C++程序设计

郭有强　梁伍七　杨　军　朱洪浩　编　著

安徽大学出版社

内 容 简 介

本教材是安徽省高等学校“十一五”省级规划教材。

C++是一种高效实用的程序设计语言，它既可以进行过程化程序设计，也可以进行面向对象程序设计，因而成为编程人员最广泛使用的工具。本教材采用通俗易懂的语言，全面系统地介绍了C++的基本概念；通过大量精选的具有典型性的例题和实训，重点讲述解决问题的思路，帮助读者理解编程思想及相关概念；通过具有综合性的项目设计，注重读者实际编程能力的培养和提高。

本教材是作者总结多年教学经验，参考大量的国内外有关资料并结合自身的实际工程项目经验编写而成，内容丰富，结构紧凑，概念阐述清楚，可作为高等院校计算机专业和非计算机专业的程序设计教材，也可供C++程序员和计算机软件技术人员作为程序设计的参考书。

本教材中所有的例程都在Visual C++ 6.0下调试通过。其配套教材《C++程序设计实验指导与课程设计》也将同时出版。

图书在版编目(CIP)数据

C++程序设计/郭有强，梁伍七，杨军，朱洪浩编著.
—合肥：安徽大学出版社，2008.5
ISBN 978-7-81110-413-4

Ⅰ.C… Ⅱ.①郭…②梁…③杨…④朱… Ⅲ.C语言
—程序设计—高等学校—教材 Ⅳ.TP312

中国版本图书馆CIP数据核字(2008)第061810号

C++程序设计 郭有强 梁伍七 杨 军 朱洪浩 编著

出版发行	安徽大学出版社	经 销	新华书店
	(合肥市肥西路3号 邮编230039)	印 刷	合肥创新印务有限公司
联系电话	发行部0551-5107716 5108397	开 本	787×1092 1/16
E-mail	ahdxchps@mail.hf.ah.cn	印 张	17.625
网 址	www.ahupress.com.cn	字 数	429千
责任编辑	李镜平	版 次	2008年6月第1版
特约编辑	罗 罹 罗季重	印 次	2008年6月第1次印刷
封面设计	孟献辉		

ISBN 978-7-81110-413-4 定价27.60元

序

关于本书的内容

本教材是安徽省高等学校“十一五”省级规划教材。

随着计算机科学技术的迅速发展，程序设计技术和程序设计语言也不断发展。目前，面向对象程序设计是软件开发领域的主流技术。这种技术从根本上改变了人们以往设计软件的思维方式，它集抽象性、封装性、继承性和多态性于一体，实现了代码重用和代码扩充，极大地减少了软件开发的繁杂性，提高了软件开发的效率。C++为面向对象技术提供全面支持，是主流的面向对象程序设计语言。因此C++在当前程序设计领域中的地位是很重要的。它是一个可编写高质量的用户自定义类型库的工具，其核心应用领域是最广泛意义上的系统程序设计。此外，C++还被成功地运用到许多无法称为系统程序设计的应用领域中，几乎所有操作系统上都有C++的实现。

C++是“带类的C”，是面向对象的程序设计(Object－Oriented Programming)语言。高等院校计算机专业和相关专业基本上都开设了该课程，其目的是为了让学生掌握面向对象程序设计的概念和方法，深刻理解面向对象程序设计的本质，并用面向对象技术来编写程序，开发软件。长期以来，通过教学实践发现学生在学习C++的过程中，普遍感觉到C++不好掌握，尤其是它的面向对象设计思想及相关概念在接受时感到困难，学生的实际动手编程的能力较差，更谈不上应用了。鉴于此，作者结合十多年本课程的教学经验和体会，编写了一本符合当前学生接受能力的、通俗易懂的教材。

