面向对象程序设计(C++)

主编　马慧彬　张宗利　赵智超

東北林業大學出版社

面向对象程序设计(C++)

主　编　马慧彬　张宗利　赵智超

东北林业大学出版社

图书在版编目（CIP）数据

面向对象程序设计：C + +/马慧彬，张宗利，赵智超主编. —哈尔滨：东北林业大学出版社，2008.4
ISBN 978-7-81131-237-9

Ⅰ. 面… Ⅱ. ①马…②张…③赵… Ⅲ. 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2008）第 061356 号

责任编辑：任 俐 朱成秋
封面设计：彭 宇

NEFUP

面向对象程序设计（C + +）
Mianxiang Duixiang Chengxu Sheji（C + +）
主编 马慧彬 张宗利 赵智超

东北林业大学出版社出版发行
（哈尔滨市和兴路 26 号）
哈尔滨市工大节能印刷厂印装
开本787×960 1/16 印张16 字数281千字
2008 年 4 月第 1 版 2008 年 4 月第 1 次印刷
印数 1—1 000 册
ISBN 978-7-81131-237-9
TP·82 定价：25.00 元

编　委

主　编　马慧彬　张宗利　赵智超

副主编　张忠武　牟晓枫

前 言

自20世纪80年代以来，面向对象程序设计的思想在软件领域中得到了很大的发展和完善。近年来，高等院校计算机及相关专业均陆续开设了这门课程。目前，讲述C++的教材较多，但多数C++教材的内容与C语言教材内容重复，真正面向对象程序设计的内容较少，面向对象的思想渗透得不够。而面向对象程序设计类的教材从内容上看，比较偏重概念与理论，实例较少。本书将面向对象思想理论与C++语言程序实现相结合，以几个C++实例贯穿教材始终，使读者能够由浅入深，循序渐进地学习，通过实例而理解并掌握面向对象程序设计思想。

面向对象方法的出现，实际上是程序设计方法发展的一个返璞归真的过程。面向对象方法的本质就是强调从客观世界中固有的事物出发来构造系统，用人类在现实生活中常用的思维方式来认识、理解、描述客观事物。这种解决问题的方法使软件开发从专业化的方法、规则和技巧中回到客观世界，回到人们习惯的思维方式。

本书编写的目的是为了给程序设计初学者提供一本清晰的入门教材。本书适合作为计算机及其相关专业的本专科教材和参考书，也可供自学者使用。本书将C++作为学习面向对象程序设计的基本语言，不仅介绍了C++的语法规则，还介绍了常用的数据结构与算法。全书以面向对象的程序设计方法贯穿始终，从面向对象的理论到面向对象的实现，再到面向对象的建模，力求使读者在掌握基本程序设计方法的同时，牢固树立面向对象的编程思想，为后续课程的学习打下基础，以适应当前软件发展的需要。考虑到读者多是在学习完结构化程序设计语言（如C语言）后接着学习面向对象程序设计的，会用结构化程序设计的思想理解C++，因此，本书重点说明面向对象思想与结构化设计的区别与联系，同时突出了C++语言与C语言的不同。

多年来，编者一直从事计算机语言与面向对象课程的教学，本书的基本框架是从编者的教案中整理出来的。依据多年教学经验，编者对书中每一部分的知识点和难点都以比较精练的语言进行讲解。同时对每一个知识点都列举了必要的例题，并且对例题进行了比较深刻的分析。本书建议教学学时为40学时，上机实验学时为20学时。本书作为本专科教材时，教学重点在第

1～第5章，作为研究生教材时，教学重点在第9～第10章。

本书的第1章、第2章、第7章由马慧彬编写，第3章、第6章由赵智超编写，第4章、第9章由张忠武编写，第5章、第8章由牟晓枫编写，第10章由张宗利编写。全书由马慧彬统稿，教学课件由马慧彬与张忠武研制，所有例题及习题由马慧彬与牟晓枫上机测试。在本书的编写过程中，查阅和参考了部分文献，在此对书后所列参考文献的作者表示感谢。由于作者水平有限，书中难免有不足和错误，恳请读者批评指正。

