普通高等院校计算机类专业精品教材

C++面向对象程序设计

王桃发 ✦ 编著

華中科技大學出版社
http://www.hustp.com

普通高等院校计算机类专业精品教材

C++面向对象程序设计

王桃发　编著

华中科技大学出版社
中国·武汉

内 容 简 介

本书共分 9 章。第 1 章主要介绍 C++在非面向对象方面的扩展。第 2 章、第 3 章、第 4 章主要介绍代码重用技术——继承，内容涉及类本身管理函数、类聚合与包含、对象内存模型、子类与基类之间的特殊关系、派生类对象构建和析构过程、对象兼容性和多继承、多态性概念、虚函数、纯虚函数、抽象类、支持虚拟机制类对象内存模型等内容。第 5 章介绍函数模板和类模板，内容主要涉及模板重载、特化、部分特化、非类型化参数等技术细节。第 6 章主要介绍运算符重载，即重载为全局运算符或类成员函数或类友元函数，并列举常用运算符重载例子。第 7 章介绍一种重要的编程技术——标准模板库，内容主要涉及迭代器、容器、适配器、函数对象、算法、存储分配器等内容。第 8 章主要介绍异常处理，内容主要涉及 C++异常处理原则、C++异常类型匹配规则、C++异常处理堆栈解退，同时还介绍 RTTI(运行时类型标识)对异常处理的支持。第 9 章介绍 C++对 I/O 的支持，内容主要涉及流、数据格式、预定义流对象、格式控制、文件操作和内存格式化。

本书既可以作为本科生面向对象程序设计的教材，也是程序员理想的参考书。

图书在版编目(CIP)数据

C++面向对象程序设计/王桃发　编著. —武汉：华中科技大学出版社，2012.12
ISBN 978-7-5609-8549-7

Ⅰ.C…　Ⅱ.王…　Ⅲ.语言-程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2012)第 290700 号

C++面向对象程序设计　　王桃发　编著

策划编辑：谢燕群
责任编辑：陈元玉
封面设计：范翠璇
责任校对：朱　霞
责任监印：周治超
出版发行：华中科技大学出版社(中国·武汉)
　　武昌喻家山　　邮编：430074　　电话：(027)81321915
录　　排：华中科技大学惠友文印中心
印　　刷：湖北恒泰印务有限公司
开　　本：787mm×1092mm　1/16
印　　张：16.75
字　　数：406 千字
版　　次：2012 年 12 月第 1 版第 1 次印刷
定　　价：33.80 元

华中出版

本书若有印装质量问题，请向出版社营销中心调换
全国免费服务热线：400-6679-118　竭诚为您服务

前　言

本书以 ISO/IEC 18842—2003 作为参考标准，并结合作者多年从事 C++理论教学、实践教学工作的经历编写而成。本书假设读者已经有了 C 语言基础，因此从一开始就介绍类。

从面向过程到面向对象，这是编程发展史中的一次大飞跃，面向对象编程早已成为编程中的主流方法。支持面向对象编程的语言很多，本书采用 C++语言作为实现工具。

在软件行业流行这样一句话："编程才是硬道理"。对编程应用而言，在适当了解编程思想之后，适当编写一些代码(甚至是针对具体应用的小规模代码)是十分必要的。基于这样的认识，本书围绕面向对象程序设计的 4 个特点(数据隐藏、函数封装、继承和多态性)展开介绍，本书编写的主要特点如下。

(1) 尽量避免涉及复杂的数据结构和算法等问题，以免影响正常的语法知识学习。在实际应用中处处都会涉及数据结构和算法等问题，但是，如果在初学编程时就纠缠于算法和算法效率之类的问题，势必影响对语言本身语法的理解和应用。因此，本书尽可能地使用一个应用背景相同的例子去展开不同的知识点，避免因需要理解不同的应用背景带来的额外负担；还使用 UML 图显示类之间的关系，避免阅读长代码带来的疲劳和可能的理解偏差。

(2) 以例子带动读者领悟、体会语法规则。一个简明又有说服力的例子对读者知识的学习和理解将会帮助很大。第 2 章使用一个显示实时时钟功能的例子来说明如何封装类、如何从面向过程过渡到面向对象。第 3 章以小型公司信息管理系统设计为例说明继承和虚拟基类如何应用于实际环境中。

