21世纪 计算机基础教育系列教材

谭浩强 主编

面向对象的C++程序设计

■李宁 著

本书附光盘一张

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

21世纪计算机基础教育系列教材

谭浩强 主编

面向对象的C++程序设计

李 宁 著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书以 C++ 这种最典型的面向对象程序设计语言为媒介，全面地介绍了面向对象程序设计的基本理论、方法和技巧，同时也提供了运用 C++ 语言解决实际问题的实际技能练习。本书每章后面都有习题，其中的部分程序设计题可安排为上机实习。本书配有光盘，其中包含了书中全部例题。

本书适合于做计算机技术及应用类专业、工程技术类专业，以及其他理工科相关专业的本科或专科教材，也可作为相关技术人员的自学参考书。

图书在版编目(CIP)数据

面向对象的 C++ 程序设计/李宁著 .—北京：电子工业出版社，2002.6
21 世纪计算机基础教育系列教材
ISBN 7-5053-7605-5

Ⅰ. 面… Ⅱ. 李… Ⅲ. C 语言—程序设计—教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2002)第 031344 号

责任编辑：冉 哲
印 刷：北京四季青印刷厂
出版发行：电子工业出版社 http://www.phei.com.cn
北京市海淀区万寿路 173 信箱 邮编 100036
经 销：各地新华书店
开 本：787×1 092 1/16 印张：19 字数：486.4 千字 附光盘 1 张
版 次：2002 年 6 月第 1 版 2002 年 6 月第 1 次印刷
印 数：5 000 册 定价：27.00 元(含光盘)

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。
联系电话：(010)68279077

《21 世纪计算机基础教育系列教材》序

21 世纪是信息时代，是科学技术高速发展的时代。计算机技术与网络技术的结合，使人类的生产方式、生活方式和思维方式发生了深刻的变化。在新世纪中，计算机知识已成为当代人类文化的一个重要组成部分了。我们要把计算机知识和应用向一切有文化的人普及。

高等学校承担着为社会培养高层次人才的任务，大学生毕业后应当成为我国各个领域中的计算机应用人才，成为向全社会推广计算机应用的积极分子。在大学里应当把计算机教育放在十分重要的位置。

我国高校的计算机基础教育起步于 20 世纪 80 年代初。20 年来从无到有地迅速发展，从理工科专业发展到所有专业，从最初只开设一门语言课到按三个层次设置课程，学时也从三四十小时增加到一二百小时。计算机基础教育已经先后上了几个台阶，现在又需要上一个新的台阶。在新世纪初，我们要求提高大学生应用计算机的能力，以适应科学技术和经济发展的需要。

我们在这里所说的计算机基础教育，是指面对全体大学生的计算机教育。非计算机专业和计算机专业中的计算机教育的特点有很大的区别。无论学生基础、培养目标、教学要求、教学内容、教学方法和教材，都和计算机专业有很大的不同。绝不可简单地照搬计算机专业的模式，否则必事倍功半。计算机基础教育实际上是计算机应用的教育，应当以应用为目的，以应用为出发点。

计算机不仅是一种工具，也是一种文化，工具是可选的，文化却是必备的。对学生来说，它还是全面素质教育的一个重要部分，通过学习计算机知识能激发学生对先进科学技术的向往，启发学生对新知识的学习热情，培养学生的创新意识，提高学生的自学能力，锻炼学生动手实践的能力。多年来的实践证明，对计算机感兴趣的学生，绝大多数都是兴趣广泛、思想活跃、善于思考、自学能力较强、喜欢动手实践的。他们绝不是只会死背书本的书呆子。

我们必须认真分析非计算机专业的特点，根据教学上的需要与可能，制定出恰当的教学要求，使学生在有限的时间内能学到最多的有用的知识。全国高等院校计算机基础教育研究会曾提出了在计算机基础教育中应当正确处理的 10 个关系，即：(1)理论与应用的关系；(2)深度与广度的关系；(3)当前与发展的关系；(4)硬件与软件的关系；(5)追踪先进水平与教学相对稳定的关系；(6)课内与课外的关系；(7)课程设置与统一考试的关系；(8)计算机课程与其他课程的关系；(9)要求学生动手能力强与当前设备不足的矛盾；(10)计算机技术发展迅速与师资现状的矛盾。在教学实践中，许多学校都创造了丰富的经验。