编者

2008年4月

目 录

1 面向对象程序设计与C++的基础知识

1.1 程序设计思想的发展

自从第一台计算机问世以来，CPU的处理能力一直按摩尔定律不断提高，即每18个月翻一番，程序设计语言与程序设计方法也随之不断地发展。初期，由于计算机运算速度与存储空间的限制，使程序员主要追求时间与空间上的高效率，而把程序的可理解性、可移植性、可扩展性等因素放到第二位。随着计算机硬件与网络技术的发展，计算机应用领域与应用规模不断扩大，程序设计不再是程序员的个人技巧与个人艺术的体现，而是要考虑在多人合作基础上的程序的可靠性、可理解性、可重用性等多方面因素。较大规模的计算机软件的应用需求促进了程序设计语言与程序设计方法学的发展。

1.1.1 程序设计方法的发展

今天的计算机已不再像当初那样只用于科学计算，它几乎渗透到社会的每一个角落，面对越来越复杂的软件，如果人们不改进设计方法，人的大脑就会不堪重负，根本无法完成复杂度高的软件设计。改进软件设计的方法很多，但目标只有一个，那就是让编程工作更轻松一些，而程序功能更强大一些。经过长期软件设计的千锤百炼，有几种设计方法得到认可并被广泛使用，它们是结构化程序设计、面向对象程序设计、构件式程序设计。

结构化程序设计（SD，Structured Programming）方法的主要技术是自顶向下、逐步求精，采用单入口/单出口的控制结构。所谓“自顶向下、逐步求精”，主要强调的是两点：

- 设计过程从比较抽象的问题出发，逐步延伸到比较具体的问题，即逐步求精；
- 同一层各子任务之间应当保持相互独立，改变某个子任务，只影响所属下层任务，而不应该影响其他子任务。

这种处理方式的优点是明显的，试想如果程序各部分之间任意关联，则对某一处的修改都将导致与之关联部分的不可预见的副作用。结构化程序设计方法非常注重程序模块的独立性，每个模块有相对完整的功能，通过接口

与其他模块交换信息，而模块之间不相互干涉内部事务。例如，程序中的函数设计就是这种设计思路的典型表现：

- 函数应有比较完整的功能；
- 函数功能由函数的名字集中体现出来；
- 函数的代码长度要适宜，复杂度要适宜；
- 内部操作采用与外部无关的局部变量；
- 标准输入/输出接口是函数的参数及返回值。

面向对象程序设计（OOP，Object Oriented Programming）方法建立在结构化程序设计基础上，其中心思想是：首先明确处理对象，然后再考虑对象特性和对象处理过程。面向对象程序设计方法包含丰富的知识内容，需要经过设计实践才能深入理解，而一旦掌握了它的精髓，将会使程序设计进入一个全新的境界。关于面向对象程序设计的方法学将在书中第9章详细介绍。

构件式程序设计（CP，Component Programming）是在面向对象程序设计基础上发展形成的，是面向对象程序设计发展到一定阶段的必然产物。我们知道，机械产品与电子产品的广泛普及很大程度上要归功于产品部件的通用化和标准化，与此形成鲜明对照的是软件设计仍然要依赖个人的知识、能力、经验。如何使计算机程序也像电子产品一样可以通过零部件组装一直是普遍关注的课题，这种思想就是构件式设计方法。尽管这种技术还不够成熟，构件功能与装配特性还有待于深入研究，但构件式程序设计以软件工业化为目标，已经表现出明显的优势和巨大的发展潜力。

1.1.2 面向机器的语言

在计算机发展的早期，由于每台计算机的指令系统不同，程序员在解决问题时首先考虑的问题是——为哪台机器服务，然后才是——完成何种任务。程序员要清楚地知道这台机器的指令系统，才可能编写出为这台机器服务的程序，这无疑使程序不可能具有良好的可移植性、可理解性；同时程序代码长，程序员工作量大，也使得程序编写成为少数科技人员的技能，不可能形成较大规模的、较普及的软件产业。当然这类计算机语言是最充分地考虑到计算机执行与存储的特点，所以往往可以编写出时空效率很高的程序。这类语言最贴近机器，因此称为低级语言，代表语言有机器语言、汇编语言等。

1.1.3 面向过程的语言

随着计算机应用的不断扩展，计算机软件的需求越来越大，程序规模与

复杂性也不断增长，这促使人们不断去发展计算机语言，不断去探索新的程序设计方法，高级语言应运而生且不断发展，尤其是在结构化程序设计思想引入之后，一些适应结构化程序设计方法的语言深受欢迎，如结构化特点鲜明的Pascal语言，灵活方便、目标代码效率高的C语言等。这个阶段程序员在解决问题时首先考虑的问题是——采用什么样的数据结构，即数据的组织形式，然后就是——在此结构上采用的算法。解决问题时注重事物的结构与解决的步骤。用这类语言编写的程序在可理解性、可移植性等方面大大提高了。但是，好的程序要求程序员从一开始就要清楚事物的脉络、结构、行为特点等，对于较大的软件来说这往往很难办到，这就促使人们从全新的角度去考虑编程的方法。