只有掌握了VC++，才能使学生真正感受到C系列的魅力，也才能真正和当前的Windows编程接轨。在很多院校的培养方案中，有的只开设C语言程序设计，这样不符合现在主流编程思想；有的先开C，再开C++，这样还是不能和当前的Windows程序设计相结合，好像还差一步；有的是直接开设C++，再开VC++，学生接受有问题；有的先开设C，再开设C++，最后开设VC++，整个C系列战线太长，占用的学时数太多。对于教材来说，大部分C++教材都是从面向过程的C++部分(基本上就是C语言内容)开始写的，一直写到面向对象的C++部分。这样做，开设过C语言的院校感觉前面部分是多余的，而且缺少可视化Windows编程部分。有的教材力求完整，编写了有关的全部内容，结果面向过程的C++部分不到位，类的部分也难到位。这样，教师在选择教材和教学过程中如何处理教材内容等方面都有所不便。基于以上原因，根据多年的教学实践，考虑到大部分院校都是把C语言作为计算机程序设计的入门语言先行开设，为从C语言平滑地过渡到C++，

本教材首先讨论了面向过程的C++对C语言的扩充部分，然后主要写面向对象的C++部分，最后介绍MFC编程。这样做很适合有C语言基础的学生，可基本完成整个C系列的学习任务。

本书的目标是帮助读者深刻理解面向对象程序设计的思想，掌握C++程序分析能力和设计技能，学会运用MFC进行Windows程序设计。

关于本书的结构

全书共9章，分成3个部分。

第1部分(第1章、第2章)讲授了C++概述和C++对C语言的扩充。这部分内容从总体上介绍了面向过程和面向对象技术的基本概念、不同的编程思维方式、相关技术以及C++在基本语法上对C语言的扩充等，最后介绍了如何利用Visual C++ 6.0调试C++控制台应用程序。

第2部分(第3章到第7章)主要讲授C++面向对象部分，是本书的核心。这部分内容以面向对象技术的四大特征为线索展开对C++的讨论，为第3部分的应用奠定理论基础。

第3部分(第8章、第9章)是面向对象技术在Visual C++中的应用部分。这部分主要介绍了利用MFC开发Windows应用程序的基本编程模式和程序设计方法。为什么要加强这部分内容呢？原因是C++加入可视化概念后，并不像一些可视化程序设计语言(如VB、Dephi语言等)那样容易掌握。Visual C++结构复杂，代码量大，专业化程度高，初学者很难把握Visual C++的脉搏。第8章通过循序渐进的方式讲述了利用MFC编程的模式、方法，配以应用详细的实训步骤，读者学习后即可基本入门，便于日后进阶。第9章还给出了基于两种编程模式的综合性较强的项目设计，以提高读者的综合编程能力。

关于本书的特点

本书首先介绍了面向过程和面向对象程序设计的区别，以及为什么要采用面向对象的方法，如何将C语言程序简单改造成C++模式；然后主要围绕面向对象程序设计的四大特征展开。每一部分为了使学生对所有概念有充分的理解，首先说明开设这部分内容的理由；在阐述每部分内容时既重点突出，又注重细节用法；为便于学生自学，采用了大量的注释；每部分都精选了大量的典型实例。最后讲述了MFC的编程模式，以使读者感受到C系列的魅力，解除学生在学习了C及C++后感觉当前没什么实际应用(因为现在大都是可视化的Windows编程)，没有像其他可视化程序设计语言那样有立竿见影的效果和如何与当前流行的可视化编程接轨的困惑，增强了学生在学习语言后的成就感，以提高学生对软件编程的兴趣和能力。本书力求用通俗易懂的语言、生活中的现象来阐述基本理论，突出重点，解释到位，讲深讲透“难点”部分。所有内容均采用提出问题、说明问题、解决问题的模式编写，能够更加贴近学生；采用先例题后理论的模式，用程序例子来说明难懂的抽象概念；对许多问题描述多个编程方案，以使读者在比较中学习，进而深刻理解相关设计思想和分析算法。本书除了正面阐述，说明什么是正确的，还注意从反面来说明什么是不正确的；在讲解MFC编程的过程中，注重操作步骤及细节，以使读者具有很强的可操作性。