在非计算机专业的教学中，首先需要解决的问题是：准确定位，合理取舍教学内容。我们必须分清楚：哪些内容是需要的，哪些内容是不需要的；哪些内容是目前暂时可以不学而留待以后学的，哪些内容是目前不必学而以后也不必学的；哪些内容是主要的，哪些是次要的。绝不可眉毛胡子一把抓，不加分析、不问主次，使学生感到难以入门。

在教学方法和教材的编写上，要善于用通俗易懂的方法和语言说明复杂难懂的概念。传统的教学三部曲是：提出概念——解释概念——举例说明。我在多年教学实践中对于计算机应用课程总结了新的三部曲：提出问题——介绍解决问题的方法——归纳出必要的概念和

结论。从具体到抽象，从实际到理论，从个别到一般。这是符合人们的认识规律的。实践证明，这样做已取得了很好的效果。

为了推动高校的计算机基础教育，我在 1996 年主编了《计算机教育丛书》，由电子工业出版社出版。编写这套丛书的指导思想是 20 个字：“内容新颖、实用性强、概念清晰、通俗易懂、层次配套”（也可简单地概括为：“新颖、实用、清晰、通俗、配套”）。先后出版的近 20 种供大学非计算机专业使用的教材，受到高校广大师生的欢迎，几年内发行达 75 万册，大家认为它定位准确、程度适当、内容丰富、通俗易懂，便于自学。

在进入 21 世纪之际，我们根据新时期的要求，按照上面所述的指导思想，重新进行规划，对原有的教材进行了筛选，淘汰了部分内容已过时的教材，同时根据计算机技术和高校计算机基础教育的发展组织了一些新教材，并对原有教材进行了修订补充，以实现推陈出新，不断提高。

我们遴选了具有丰富教学经验的高校老师编写这套教材。在这套系列教材中，我们提供了多种课程的教材供各校选用，其中包括必修课和选修课。不同专业、不同层次的学校都可以从中选到合用的教材，我们还将根据发展不断推出新的教材。

参加本丛书策划、组织和编写工作的有：谭浩强、史济民、薛淑斌、吴功宜、边奠英、徐士良、赵鸿德、李盘林、孟宪福、张基温、宋国新、徐安东、毛汉书、李凤霞、许向荣、周晓玉、张玲、刘星、秦建中、王兴岭等。电子工业出版社对本丛书的出版给予了大力的支持，使得本丛书得以顺利出版。

由于我们的水平和经验有限，加以计算机科学技术发展很快，本丛书肯定会有不少缺点和不足，诚恳地希望专家和读者不吝指正，我们将继续努力工作，使本丛书能尽量满足读者的要求。

全国高等院校计算机基础教育研究会会长
《21 世纪计算机基础教育系列教材》主编　谭浩强

2001 年 7 月 1 日

前　言

面向对象方法是正在快速发展并成为主流的软件系统开发方法，包括面向对象的系统分析、面向对象的系统设计和面向对象的程序设计。系统分析、系统设计、程序设计是软件开发的三个基本步骤，但从面向对象方法的发展过程看，面向对象程序设计是其中最先发展起来的，在技术上也最为成熟。

面向对象的程序设计（Object Oriented Programming，简称 OOP）是一种围绕真实世界的概念来组织模型的程序设计方法，与传统的面向数据及数据处理过程的方法不同的是，这种方法把数据及与之相关的处理过程作为一个有机的整体看待，这样的整体我们称之为对象。通过对象这种实体，更容易完整地反映现实问题本质，因而便于对问题进行自然的、符合人们思维习惯的分析，进而建立能够更直观地反映客观世界的模型。根据这样的模型设计出的软件是由一系列对象组成的，通过对象间的消息传递，完成系统的功能。面向对象的程序设计方法克服了传统设计方法的主要缺点，因而得到日益广泛的应用，是软件工程领域的重大突破。

本书以 C++这种最典型的面向对象程序设计语言为媒介，全面介绍面向对象程序设计的基本理论、方法和技巧。之所以要借助于 C++语言，是因为 C++全面支持面向对象程序设计，能较为全面地反映面向对象方法中的各种概念，同时又应用广泛。因此，通过 C++语言来学习面向对象程序设计，不但有利于全面、系统地掌握面向对象程序设计的基本概念、理论和方法，同时也可获得运用 C++语言解决实际问题的实用技能。