1.1.4 面向对象的语言

要想适应现在的软件需求，就应该实现软件生产的工业化，像生产汽车一样，由各个零部件组装在一起。对事物的看法也从注重结构改成更加注重事物的行为特点，以便生产出适用的“零部件”，最后将它们有机地结合起来。这是面向对象的编程思想，适应这种编程思想的语言称为面向对象的语言。

面向对象的语言都提供了用于创建“零件”的方法，并称这些零件为“对象”，还提供了对象之间规范的信息传递方式、对象之间的关系结构等，这为合理地组装这些对象从而形成程序提供了可能性。用面向对象的思想解决问题时，首先考虑的问题是——包含多少对象、每个对象应如何封装，然后要考虑对象引发的事件是什么、对象之间的关系是什么。这类语言中具有代表性的有：首先提出“类”等概念的Smalltalk语言；具有多重继承、异常处理、自动内存管理等特点的纯面向对象语言Eiffel语言；特别是应用最广泛的、具有良好代码可重用性的C++语言。

1.1.5 用于编写可视化程序的语言

用于编写可视化程序的语言首先是面向对象的，这一类的语言目前被广泛使用，因为它适应了现代软件的需求，现在的用户更加强调操作上的简便性、界面的友好性、一致性等。这一类语言把GUI（Graphical User Interface图形用户接口）界面中常用的对象模具先设计好，以备程序员随时使用，这些对象模具称为“控件”。现在常用的语言如Visual C++，Delphi，Power Builder，Visual Java等都可以用来编写可视化的应用程序。程序设计过程简捷，界面友好，大大减轻了程序员的工作量。面向对象与可视化的程序设

计语言的发展大大促进了软件大工业时代的到来。

1.2 面向对象程序设计的基本概念

面向对象程序设计以类作为构造程序的基本单位，具有数据封装、数据抽象、继承、多态性等基本特点。这种思想认为客观世界由各种各样的实体，也就是对象组成，每个对象有自己的内部构造，每个对象有自己的内部状态和运动规律，不同对象间的相互联系和作用构成整个系统。这是一种模拟现实世界的模型而产生的编程方法，是对事物的功能抽象与数据抽象，并把解决问题的过程看成是一个分类演绎的过程。

1.2.1 对象（Object）

这里所说的对象是组成系统的相对独立、又相互联系的客观实体，它可能包括各种属性或可能做多种动作。在计算机中对象表现为由一组数据及在这组数据上所做的函数组成的封装体。例如，在银行出纳业务中，包括具体的出纳部门、顾客、业务办理过程、货币的单位等都可视为对象。

1.2.2 类（Class）

类描述了一组有相同特性（数据元素）和相同行为（函数）的对象的共有特征，是把那些在程序执行期间除了状态之外其他方面都一样的对象归结在一起，经过抽象而构成的。类实际上就是数据类型，例如，复数是由两个分别称为实部和虚部的属性构成的，可以进行加、减、乘、求模等运算。从广义上讲，整型等基本数据类型也可以视为类。类与基本数据类型的区别在于程序员是为了与具体问题相适应来创建类，而不是被迫使用已存在的数据类型。这些已存在的数据类型的设计动机是为了描述机器的存储单元，而程序员可以通过增添他所需要的新数据类型来扩展这个程序设计语言。从一组实体中抽象定义而来的数据类型就称为抽象数据类型。抽象数据类型是面向对象程序设计中的一个基本概念，在支持面向对象程序设计的语言工具中，抽象数据类型可以进行与基本类型一样的类型检查，一样准确地工作。而由此类型创建的变量一般就称为对象或实例。

1.2.3 面向对象程序设计的三大特性

面向对象程序设计具有三个重要特性，它们是封装性、继承性、多态性。只有很好地理解了这三个基本特性，才能真正掌握面向对象的技术。

封装性（Encapsulation）是指将一组数据和这一组数据有关的操作集合组装在一起的能力与处理机制。经过封装以后，用户不能直接随意操作对象的内部数据，只有通过标准接口、使用规范方式才能访问这些数据。数据的具体细节被封装起来，每个对象只能通过标准接口与外界通信。这样做的好处是对对象内部数据有保护的作用，同时又使得用户对实体对象的操作更加规范。

