普通高等教育“十一五”国家级规划教材 计算机系列教材

C++程序设计

——基于软件设计思想和案例

徐洪智 张彬连 编著

清华大学出版社

计算机系列教材

徐洪智 张彬连 编著

C++程序设计
——基于软件设计思想和案例

清华大学出版社
北 京

内 容 简 介

本书以类和对象为基础，介绍了C++中的封装、继承、多态、模板、I/O和异常处理机制，基本涵盖了C++面向对象编程的全部技术特征；分析了对象在内存中的布局，使读者更加了解C++的本质；书中运用了大量简短的例题，并进行了简要的分析，书中渗透了软件工程的思想，例题的解答都经过上机调试并配有适当的注释，便于读者学习。最后设计了4个小型综合案例，以进一步提升读者在C++面向对象程序设计方面的水平。

本书不但适合作为高等院校C++程序设计的教学用书，也适合具有一定C语言基础的读者自学C++编程。

图书在版编目(CIP)数据

C++程序设计：基于软件设计思想和案例/徐洪智，张彬连编著. —北京：清华大学出版社，2014
计算机系列教材
ISBN 978-7-302-35932-6

Ⅰ. ①C…　Ⅱ. ①徐… ②张…　Ⅲ. ①C程序—程序设计—高等学校—教材　Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2014)第061841号

责任编辑：白立军
封面设计：常雪影
责任校对：李建庄
责任印制：刘海龙

出版发行：清华大学出版社
　　网　　址：http://www.tup.com.cn，http://www.wqbook.com
　　地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
　　社 总 机：010-62770175　　**邮　　购**：010-62786544
　　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　　质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
　　课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：18.5　　**字　　数**：459千字
版　　次：2014年4月第1版　　**印　　次**：2014年4月第1次印刷
印　　数：1～2000
定　　价：34.50元

产品编号：057478-01

十余年来，C++ 已经成为最流行、应用范围最广的编程语言之一，被广泛应用于工程技术的不同领域，如控制系统、通信系统、计算机辅助设计和嵌入式系统等。同时，面向对象 C++ 程序设计也成为计算机科学与技术、软件工程等相关专业的基础课程之一，其主要特征是抽象、封装、继承和多态。由于引入这些特征，面向对象程序比传统的结构化程序具有更高的可重用性、易扩充性和易维护性，面向对象开发方法已成为开发大、中型软件的主流方法。

C++ 是一种通用的程序设计语言，可支持底层程序设计、结构化程序设计和面向对象程序设计。因此，编写 C++ 书籍的作者经常会面临两难的选择：是用结构化＋面向对象，还是纯粹面向对象。如果选择第一种方式，书中将有大量篇幅介绍 C 语言中的知识；如果选择第二种方式，书中将主要介绍 C++ 与 C 不同的部分。本书作者考虑到大多数高校在开设 C++ 课程时，学生已经完成了 C 语言的学习，所以采用了第二种方式，书中基本没有包含 C 程序设计课程中的内容。

本书作者向计算机专业和软件工程专业的学生教授 C++ 程序设计已有 12 年之久，在教学中遇到了很多问题，也积累了不少经验。作者在多年之前曾萌生编写一本教材的念头，试图通过一些简单的案例来探究 C++ 的本质内容，又担心自己对 C++ 把握不够，所以进展一直很慢，直到今天才完成书稿。

本书的主要特色如下。

（1）较全面地介绍 C++ 的基本语法及相关知识，渗透面向对象程序设计思想于问题的解答之中。

（2）分析了对象在内存中的布局，直接揭示 C++ 程序的本质。

（3）例题丰富，每道题均配有简要的分析且在 CodeBlocks 中进行了调试。

（4）控制整本书的字数，便于读者自学，更符合大学课堂内教学的实际需要。

由于编者水平有限，书中难免存在错漏之处，敬请读者批评指正，以便我们及时修改。

编　者

2014 年 2 月

第 1 章　C++ 程序设计简述

本章学习目标：

- 了解结构化程序设计和面向对象程序设计的基本思想；
- 掌握面向对象程序设计的基本概念；
- 初步掌握 C++ 中的头文件、名字空间、new 和 delete、输入输出。

C++ 语言是一种优秀的面向对象程序设计语言，在 C 语言的基础上发展而来，它以其独特的语言机制在计算机科学的各个领域中得到了广泛的应用。面向对象的设计方法是在结构化程序设计方法基础上的一次质的飞跃，C++ 完美地体现了面向对象的各种特性。