本书内容丰富，结构紧凑，概念阐述清楚，例题典型丰富，注重能力培养；提供了大量实例、实训、习题和实验，突出实训环节，以便读者深刻理解编程思想和增强实际动手编程

能力。

本书中所有的例程都在 Visual C++ 6.0 下调试通过。

本书适用对象

本教材是作者总结多年教学经验，参考大量国内外有关资料，并结合自身的实际工程项目经验编写而成，融会贯通了 C++面向过程、C++面向对象以及利用 VC++进行 Windows 程序设计等方面内容，是一本内容较全面的教材；可作为高等院校计算机专业和非计算机专业的程序设计教材，也可供 C++程序员和计算机软件技术人员作为程序设计的参考书。

本书由郭有强主编（编写第 1、2、4、8、9 章），梁伍七副主编（编写第 3 章及本书全部例题源代码的测试）。参加本书编写工作的还有杨军（编写第 6、7 章），朱洪浩（编写第 5 章及附录）。全书由郭有强总体设计并统稿。杨军制作了电子讲稿以及本书全部例题的源代码，一起放在网站 www.bbxy.edu.cn 上，供教师下载。参加本书配套的《C++程序设计实验指导与课程设计》教材编写和文字工作的有王一宾、张怡文、叶家鸣、侯传宇，他们做了大量的工作，在此表示感谢！感谢袁兆山教授、胡学钢教授、王浩教授，他们对本书给予了极大的关注和支持，提出了宝贵的建设性意见。感谢本书所列参考文献的作者！

由于作者水平有限，加之时间仓促，错误与疏漏之处在所难免，敬请读者不吝赐正。在使用该书时如遇到什么问题需要与作者商榷，或想索取其他相关资料，请与作者联系。联系方式：bbxyguo@163.com。

郭有强

2008 年 5 月

目 录

第 1 章　C++概述

学习目标

- 掌握面向过程和面向对象编程的特点和两者的不同
- 了解 C++的发展过程
- 能够将 C 语言源程序简单转变成 C++书写风格
- 掌握利用 Visual C++ 6.0 集成开发环境调试 C++控制台应用程序

1.1　面向过程和面向对象编程概述

面向过程的程序设计(The Procedure－Oriented Programming)方法诞生于 20 世纪 60 年代，其后风行全球，成为软件开发的基础。

面向过程程序设计采用结构化思想，这种程序设计方法的特点是“就事论事”，按照人们解决问题的习惯进行编程：把大问题细分为许多小任务，分而治之，各个击破。总的设计思路是：自顶向下，逐步求精。对于一个复杂过程，按其功能分解为若干个有序的基本模块。然后，再把每个模块进一步细化，直到子模块变得清晰，易于实现。每一个模块内部都可以由顺序、选择和循环等三种基本结构组成。系统功能划分如图 1.1 所示。

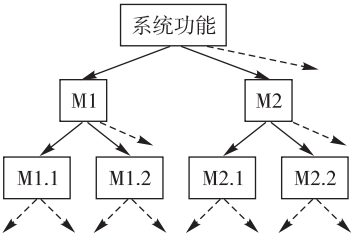

图 1.1　系统功能划分

这里着眼于过程的模块化，而非数据的模块化，使得数据和相关运算相分离，程序被描述为：

程序 =(模块 + 模块 + …)

模块 =(算法)+(数据结构)

在面向过程的程序中，所有数据是公开的。一个函数可以使用和改变任意一组数据，而一组数据又可能被多个函数使用。这种数据与运算相分离的结果失去了数据的安全性。一旦数据结构发生变化，相关的算法也必须随之改动。对于相同的数据结构，若操作不同，也要编写不同的程序。因此，面向过程的程序代码重用性不好。程序模块的划分，因人而异，缺乏统一的标准，为程序员之间的交流带来诸多不便。另外，面向过程程序设计的逐步细化过程前后关系密切，描述符号不同，一旦先期需求改变，将直接影响后继需求分析的描述，给