(3) 以对比带动读者加深对易混淆知识的理解。对比有助于学习，也有助于纠正错误的思维和理念。本书对一些容易混淆的内容或以例子或以列举、列表的方式加以说明。如 C 与 C++在动态申请空间方面的差别、浅拷贝和深拷贝问题、对象兼容性问题、编译器选择模板策略问题、容器对象与容器元素对象差别问题、C 与 C++在异常处理方面的差别和异常退栈问题。

(4) 以适当的调试加深对程序底层实现的理解。当程序运行不正常(出错或不稳定)时，利用调试功能可以有效地找出问题并加以解决。在学习阶段，利用调试功能可以跟踪变量，也可以观测运行时各个对象在内存中的位置和变化情况，跟踪虚函数的调用过程，观察虚表中的各个值(虚函数跳转地址)，跟踪异常退栈过程。因此，在不同章节中列举了一些跟踪结果(具体的值随计算机的不同可能稍有差别)，以利于学习和分析。

(5) 分析程序内存模型，强调编写思路清晰、使用健壮的代码。分析程序内存模型对学习和今后从事软件工作都很有帮助，只有深入了解程序内存模型之后，才能做到"手中无键盘，心中有程序"，从而编写出健壮、高效的代码，避免内存泄漏、思路含糊不清。本书第 2 章分析了简单对象的内存模型，第 3 章分析了派生类对象的内存模型结构，第 4 章跟踪、分析了支持虚拟机制类对象的内存模型，并对它们做了比较分析，指出其不同点和共同点。

(6) 实用性。本书的实用性体现在两方面。一方面，注重实用技巧、技术，如强调多使用 const 关键字、const 成员函数、异常处理、深拷贝(尽量不使用“免费”的浅拷贝)；强调避免全局变量带来的耦合问题，建议尽量使用类静态成员变量；强调多使用 STL 技术。另一方面，一些例子、练习来源于实际应用背景，具有实用的价值。

本书始终贯穿面向对象程序设计理念，尽量避免涉及数据结构和算法方面的内容，力求简明、准确，并注重实用性。

编　者

2012 年 12 月

目　录

第1章　面向对象程序设计与C++

摘要　本章主要介绍面向对象程序设计的基本概念，如类、对象；然后介绍支持面向对象程序设计的编程语言；最后重点介绍C++语言。本章对流行的C++语言IDE编程工具VC 6.0作了介绍，以便让读者掌握一种简单的工具。本章还介绍C++语言在非面向对象程序设计方面的扩展，如函数重载、引用、名称空间、内联函数、内存动态管理运算符new和delete。

1.1　面向对象程序设计简介

1.1.1　从面向过程到面向对象

面向过程是一种曾经很流行的设计模式，该方法的核心是以算法为主线，把数据和过程作为相互独立的部分，数据代表问题空间中的客体，程序则用于处理这些数据。它编写代码的思路类似于CPU内指令的执行过程。使用这种方式编写程序的特点是，程序由过程定义和过程调用组成。面向过程的程序可以用以下公式来表示：

程序 = 过程 + 调用

使用面向过程的方式开发代码具有易理解、易上手的优点，但随着代码量的增加和开发周期的加长，其缺点渐渐暴露出来，如下。

(1) 忽略数据和操作之间的内在联系。

(2) 类型检查功能比较弱，编译器在编译阶段检查不出有些潜在错误，稳定性不好。

(3) 使用错误的数据也能调用正确的程序模块。

(4) 忽略数据和操作之间的内在联系。

(5) 调试困难，不同于团队式开发。

(6) 可重用性差，不利于版本升级。

20世纪90年代以来，面向对象程序设计(Object Oriented Programming，OOP)渐渐流行起来，并且成为今天程序设计的主流模式。

面向对象程序设计以对象为中心。对象是由数据和容许的操作组成的封装体，与客观实体有直接的对应关系。程序通过对象去调用函数，亦称为给对象发消息。这里，消息包含操作名和消息参数。

面向对象程序设计可以用以下公式来表示：

程序 = 对象 + 消息

面向对象程序设计模式克服了面向过程程序设计模式的缺点，具有与人类习惯的思维方法一致、稳定性好、调试容易、易扩展、支持团队开发、升级简单、可维护性好、支持大规模代码开发等一系列优点。