1.1　面向对象方法及面向对象语言

面向对象的方法是以对象作为最基本元素的一种分析问题和解决问题的方法。结构化方法处理问题以过程为中心，面向对象方法以“对象”为中心。对象是由数据和允许的操作组成的封装体，与客观实体有直接对应关系，一个对象类定义了具有相似性质的一组对象。面向对象的方法能更自然、更直接地反映现实世界的问题空间，能更好地适应复杂大系统不断发展与变化的要求。

面向对象语言可以分为两大类：纯粹的面向对象语言和混合型的面向对象语言。纯粹的面向对象语言中，几乎所有的语言成分都是“对象”，如 Java、Smalltalk、Eiffel 等，这类语言强调开发方法的研究和开发快速原型的能力。混合型的面向对象语言是在传统的过程化语言中加入了各种面向对象的语言机制，如 C++ 、Objective-C 等，它所强调的是运行效率和使传统程序员容易接受面向对象的思想。

1.2　结构化程序设计与面向对象程序设计

1. 结构化程序设计

结构化程序设计的主要观点是采用自顶向下、逐步求精及模块化的程序设计方法；使用 3 种基本控制结构构造程序，任何程序都可由顺序、选择、循环 3 种基本控制结构构造，设计中主要强调的是程序的易读性。“自顶而下，逐步求精”设计思想的出发点是从问题的总体目标开始，抽象低层的细节，先专心构造高层的结构，然后再一层一层地分解和细化。这使设计者能把握主题，高屋建瓴，避免一开始就陷入复杂的细节中，使复杂的设计过程变得简单明了，过程的结果也容易做到正确可靠。

结构化程序设计提倡“独立功能，单出口、单入口”的模块结构，减少模块的相互联系

使模块可以作为插件或积木使用，降低程序的复杂性，提高可靠性。编写程序时，所有模块的功能通过相应的子程序（函数或过程）的代码来实现。程序的主体是子程序层次库，它与功能模块的抽象层次相对应，编码原则使得程序流程简洁、清晰，增强了程序的可读性。

由于模块相互独立，因此在设计其中一个模块时，不会受其他模块的牵连，可将原来较复杂的问题化简为一系列简单的模块进行设计。模块的独立性还为扩充已有的系统、建立新系统带来方便，因为人们可以充分利用现有的模块进行积木式的扩展。但结构化程序设计也存在一些不足之处，在设计中数据与处理过程是分离的，在编程时需要时刻考虑数据的格式，数据和程序要始终相容，由于数据和相关处理分离，导致软件的可重用性变差。

2. 面向对象程序设计

面向对象程序设计（Object Oriented Programming，OOP）是一种计算机编程架构。OOP 的一条基本原则是计算机程序是由单个能够起到子程序作用的单元或对象组合而成。OOP 达到了软件工程的 3 个主要目标：重用性、灵活性和扩展性。为了实现整体运算，每个对象都能够接收信息、处理数据和向其他对象发送信息。面向对象程序设计的主要概念包括对象、类、方法。

1）对象（Object）

对象是系统用来描述客观事物的一个实体，它是构成系统的一个基本单位。对象是由属性和服务构成的。属性用来表示对象的状态，服务用来描述对象的行为操作。如一个圆 Circle 对象，在某一应用中其属性可能有半径、圆心位置和颜色等，对圆的操作有设置半径、设置圆心位置、设置颜色、读取半径、读取圆心位置和读取当前颜色等。对象具有如下特性：具有唯一标识名，每个对象都有自身唯一的标识，通过标识可找到相应的对象，对象的整个生命期中，它的标识都不改变，不同的对象不能有相同的标识，如在一个平面上可能有圆 c1 和圆 c2，这是两个不同的圆；具有一个状态，一个对象用数据值来描述它的状态，如圆 c1 和圆 c2 都有各自的半径、圆心位置和颜色等；具有一组服务，这组服务用于操作对象并改变对象的状态；对象的成员仍可以是对象，如圆心可以是一个点 Point 对象；模块独立性，即外部只需了解对象具有的功能，功能的实现细节隐藏在模块的内部；动态连接性，即对象之间存在相互关系和相互作用；易维护性，模块的独立性决定了它的易维护性。