程序的维护带来诸多不便。

面向过程的程序设计方法本质上是过程驱动的。虽然在处理问题的方法上符合人们思考问题的规律，但它将数据与操作数据的函数分离开来，未能如实地反映客观世界的规律。事实上，客观世界中的事物总是分门别类的；每个类有自己的数据与操作数据的方法，二者是密不可分的。

面向对象程序设计的基本思想是：现实世界由各种对象组成，任何客观存在的事物都是对象，复杂的对象是由简单对象结合而成的。面向对象程序设计的基石是：类和对象。类是具有相同属性结构和操作行为的一组对象共性的抽象，对象是描述客观事物的属性结构及定义在该结构上的一组操作的结合体。在此，程序被描述为：

程序 ＝（对象 ＋ 对象 ＋ …）

对象 ＝（数据结构 ＋ 算法）

对象之间通过消息和方法机制完成相应的操作。

程序员根据具体情况，先设计一些类；每个类有数据成员和操作这些数据的成员函数。然后，定义各个类的对象，并将数据赋给各个对象。对象的数据是私有的，外界只能通过公有的成员函数才能访问该对象的数据。这样就保证了数据的安全性，而且程序员也易于对数据进行跟踪。类的继承性使得每一个新类得以继承基类、父类的全部属性和方法，并加入自己特有的属性和方法，从而使得代码的重用成为可能。类对数据结构和算法的绑定，使得程序便于修改和调试，便于程序的维护和扩充。

每个对象是数据和操作代码的完整结合体。各个对象通过消息传递而相互作用。所以，面向对象的程序本质上是事件驱动的。这一点很重要，它使得一个原先很复杂的程序变得简单清晰。这种优势在可视化程序设计中极为明显，如图 1.2 所示。

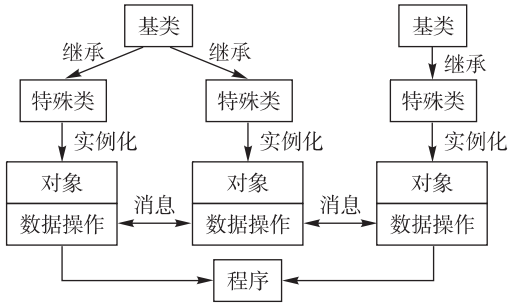

图 1.2　面向对象各要素之间的关系

面向对象程序设计语言有以下 4 个特征：

（1）抽象性——许多实体的共性产生类；

（2）封装性——类将数据和操作封装为用户自定义的抽象数据类型；

（3）继承性——类能被复用，具有继承（派生）机制；

（4）多态性——具有动态联编机制。

C＋＋改进和扩充了 C 语言的类型系统。最重要的扩充是支持面向对象的程序设计。

面向过程以算法为中心，由算法完成对数据的操作，面向对象技术是以属性为中心，以消息和方法机制完成对对象的操作，对象作为数据而不是作为过程被描述。继承指的是子类继承父类的数据结构和方法，并加入自己所特有的信息，构成新的类；它是父类和子类之

间信息关联的一种机制。多态体现在程序运行过程中，不同对象收到相同消息产生完全不同的结果的现象。继承和多态是面向对象特有的要素。

面向过程与面向对象两者要素对应关系如图 1.3 所示。

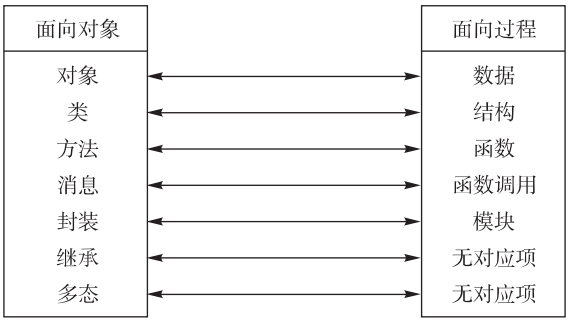

图 1.3　面向过程和面向对象之间的对应关系

上面概述了面向对象程序设计的思路和特点。用面向对象的思路去思考问题，但这决不意味着可以忽略面向过程的程序设计方法的学习；恰好相反，只有掌握面向过程的程序设计方法，才能更熟练地编写 C＋＋类中的成员函数，而且对于简单的程序，用面向过程的设计方法也就足够了。