继承性（Inheritance）是面向对象程序设计工具必须解决的一个问题。两个类型可以有共同的特性和行为，但是一个类型可能比另一个类型有更多的特性，也可以处理更多的消息，因此可以让一个类型从另外一个类型中继承过来一些特性和行为，同时也可以修改、添加某些功能，这样就建立起一系列有层次的类。在面向对象的程序设计语言中，这种下级可以继承上级的某些特征的特性，由继承机制实现。继承机制提供了函数的通用性，并允许根据需要具体化。例如，用于计算机辅助设计系统或游戏中的形体问题，就是比较典型的例子。这里基本类型是“形体”，每个形体有大小、颜色、位置等，并且每个形体都具有可被绘制、擦除、移动、着色等属性，因此可以派生出特殊类型的形体，如圆、矩形、三角形等，这些派生的类型可能添加一些行为或特性，如圆有半径、矩形有长与宽等，也可能是行为互不相同，如面积的求解方法就不同。类型层次结构既体现了形体间的相似，又体现了它们的区别。

所谓多态性（Polymorphism），简单地说，就是一致的接口，不同的操作，即在程序中同一个符号或名字在不同情况下具有不同的解释。在面向对象程序设计当中，由程序员设计的多态性有两种基本形式：编译期多态与执行期多态。编译期多态是指在程序编译时就可以确定符号或名字的解释，在编译中也称为“早捆绑”，这在C++中，主要通过重载机制实现；执行期多态是指必须等到程序动态运行时才能确定符号或名字的具体解释，也称为“晚捆绑”，它是面向对象的一大特性，主要通过继承机制与虚函数实现。

1.3 C++语言的基本程序结构

1.3.1 C++的语言简介

自从1972年AT&T贝尔实验室的Brian Kernighan和Dennis Ritchie设计了C语言，并以之作为UNIX操作系统的程序设计语言后，从个人计算机到巨型计算机都广泛地支持C语言。进入20世纪80年代后，面向对象程序设

计在程序设计领域引起了普遍重视，AT&T 贝尔实验室的 Bjarne Stroustrup 集 C 语言、Smalltalk 语言等多个语言的优势开发了面向对象的程序设计语言 C++。C++不单纯是 C 语言的扩充，而是一种全新的、完备的程序设计语言，由于它有众多优点，因此现已被广泛使用。但是由于 C++语言尚未标准化，因而不同版本、不同开发环境支持的 C++功能都不尽相同。常用的 C++开发环境主要有由 Sun 公司开发的用于 UNIX 操作系统的 SPARCompiler C++3.0.1，由 Borland 公司开发的用于 DOS 操作系统的 Borland C++ 3.0，由 Microsoft 公司开发的用于 WINDOWS 操作系统的 Visual C++系列版本。本书中以微机上较常用的 Visual C++6.0 为标准进行讨论。

1.3.2 C++的基本程序结构

示例 1-1 是一个完整的 C++程序，可以看出它与 C 语言有相似之处，也有许多不同之处。下面通过分析这个简单程序，让大家学会 C++编程的基本知识。

一个程序就像一篇文章一样，具有一定的逻辑结构和组织形式，当然无论什么样的形式都要力求便于理解和阅读，这可以通过适当地加以注释加深程序理解、采用缩进的格式突出程序结构。程序中以“//”开头的直到该行行尾的内容就是注释，计算机运行时会跳过所有的注释语句，注释对程序运行效果不产生任何影响，但对程序员来说，注释可以大大提高程序的可读性与可理解性，因此适当的注释是必要的，这对提高程序质量有很大意义。

一个完整的程序由若干文件组成，当 A 文件中要使用 B 文件中的内容时，要在 A 文件中使用#include <B 文件名>编译预处理语句将 B 文件包含进去，这一点与 C 语言是一样的。示例 1-1 中的第一语句的就是由于程序中要使用 iostream.h 文件中定义的用于输入与输出的操作，因此将这个函数库包含在文件中，使得编译时能把相关内容联系在一起。

程序中定义了一个 location 类，用于描述平面上一点的位置，有两个属性——横坐标 x 与纵坐标 y，有三个行为，它们是初始化平面上一点 Init（）、读取点的横坐标 Getx（）与读取点的纵坐标 Gety（）。这三个函数中 Getx（）与 Gety（）的函数体定义在类体之内，而 Init（）定义在类体以外，定义在类体外的函数体与类体一起构成类完整的定义。