2）消息（Message）

对象之间进行通信的结构称为消息。消息具有 3 个性质：一个对象可以接收不同的消息而产生不同的反应；相同的消息可以发给不同的对象，这些对象做出的反应可以截然不同；消息的发送可以不考虑具体的接收者。在 C++ 中，对象之间的相互通信通常表现为函数调用，当把消息发送给某个对象时，消息包含接收对象去执行某个函数的信息。发送一条消息至少要包括接收消息的对象名、发送给该对象的消息名，即对象名和方法名。当发送消息时，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局部变量名，或者是所有对象都知道的全局变量名。

3）类(Class)

类是对象的抽象及描述，是具有共同属性和操作的多个对象的相似特性的统一描述体。每个类要有一个名字标识，用以表示一组对象的共同特征。类的每个对象都是该类的实例。类提供了完整的解决特定问题的能力，它描述了数据结构(对象属性)、算法(服务、方法)和外部接口(消息协议)。人们可以把类理解为一种类型，如整型、实型、字符型是基本数据类型，如“int a=3;”中，a就是整数类型的一个实例，其值为3。如有屏幕上一个点，则可能用一个点类型表示，如“Point p(7,8);”中，p就是点Point类型的一个实例，其值由两部分构成，x坐标为7，y坐标为8。

4）方法(Method)

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。界面就是消息的模式，它给出了方法的调用协议；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：对象根据接收到的消息，调用相应的方法；有了方法，对象才能响应相应的消息。只要方法界面保持不变，改动方法体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。如上述Point类型，在某些场合中可能会有一个成员函数void SetColor(Color r)用来设置点的颜色或者有一些其他的成员函数来对某个(某些)点进行操作。

面向对象系统的基本特性有抽象性、封装性、继承性和多态性。

1）抽象性(Abstract)

面向对象鼓励程序员以抽象的观点看待程序，即程序是由一组对象组成的。人们认识事物有两种方法：从特殊到一般的归纳法和从一般到特殊的演绎法。将一组对象的共同特征抽象出来，从而形成“类”的概念。对于一个具体的类，它有许多具体的个体，人们称这些个体为“对象”。如圆Circle类，它可以有对象c1、c2、c3等，c1、c2、c3都是圆。

2）封装性(Encapsulation)

数据封装是指一组数据和与这组数据有关的操作集合组装在一起，形成一个能动的实体即对象。封装给数据提供了与外界联系的标准接口，只有通过这些接口，使用规范的方式，才能访问这些数据。数据封装使程序员设计程序时可以专注于自己的对象，同时切断了不同模块间数据的非法使用，减少了出错的可能性。如要设置圆的半径，只能通过Circle类给外界提供的SetRadius(int r)标准接口进行操作。

3）继承性(Inheritance)

继承性是子类自动共享父类数据结构和方法的机制，这是类之间的一种关系。在定义和实现一个类的时候，可以在一个已经存在的类的基础之上进行，把这个已经存在的类所定义的内容作为自己的内容，并加入若干新的内容。新对象通过继承能够获得父类对象原有的特点和功能。在程序设计中，继承一方面可以减少代码冗余，另一方面可以通过协调性来减少相互之间的接口和界面，有利于系统的维护。

4）多态性(Polymorphism)

多态性是指不同的对象接收到相同的消息时产生多种完全不同的行为的现象。C++支持两种多态性：编译时的多态性和运行时的多态性。编译时的多态性通过重载函数实现，而运行时的多态性通过继承和虚函数实现。

1.3 C++ 的特点

C++ 语言是一种优秀的面向对象程序设计语言，支持面向对象的程序设计方法。面向对象的设计思想是在原来结构化程序设计方法基础上的一个质的飞跃，特别适合于中型和大型的软件开发项目。C++ 完美地体现了面向对象的各种特性，从开发时间、费用到软件的重用性、可扩充性、可维护性和可靠性等方面，C++ 均具有很大的优越性。同时，C++ 从 C 语言的基础上发展而来，是 C 语言的一个超集，这就使得许多 C 代码不经修改就可被 C++ 编译通过。C++ 具有如下一些特点。

(1) C++ 从 C 中继承了过程编程方式的高效性，并集成了面向对象编程方式的功能。

(2) 编程功能强大，程序结构清晰、易于扩充，适合于各种应用软件、系统软件的设计。

(3) 移植性好，几乎所有的计算机都可以使用 C++ 编程，把用 C++ 编写的程序从一台机器迁移到另一台机器上变得非常容易。

(4) 程序可读性好，生成的代码质量高，但运行效率仅比汇编语言慢 10%～20%。

1.4 第一个 C++ 程序

例 1-1 在显示器上输出字符串“Hello world!”。

分析：和 C 程序一样，C++ 程序的开始部分一般也为文件包含，程序的入口函数仍为 main()，main()函数的返回一般为 int 类型(CodeBlocks 中要求返回 int，也有一些编译器中可返回 void，如 VC++ 6.0)，C++ 的输入输出和 C 相比有些区别。程序设计如下：

```
#include <iostream>
using namespace std;                    //使用 C++标准库
int main()
{
    cout <<"Hello world!" <<endl;
    //输出"Hello world!"字符串，endl 用于换行，相当于 C 语言中的 printf("\n");
    return 0;                           //main()函数的返回值
}
```