1.2　C＋＋的发展过程

C 语言是面向过程的程序设计语言，不适合用于开发大型程序。当程序规模稍大或对象之间的关系稍微复杂时，程序员就很难控制程序的复杂性。例如，往往需要设计带有可视界面的应用程序，即使界面上只有少数几个控件，但用 C 语言就很难设计。

为了解决上述问题，并保持 C 语言的简洁、高效和接近汇编语言的特点，1980 年贝尔实验室的 Bjarne Stroustrup 博士及其同事对 C 语言进行了改进和扩充，这种新型的 C 语言最初被称为“带类的 C”；1983 年正式命名为 C＋＋(C Plus Plus)，其含义是 C 语言的扩充。后来又把运算符重载、引用、虚函数、模板等功能加入到 C＋＋中，使 C＋＋的功能日趋完善。目前，C＋＋已成为面向过程和面向对象的主流通用程序设计语言。

C＋＋受到软件厂商的极大支持，纷纷推出其商业化 C＋＋编译系统，从早期的 Turbo C＋＋、Borland C＋＋、Watcom C＋＋、Quick C＋＋到目前流行的 Visual C＋＋和C＋＋，Builder。C＋＋也受到开放源代码组织的积极支持，它们也纷纷推出自己的非商业化的 C＋＋编译系统，如 GNU C＋＋、DEV C＋＋等。

C＋＋的标准化工作始于 1989 年，于 1994 年制定了 ANSI C＋＋标准草案。经过不断修改完善，于 1998 年 11 月被国际标准化组织(ISO)批准为国际标准。

C＋＋全面兼容 C 语言，除了具备 C 语言的特点外，还具有以下特点：

(1) 全面兼容 C 语言，全面支持面向过程的结构化程序设计。C＋＋是 C 语言的超集，大多数 C 语言程序代码略做修改或不做修改，就可以在 C＋＋编译系统下编译通过。这样，既保护了用 C 语言开发的丰富软件资源，也保护了丰富的 C 语言软件开发人力资源。

(2) 全面支持面向对象程序设计。以对象为基本模块，使程序模块的划分更合理，模块的独立性更强，程序的可读性、可理解性、可重用性、可扩充性、可测试性和可维护性等更好，

程序结构更加合理。

(3) 全面支持面向过程和面向对象的混合编程,充分发挥两类编程技术的优势。

1.3 将C源程序简单改写成C++源程序

【例1.1】 输入两个整数,输出其中的大者。

程序代码如下:

```
//e1_1.c
#include "stdio.h"
main()
{
   int a,b,c;
   scanf("%d,%d",&a,&b);                    //输入两个整数
   c = max(a,b);
   printf("max is %d",c);                   //输出信息
}
int max(int x,int y)
{
   int z;
   z=(x>y?x:y);
   return z;
}
```

下面用C++来编写一个功能完全相同的程序。

```
#include <iostream.h>
void main()
{
   int a,b,c;
   cin>>a>>b;                               //输入两个整数
   c = max(a,b);
   cout<<"max is"<<c<<endl;                 //输出信息
}
int max(int x,int y)
{
   int z;
   z = (x>y?x:y);
   return z;
}
```

对比分析:

C++的预处理程序指令为#include <iostream.h>,在程序中包含了iostream.h这

一输入/输出流头文件的内容。采用 C++风格的流输入/输出，将数据输出至屏幕，或从键盘输入数据，便应将此头文件包括在程序内。iostream.h 的内容将在以后详细解释。

cin 和 cout 分别为输入流对象和输出流对象，“>>”为抽取操作符，“<<”为插入操作符，相关的内容在后面介绍。“cin>>”是把标准输入设备(键盘)接收到的数据，存入“>>”后面的变量中。“cout<<"max is"”是将“<<”后面的内容送到标准输出设备(显示器)。如果把“>>”和“<<”看作是表示方向的符号，就会发现数据确实是在按照一定的方向流动，这就是“流”这个名称的由来。