1.1.2 面向对象程序设计基本概念

面向对象程序设计模式以对象为中心，而对象由调用类的构造函数来创建。下面先简介这些基本概念，后面章节再深入讨论它们。

1. 对象

对象可以用来映射客观世界中的任何实体，也就是说，应用领域中任何有意义、与所要解决问题有关的任何事物都可以作为对象，它既可以是具体的物理实体的抽象，也可以是人为的概念，或者是任何有明确边界和意义的东西。例如，一个人、一个控制系统、一家公司、一个窗口、一项投资业务等，都可以作为对象。总之，对象是对问题域中某个实体的抽象，设立某个对象就反映了软件系统保存了有关它的信息并具有与它进行交换的能力。

面向对象程序设计方法中涉及的对象是系统中用来描述客观事物的一个实体，是构成系统的一个基本单位，它由一组表示其静态特征的**属性**和它可以执行的一组**操作**组成。例如，一辆汽车是一个对象，它包含了汽车的属性(如型号、载重量、外形尺寸、颜色)及其操作(如启动、加速、刹车、左转弯、右转弯、空挡等)；一个窗口是一个对象，它包含了窗口的属性(如窗口风格、位置、大小、颜色等)及其操作(如打开、关闭、最大化、最小化、拖动等)。

客观世界中的实体通常既具有静态的属性，又具有动态的行为。因此，面向对象方法中的对象是由描述该对象属性的数据及可以施加于这些数据的所有操作封装在一起的统一体。对象可以执行的操作表示它的动态行为，即所谓的方法或服务。而属性是对象所包含的信息，它在设计时确定，一般只能通过执行对象的操作来改变属性。

操作描述了对象执行的功能，若通过消息传递，则操作还可以为其他对象使用。操作过程对外是封闭的，即用户只能看到这一操作实施后的结果。这相当于事先已经设计好的各种过程，只需要调用就可以了，用户不必关心这些方法是如何编写的。

对象具有如下一些基本特点。

(1) 标志唯一：指对象是可区分的，并且由对象的内在本质来区分，而不是通过描述来区分。从内存上去观察，对象就是占用一段内存的一个数据结构，但它不同于普通的数据结构，它与构造它的类存在着特殊关系——可调用封装于该类的函数，即成员函数。

(2) 分类性：可以将具有相同属性和操作的对象抽象成类。

(3) 多态性：不同的对象对同一消息做出不同的响应行为。

(4) 封装性：从对象外面只能看到对象的外部特性，对象的内部(即内部状态和处理能力的行为)对外是不可见的。

(5) 模块独立性：对象是面向对象软件的基本模块，对象以数据为中心，操作围绕其数据所需的处理来设置，没有无关的操作。

2. 类

将属性、操作相似的对象归为类，也就是说，类是具有共同属性、共同方法的对象的集合。所以，类是对象的抽象，它描述了属于该对象类型的所有对象的性质，一个对象对应类的一个实例。

当使用“对象”这个术语时，既可以指一个具体的对象，也可以泛指一般的对象，但是，当使用“实例”这个术语时，它必然指一个具体的对象。

类是面向对象程序设计语言的一个最基本元素，是一个最基本的封装单位。它隐藏了相关的数据，同时它封装了一系列函数，这些函数实现对隐藏数据的操作(读或写)。

类是用户自定义的新型数据类型，它完全不同于内置的数据类型(如 int、double 等)。

C++中类定义的语法规则如下：

```
class 类名{
    访问限制:
        返回值类型 函数名 1(参数列表){//函数体}
        返回值类型 函数名 2(参数列表){//函数体}
        返回值类型 函数名 3(参数列表){//函数体}
        ……
    访问限制:
        返回值类型 函数名 4(参数列表){//函数体}
        返回值类型 函数名 5(参数列表){//函数体}
        返回值类型 函数名 6(参数列表){//函数体}
         ……
    访问限制:
        数据类型 变量 1;
        数据类型 变量 2;
    访问限制:
        数据类型 变量 3;
        数据类型 变量 4;
        ……
};
```

说明：

(1) “class”是关键字，表示后续的语法单位是一个类。

(2) 取类名的语法规则与取变量名的语法规则相同。通常，类名的取法应该能反映出这个类的功能。

(3) 访问限制方式有三种，由三个关键字来体现，它们是 public、protected 和 private。

(4) 对 public 访问限制：允许从类外边进行访问，比如，允许“类名:: 成员函数(实参列表)”、“类对象.变量”、“类对象指针->变量” 这样的访问。

(5) 对 private 访问限制：不允许从类外边进行访问，比如，不允许“类名::成员函数(实参列表)”、“类对象.变量”、“类对象指针->变量” 这样的访问。当然，通过 public 成员函数还是可以访问到这些 private 成员的。