编译运行该程序后，输出结果如下：

```
Hello world!
```

该程序只有三部分，分别为包含的头文件、应用的名字空间和 main()函数，该程序和同样功能的 C 程序主要不同之处在第二行的 using namespace std，C++ 程序中说明了程序应用的名字空间，而在 C 程序中则没有这一说明。另外 C++ 的输出格式和 C 有些区别，C++ 中用了 cout，在 C 中人们经常用 printf()输出数据。当人们保存该程序源文件时还会发现文件的后缀为 cpp。

1.5 C++语言的文件扩展名

1. 源文件扩展名

C语言源程序文件的扩展名为c。

C++源程序文件的扩展名为cpp(意即C Plus-Plus),不管内部代码如何编写,即使内部全为C语言代码,C++源文件的扩展名总是cpp。

2. 头文件扩展名

C语言程序的头文件的扩展名为h。

C++语言程序的头文件的扩展名为h或没有扩展名,如例1.1中的iostream,就没有用扩展名。一般情况下,自定义的头文件使用h,而使用C++标准库中的头文件时,则加或不加h有些区别。一般情况下,头文件的应用方法如下:

```
#include <stdio.h>                    //包含C语言头文件
#include <iostream>                   //包含C++标准头文件
#include "myheader.h"                 //包含自定义头文件
```

3. iostream和iostream.h的区别

iostream和iostream.h是两个不同的文件,人们在编译器的include文件夹里面可以看到,如果打开文件就会发现,里面的代码也不一样。早期C++的实现将标准库功能定义在全局空间里,声明在带.h后缀的头文件里,新的C++标准为了和C区别开,也为了正确使用命名空间,规定头文件不使用后缀.h。因此,当使用<iostream.h>时,相当于在C中调用库函数,使用的是全局命名空间,也就是早期的C++实现;当使用<iostream>的时候,该头文件没有定义全局命名空间,必须使用"namespace std;",这样才能正确使用cout。

1.6 注释符

注释在程序编译时被编译器忽略,不会参与程序运行,但对人们读懂代码非常有用,所以从程序阅读和程序维护的角度来说,在程序代码中编写相应的注释是十分必要的。C++可用两种注释方式,分别是段注释(/*…*/)和单行注释(//)。

段注释方式为

```
/*在这里添加注释的文字,对是否换行没有限制*/
```

段注释和C语言的注释方式一样,适合于加入注释的内容有多行的情况。

单行注释的方式为

```
//这里是注释的文字
```

不能换行，若要换行，必须使用分段注释或每行单独注释。

需要特别注意的是，人们在编写和修改程序代码的时候，就要考虑在代码中添加或修改注释，帮助人们今后快速读懂代码。假想自己过一段时间后能不能快速阅读并很快掌握这段代码的意思（包括每个变量的含义、初值等），别人是否也能快速读懂呢？另外在调试程序时，注释符有时也会起到一些作用，例如，人们有时希望不让一行或一段程序执行，可以使用注释符去实现。

1.7 名字空间

名字空间是面向对象程序设计的重要概念。它是一种作用域，主要用于解决大型程序中命名冲突的问题，为用户提供一个空间，在里面可定义自己的函数或类，当定义的函数或类和他人定义的重名时，可用名字空间来区分。

名字空间用花括号把文件的一部分括起来，并以关键字 namespace 开头，例如：

```
namespace  ns1
{
    const double PI=3.1415926;
    //其他定义
    int Fun(int a,int b)              //在 ns1 名字空间中定义了一个名为 Fun 的函数
    {
        return a+b;
    }
}
```

PI 和 Fun 是名字空间 ns1 中的常量和函数，函数 Fun 在 main 函数中的用法如下：

```
int main()
{
    cout<<ns1::Fun(15,4);
                    //使用的是 ns1 名字空间中的 Fun 函数，如果去掉 ns1，编译时会出错
    return 0;
}
```