虽然 C 语言风格的输入输出在 C++中也是允许的，但是，C++风格的输入输出更方便。

1.4　面向过程和面向对象的程序设计方法比较

为了对 C++程序的基本结构有所了解，【例 1.2】、【例 1.3】和【例 1.4】分别给出了用面向过程和面向对象的程序设计方法计算圆面积的 C++程序。

【例 1.2】　面向过程程序设计。输入圆的半径，计算并输出该圆的面积。

程序代码如下：

```
//e1_2.cpp
#include<iostream.h>
void main()
{
   float r;                                    //定义浮点型变量 r,用于存放圆的半径
   cout<<"输入圆的半径:";
   cin>>r;                                     //从键盘上输入圆的半径送给变量 r
   cout<<"半径为"<<r<<"的圆的面积="<<3.14159f * r * r<<'\n';
                                               //输出运算结果
}
```

程序运行结果为：

```
输入圆的半径:1.5
半径为 1.5 的圆的面积=7.06858
```

程序说明：

(1) 程序中加入必要的注释，可提高程序的可读性。注释有两种：①用“/*”和“*/”把注解括起来，对程序进行详细说明，它可出现在程序的任何位置，可加入多行注释、单行注释，也可加入嵌入注释；②用“//”表示从此开始到本行结束为注释，通常用于单行注释。编译器对程序中的注释不做任何处理，注释对目标代码没有任何影响。

(2) C++严格区分大小写字母。在书写程序或编辑程序时，要注意这一点。

【例 1.3】　面向过程的模块化程序设计。输入圆的半径，计算并输出该圆的面积。

程序代码如下：

```
//e1_3.cpp
#include<iostream.h>
```

```
float area(float r)                          //定义求半径为 r 的圆的面积的函数 area
{ return 3.14159f * r * r; }
void main()
{
  float r;                                   //定义浮点型变量 r,用于存放圆的半径
  cout<<"输入圆的半径:";                     //显示提示信息,提示用户输入数据
  cin>>r;                                    //从键盘上输入圆的半径送给变量 r
  cout<<"半径为"<<r<<"的圆的面积="<<area(r)<<'\n';
                                             //输出运算结果
}
```

模块化程序设计的主要思路是:把一个复杂问题按功能分解成若干个较为简单的子问题,每个子问题通过定义一个函数来解决;如果子问题还不够简单,再继续按功能分解下去,直到所有子问题都能解决为止。这样解决一个复杂问题的函数就可以通过调用一系列解决子问题的函数来实现。这就是所谓的“自顶向下,逐步求精”的程序设计方法,是“化整为零,各个击破”思想在程序设计中的体现。

【例 1.4】 面向对象程序设计。输入圆的半径,计算并输出该圆的面积。

程序代码如下:

```
//e1_4.cpp
#include<iostream.h>
class Circle                                 //定义一个计算圆的面积的类 Circle
{
  private:
  float r;                                   //定义成员数据变量 r,用于存放圆的半径
  public:
  Circle(float a) { r = a; }                 //定义构造函数,用于创建和初始化对象
  ~Circle( ){}                               //定义析构函数,用于清理和撤销对象
  void SetRadius(float a) { r = a; }         //定义成员函数,用于设置圆的半径 r
  float GetRadius(){ return r; }             //定义成员函数,用于获取圆的半径 r
  float Area(){ return 3.14159f * r * r; }   //定义成员函数 Area,用于计算圆的面积
};
void main()
{
  float r;                                   //定义浮点型变量 r,用于存放圆的半径
  cout<<"输入圆的半径:";                     //显示提示信息,方便用户输入数据
  cin>>r;                                    //从键盘上输入圆的半径送给变量 r
  Circle c(r);                               //定义 Circle 类的对象 c
  cout<<"半径为"<<c.GetRadius()<<"的圆的面积="<<c.Area()<<'\n';
                                             //输出运算结果
}
```