离，而且对同一数据的操作往往分散在程序的不同的地方。这样，如果一个或多个数据的结构发生了变化，那么这种变化将涉及程序的很多部分甚至遍及整个程序，致使许多函数和过程必须重写，严重时会导致整个软件结构的崩溃。这就是说，传统程序复杂性控制是一个很棘手的问题，这也是传统程序难以重用的一个重要原因。

维护是软件生命周期中的最后一个环节，也是非常重要的一个环节。传统程序设计是面向过程的，其数据和操作分离的结构，使得维护数据和处理数据的操作过程要花费大量的精力和时间，严重地影响了软件的生产效率。

总之，要提高软件生产的效率，就必须很好地解决软件的重用性、复杂性和可维护性问题。但是传统的程序设计是难以解决这些问题的。

2．传统程序设计难以应付日益庞大的信息量和多样性的信息类型

随着计算机科学与技术的飞速发展和计算机应用的普及，当代计算机的应用领域已从数值计算扩展到了人类社会的各个方面，所处理的数据已从简单数字和字符，发展为具有多种格式的多媒体数据，如文本、图形、图像、影像、声音等，描述的问题从单纯的计算机问题到仿真复杂的自然现象和社会现象。于是，计算机处理的信息量与信息类型迅速增加，程序的规模日益庞大，复杂度不断增加。这些都要求程序设计语言有更大的信息处理能力。然而，面对这些庞大的信息量和多样的信息格式，传统程序设计方法无法应付的。

3．传统的程序设计难以适应各种新环境

当前，并行处理、分布式、网络和多机系统等，已经或将是程序运行的主要方式和主

流环境。这些环境的一个共同的特点是都具有一些有独立处理能力的节点，节点之间有通讯机制，即以消息传递进行联络。显然传统的程序设计技术很难适应这些新环境。

综上所述，传统的程序设计不能够满足计算机技术的迅猛发展的需要，软件开发迫切需要一种新的程序设计范型的支持。那么，面向对象程序设计是否能担当此任，下面再分析一下面向对象程序设计的一些优点。

1.2.2　面向对象程序设计的主要优点

从面向过程程序设计到面向对象程序设计是软件开发史上的一个里程碑。面向对象程序设计主要将精力集中处理对象的设计和研究上，而改变了过去人们设计软件的思维方式，即程序设计者主要设计和研究的是数据格式和过程，这样极大地减少了软件开发的复杂性，提高了软件开发的效率。面向对象程序设计主要具有以下优点。

1．可提高程序的重用性

重用是提高软件开发效率的最主要的方法，传统程序设计的重用技术是利用标准函数库，但是标准函数库缺乏必要的“柔性”，不能适应不同场合的不同需要，库函数往往仅提高最基本的、最常用的功能，在开发一个新的软件系统时，通常大部分函数仍需要开发者自己编写，甚至绝大部分函数是新编的。

面向对象程序设计能比较好地解决软件重用的问题。对象所固有的封装性和信息隐蔽等机制，使得对象内部的实现与外界隔离，具有较强的独立性，它可以作为一个大粒度的程序构件，供同类程序直接使用。

有论证方法可以重复使用一个对象类：一种方法是建立在各种环境下都能使用的类库对象集，供相关程序直接使用；另一种方法是从它派生出一个满足当前需要的新类。继承性机制使得子类可以重用其父类的数据和程序代码，而且可以在父类代码的基础上方便地修改和扩充，这种修改并不影响对原有类的使用。由于可以像使用集成电路（IC）构建计算机硬件那样，比较方便地重用对象类来构造软件系统，因此有人把对象称为“软件 IC”。

2．可控制程序的复杂性

传统的程序设计方法忽略了数据和操作之间的内在联系，它把数据与其操作分离，于是存在使用错误的数据调用正确的程序模块或使用正确的数据调用错误程序模块的危险。使数据和操作保持一致，控制程序的复杂性，使程序员的一个沉重的负担。面向对象程序设计采用了数据抽象和信息隐蔽技术，把数据及对数据的操作放在一个类中，作为相互依存、不可分割的整体来处理。这样，在程序中任何要访问这些数据的地方都只需简单地通过传递消息和调用方法来进行，这就有效控制了程序的复杂性。

3．可改善程序的可维护性

用传统程序设计语言开发出来的软件很难维护，是长期困扰人们的一个严重问题，是软件危机的突出表现。但面向对象程序设计方法所开发的软件可维护性较好。在面向对象程序设计中，对对象的操作只能通过消息传递来实现，所以只要消息模式即对应的方法界面不变，方法体的任何修改都不会导致发送消息的程序修改，这显然对程序的维护带来了方便。另外，类的封装和信息隐蔽机制使得外界对其中的数据和程序代码的非法操作作为不可能，这也就大大地减少了程序的错误率。