也可以为

```
using namespace ns1;                  //使用 ns1 名字空间
int main()
{
    cout<<Fun(15,4);                  //这里不需要加 ns1
    return 0;
}
```

在 C++ 标准程序库中的所有标识符都被定义于一个名为 std 的 namespace 中，人们使用标准 C++ 库中的组件时，只要写一个 using 指示符（using namespace std;）就可以使

用该空间的标识。如果不用“using namespace std;”则使用的时候要在每个标识前都加上 std::,类似 Fun 函数前要加上 ns1 一样。

注意：如果使用了名空间 std,则在使用#include 编译预处理命令包含头文件时,必须去掉头文件的扩展名.h,否则会出错。

1.8 C++中的 new 和 delete

程序中经常见到关于内存的动态分配和撤销的问题。在 C 语言中,使用 malloc 函数向系统申请分配指定 size 个字节的内存空间,该函数返回 void * 类型,malloc 函数不对申请到的空间做任何初始化处理,当这块空间不再需要时,使用 free 函数进行释放。而 C++ 中,对于内存的分配和撤销,则引入了 new 和 delete 操作符,new 操作符可以申请空间,并调用构造函数进行对象初始化,delete 操作符释放对应 new 申请的空间,并调用析构函数做一些清理工作。在 C++ 中运用 new 和 delete 之后,完全可以抛弃 C 语言中的 malloc 和 free 函数。以下是 C 和 C++ 中的一些用法。

```
int *parrc;
parrc=(int*)malloc(100*sizeof(int));
/*其他程序语句*/
free(parrc);                              /*释放 parrc 所指向的空间*/
```

以上为 C 语言格式的内存申请方式,申请 100 个 sizeof(int)大小的连续空间,然后把这些空间转换为 int 型格式并返回这片空间的首地址给 parrc,运行其他程序语句后释放这片空间。

在 C++ 中,使用 new 和 delete 之后,程序更简洁,和上段 C 程序实现相同功能的代码如下:

```
int *parr;
parr =new int[100];
                    //申请 100 个连续的 int 类型大小的空间,比 malloc 的表达似乎更加简洁
/*其他程序语句*/
delete []parr;                            //释放 parr 所指向的空间
```

以上为 C++ 语言格式的内存申请方式,申请 100 个 int 型大小的连续空间,然后把这片空间的首地址返回给 parr,运行其他程序语句后释放这片空间。

又如:

```
char *name;                               //定义一个字符类型的指针
name=new char[12];                        //申请 12 个连续的 char 类型大小的空间
/*其他程序语句*/
delete []name;                            //释放 name 所指向的空间
```

C 语言中 malloc 申请的空间是字节类型,一般需要做强制类型转换,而 C++ 中 new 申请的空间一般直接指定其类型,不需要再做强制类型转换。new 和 delete 在 C++ 程序

设计中具有重要的作用，它们分别会调用类的构造函数和析构函数，在本书后面的章节再做较详细的介绍。

1.9 C++ 语言的输入输出

在 C 语言中，所有的数据输入输出都是由库函数完成的，因此都是函数语句，如 printf 函数、scanf 函数、putchar 函数和 getchar 函数等。

C++ 的输入输出采用“流”的形式，流是一种抽象的形态，是指计算机里面的数据从一个对象流向另一个对象。数据流入和流出的对象通常为计算机中的屏幕、内存、文件等输入输出设备，数据的流动由 I/O 流类来实现，C++ 中预定义的流对象有 cin 和 cout（有关流的讨论，具体见第 7 章）。

1. cin

cin 为标准输入流对象，在默认情况下是从键盘输入。

2. cout

cout 为标准输出流对象，在默认情况下是输出到显示器。

输入输出函数声明：

```
#include <iostream.h>
```

或

```
#include<iostream>                    //使用名字空间 std,必须去掉扩展名 h
using namespace std;
```

例 1-2 C++ 的输入输出举例，从键盘输入，输出字符串和数字。

分析：使用 cin 从键盘输入变量的值，使用 cout 输出字符串或变量的值到显示器。

```
#include <iostream>                    //使用名字空间 std,必须去掉扩展名 h
using namespace std;
int main()
{  char  name[16];
   int  age;
   cout<<"Please input your name:";  //默认输出到显示器
   cin>>name;                        //默认从键盘输入
   cout<<"How old are you:";
   cin>>age;
   cout<<"Name is "<<name<<endl;     //endl 用于结束当前行
   cout<<"Aage is "<<age<<endl;
   return 0;
}
```

编译运行该程序后，输出结果如下：