类的定义是说明性语句，真正的执行是从 main（）开始的，主函数中一般首先定义对象或普通变量，然后是函数与语句的调用。当然，我们现在还无法完全理解这个 C++程序的所有内容，但通过这个程序我们可以知道 C++大体的程序结构。

示例1-1 程序结构示例
——平面上的位置描述

```
#include <iostream.h>
    class location
      {private:
         int x, y;
       public:
         void Init (int, int); //初始化平面上一点
         int Getx( ){return x;} //返回横坐标
         int Gety( ){return y;} //返回纵坐标
      }; //定义类体
    void location::Init (int Initx, int Inity)
      {x = Initx; y = Inity;}
      void main( )
      {location a1, *pa1 = &a1; //定义对象,对象的指针
       a1.Init(5,3);
       int x = pa1 -> Getx( );
       int y = pa1 -> Gety( );
       cout<<x<<" "<<y<<endl; //输出位置坐标后换行
       cout<<a1.Getx( )<<" "<<a1.Gety( )<<endl; //输出位置坐标后换
       行
      }
```

1.3.3 C++的程序运行

我们用C++语言编写了一个程序后，使用计算机提供的编辑工具将该程序以文本形式存放在计算机中，这种形式的程序称为源程序。为了让计算机能执行源程序，还要借助于编译程序将源程序的代码转换成计算机能理解的形式，这种经过编译形成的程序称为目标程序。为了方便程序能在一些操作系统下正确运行，应将目标程序与现有的库文件通过连接程序的连接生成可执行文件，它是能脱离编辑工具，直接在操作系统下运行的程序。具体过程如图1-1所示。在不同的C++语言开发环境中，完成以上过程的具体操作有较大差异，如UNIX操作系统下的一些C++编译程序与WINDOWS下的C++程序开发环境有很大差别。

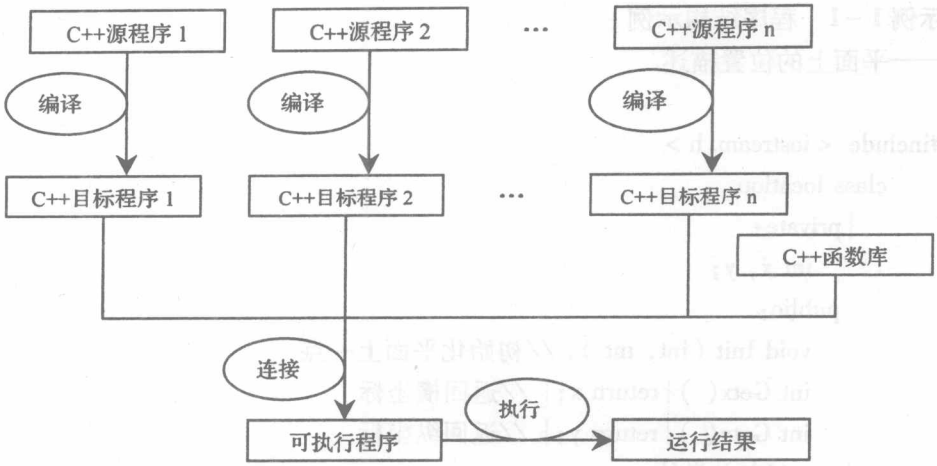

图 1-1 C++程序运行过程

1.4 面向对象程序设计过程

这里所说的面向对象的程序开发过程是指从粗设计到编译代码的最初阶段。至于大规模的软件设计将在第 9 章介绍。

一个面向对象的程序要通过多个对象相互通信，共同协作来完成某一任务，因此对象是面向对象程序设计的核心。建立、构造对象是设计程序首先要解决的问题。人们利用计算机开发程序来解决某些问题，而这些问题可能来源于现实世界，如解决学生的学籍管理问题；也可能来源于思维世界，如人工智能中的博弈程序。无论是现实世界还是思维世界都是非常庞大的，我们在编程时只关心其中的某一部分，这个被我们关心的部分称为程序的参照系统。

参照系统是由各种事物组成的，我们把这些事物中具有共同特征的事物抽象成某些概念，这个过程称为抽象。有了这些抽象出来的概念，就可以在语言的支持下创建类，所以说，类实际上是那些实体的一种模型。在 C++中可以通过类定义代表具体实体的对象。参照系统与程序系统之间的对应关系如图 1-2 所示。

面向对象的设计就是指通过建立一些类以及它们之间的关系来解决问题。我们要根据对象间的关系，建立类的体系，明确它们之间是构成关系还是类属关系，从而确定类之间是包含、引用还是继承。面向对象程序设计的