(6) 对 protected 访问限制：不允许从类外边进行访问。这种成员主要用于继承。private 和 protected 访问限制的成员，它们被继承后在子类中的可见性是不同的。

(7) 类定义体位于“{}”之间，并且以“;”作为类定义结束标志。

(8) 类的成员函数也仅在类体内声明，在类体外实现，但为了与全局函数相区别，成员函数名前需要加上作用域限定“**类名::**”，比如：

```
void Circle::setRadius(double radius){
            this->radius=radius
  };
```

例 1.1 类定义的应用。

以下是数学中对平面上圆(Circle)的一般描述。

平面上的圆有中心位置坐标及半径属性，对平面上的圆，用户希望重新设置它的半径或获知它的半径、面积，那么，按 C++的语法规则，可以这样设计一个类：

```
class Circle{ //一般类名的首字母大写
   public:
   //类 Circle 的构造函数
   Circle(double x,double y,double radius){
      this->x=x;
      this->y=y;
      this->radius=radius;
   }
   void setRadius(double radius){
      this->radius=radius;
   }
   //类成员函数: 首单词的首字母小写，以后的单词的首字母均大写
   double getArea(){
      return 3.14159*radius*radius;
   }
   private:
      double x;
      double y;
      double radius;
};
```

在以上程序中，Circle 仅仅是一个说明，程序运行时它不占内存。在该 Circle 类中，数据成员变量 x、y、radius 的访问限制为 private，无法从外边读取或修改它们，所以它们被“隐藏”于 Circle 类中。Circle 类中的成员函数 setRadius()和 getArea()的访问限制为 public，因此可以从外边调用它们，但是它们的可见范围为类 Circle，因此它们被“封装”在类 Circle 中。

另外，请注意这里的两个语法特点。

(1) 函数 Circle()与类同名、无返回值，是类的构造函数。

(2) this 是一个特殊的指针变量，它不是由程序员定义的，而是由编译器在类的成员函数中悄悄定义的，专门用于存放调用类成员函数的**当前对象**的地址。此处使用 this 是为了区别同名的对象属性、函数参数，如“this->radius”是指 Circle 当前对象中的 radius，而不是形参 radius。

可见，类 Circle 把数据和数据操作(函数)紧紧地联系在一起，因此可以这样去创建一个对象：

```
Circle c(1.1,2.2,3.3);
```

然后通过对象 c 去获取半径为 3.3 的圆的面积：

```
c.getArea();   // 返回 34.211915
```

3. 类与对象的关系

按照认识事物的规律，首先看到的是一个又一个具体、生动的事物(实体)。经过观察，发现这样的实体有一些共同的东西(属性)。这是一个抽象的过程，也就是认识论中所叙述的从特殊到一般的认识过程。有了这样的抽象，类的建立就显得十分自然。然而，应用中的不同事件是相互区别的，因而必须用类去构造一个一个不同的实体，这就是认识论中的从一般到特殊的过程。

当编写面向对象程序语言的代码时，首先需要设计好一个或者多个类。类只是一个类型说明，把属性和方法封装在一起。虽然程序无法直接使用类，但可以用类去实例化出一个又一个的实体，即所谓的对象。

1) 成员函数与消息

若把一个函数的可见性限制在一个类中，则该函数就是该类的成员函数(member function)。

在面向对象程序设计规范中，允许让对象去调用它所在类中的某一个成员函数，即所谓的向该对象发送“消息”。

2) 类的其他关系

在面向对象程序设计中，代码重用是一个最基本的特征，因此类之间存在继承的关系。如图 1.1 所示，类 Circle 和类 Point 之间存在继承关系。类之间还存在包含和聚合的关系。如图 1.2 所示，类 Person 的属性中包含有类 Key 的对象指针，因此这两个类之间存在聚合关系。

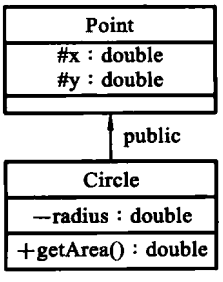

图 1.1　类继承 UML 图

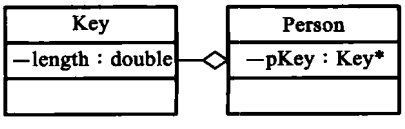

图 1.2　类聚合 UML 图

统一调用界面也是面向对象程序设计的基本特征。所谓统一调用界面，简单来说就是指表面上调用的成员函数是同一个名字，但运行时调用真正代码可以不相同。这就是通常所说的多态性。

这里先介绍简单的概念，后续再讨论较复杂的概念。另外，当类的成员变量、成员函数比较多时阅读代码比较困难，特别是当类之间的关系较复杂时阅读代码难度更大。因此通常使用 UML 图来描述类和类之间的关系，如图 1.1 和图 1.2 均为 UML 图，这方面的介绍可以参见附录 B。为了方便编辑，本书以“↑”替代 UML 图画法规定中的↑。

1.1.3 几种典型的面向对象程序设计语言

虽然程序是为人类服务的，但代码最终只能在实实在在的计算机上运行。可运行的机器码本身不存在面向过程和面向对象的本质差别(可能有效率方面的差别)。面向对象程序设计是一种思想(理念)，而这种思想是无法编写出可以在计算机上运行的代码的，因此历史上曾出现一些支持这种编程思想的编程语言，下面进行简单介绍。

1. Smalltalk 语言

Smalltalk 是公认的第一种纯正的面向对象程序设计语言。这里的“纯正”是指它的一切元素都是对象(如数字、符号、串等)，类也是对象(元素的对象)。

目前 Smalltalk 语言并不流行，当前最新的版本是 Smalltalk-80。

2. Simula 语言

1968 年，荷兰学者 E.W.Dijkstra 提出了程序设计中常用的 goto 语句的三大危害，此举引发了软件界长达数年的论战，并由此产生结构化程序设计方法，同时诞生了基于这一设计方法的程序设计语言——Pascal。但是 20 世纪 70 年代末期，这种设计方法已无法满足用户的需求。

Simula 67 是 20 世纪 60 年代后期重要的面向对象程序设计语言，在其中引入了几个面向对象程序设计最重要的概念和特性，如数据抽象概念、类机构和继承机制。

3. C++语言

C++是以 C 语言为基础发展起来的语言。C 是面向过程的语言，而 C++是面向对象的语言。C++并不是纯面向对象程序设计语言(如它允许全局的元素、全局变量和全局函数，不强制面向对象程序设计)，它并没有强制使用面向对象方法。

C++的发明者 Bjarne Stroustrup 这样评价 C++:C++的优点自始至终都是这么几条，灵活、高效，而且并非专有语言。现在 ISO C++标准的出现，巩固了最后一点。

4. Java 语言

SUN 公司于 20 世纪 90 年代在 C ++语言的基础上开发了另一种面向对象程序设计语言——Java。Java 语言具有面向对象、一次编译到处运行等优点，并提供丰富的类库，支持可视化编程。

5. C#语言

C#是从 C/C++语言发展而来的语言，Microsoft 公司于 2000 年 6 月 26 日对外正式发布 C#语言。C#语言有与 Java 语言抗衡、争夺市场的意味。使用 C#语言编写程序就可以利用.NET 框架内带的各种优点来完成各种高级的功能，如可视化编程、数据库编程、网络服务程序编程等。

1.2 C++语言

1.2.1 C++语言的演变过程

C++的发明者 Bjarne Stroustrup 在他的《The Design and Evolution of C++》一书中阐述了 C++语言的演变过程。

从名字上来看，C++与 C 有着千丝万缕的联系。

C 语言最初作为 UMX 操作系统的描述语言，是在 B 语言的基础上于 1972 年在美国贝尔实验室发明的。它具有语法简单、功能强大、性能好、像使用汇编语言编写程序一样运行效率高等一系列优点。

早期，C++被称为“带类的 C”，1983 年正式取名为 C++。1985 年，Bjarne Stroustrup 编写《C++程序设计语言》一书，标志着 C++ 1.0 的诞生。此后贝尔实验室又推出 C++ 2.0 版本、C++ 3.0 版本和 C++ 4.0 版本。

随着 C++的流行，C++的标准化工作也显得十分重要，主要经历了以下几个阶段。

(1) 1989 年，标准化准备阶段。

(2) 1994 年，美国 ANSI 的 C++标准草案出台。

(3) 1998 年，确定 14882:1998 标准(由 ISO/IEC 共同制定，ISO 为 International Standard Organization 每个英文单词第一个字母的缩写，IEC 为 International Electrotechnical Commission 每个英文单词第一个字母的缩写)。

(4) 2003 年，确定 14882：2003 标准。