本书适合作为计算机技术及应用类专业、工程技术类专业，以及其他理工科相关专业的本科或专科教材，也可作为相关技术人员的自学参考书。

在作为教材使用时，应注意以下几个问题。

1．一般来说，应该在学习了一门高级语言程序设计的基础上，再进一步学习面向对象程序设计。如果先修的这门高级语言是 C 语言，则本教材的前 5 章可按复习的方式快速通过。第 6，7 章所讨论的虽然也是 C 语言已有的内容，但涉及到 C/C++语言的难点和重点，最好按教材的体系再细致地捋一遍，为后面各章的学习打好坚实的基础。如果先修的是 PASCAL 语言，在前 7 章的学习过程中应注意对比这两种语言对同一事物的不同表达方式，充分利用从 PASCAL 语言的学习中已经获得的知识和经验。总之，教师可根据实际情况和自己的经验，灵活地使用本教材。

2．应充分利用每章后面的习题，其中的部分程序设计题可安排为上机实习。上机实习是学好程序设计的重要环节，应安排足够的时间（例如，不少于 40 学时）。

3．C++Builder 是本书推荐的上机实习编程工具。当然，也可以使用 Visual C++，但该编程工具直到版本 6.0 为止尚未采用 C++语言的一些新的标准，用它来编译运行第 11、12 章的某些例题和习题可能会出现问题。本书的例题和习题中的程序都是在 DOS 环境下运行的，不涉及 Windows 环境下的编程问题。

4．本书的主题是“面向对象程序设计”，而不是“C++Builder 程序设计”或“Visual C++程序设计”，全面掌握 C++Builder 或 Visual C++这类可视化开发工具并不是本教

材的目标，完成本教材的学习只需利用其中最基本的功能。教师可以把 Windows 环境下的编程问题补充到课程中，但应注意，由于挤占学时和过早地把学生的注意力吸引到如何设计菜单、按钮、对话框等问题上，有可能冲击对面向对象程序设计本质内容的学习。

由于学识水平和时间的限制，不妥之处在所难免，欢迎批评指正。

李　宁

2002 年 1 月

目　录

第 1 章　C++与面向对象程序设计

1.1　面向对象程序设计的概念

面向对象方法是正在快速发展并成为主流的软件系统开发方法，包括面向对象的系统分析、面向对象的系统设计和面向对象的程序设计。系统分析、系统设计、程序设计是软件开发的三个基本步骤，但从面向对象方法的发展过程看，面向对象程序设计是其中最先发展起来的，在技术上也最为成熟。

面向对象程序设计是一种围绕真实世界的概念来组织模型的程序设计方法，它采用对象来描述问题空间的实体。对象是能体现现实世界物体的基本特征的抽象实体，反映在软件系统中就是一些属性和方法的封装体。从程序设计的角度看，对象就是“数据+作用于数据上的操作（方法）”。

类（class）是对同类型对象（具有相同的属性和方法的对象）集合的一种抽象。类规定了所属对象的共同属性和方法，就是一种抽象的数据类型；而对象则是类的实例。对象和类的关系类似于一般程序设计语言中变量和变量类型的关系。

面向对象方法有三个基本特征：

① 封装性

封装性是对象和类概念的主要特性。封装是把数据和作用于数据的方法结合在一起而形成一个整体，它隐藏了实现的细节，使得外部只能通过规定的接口访问对象中的数据。封装保证了模块具有较好的独立性，使得程序维护较为容易。

② 继承性

继承是一种联结类的层次模型，它提供了一种明确表述共性的方法。一个新类可以从现有的类派生，这个过程称为继承。新类继承了原始类的特性，称为原始类的派生类（子类），而原始类则称为新类的基类（父类）。类可以通过修改继承的方法和增加新类的方法使派生类更适合特殊的需要。基类—派生类的关系体现了客观世界中一般与特殊的关系。继承很好地解决了软件的重用问题。

③ 多态性

一个面向对象的系统常常要求一组具有相同表层语义的方法能在同一接口下为不同的对象服务，这就是所谓的多态性。多态性是现实生活中的一种常见现象，很容易找到有关例子。例如，任何一辆汽车都有方向盘、换挡操纵杆以及用于控制离合器、制动装置和油门的三个踏板，而且它们的相对位置关系是固定不变的。换句话说，不管汽车在内部构造和运行原理方面有多大的不同，操纵汽车的“接口”是不变的，因此只要学会驾驶一种汽车，就等于学会了开任何类型的汽车。再如，与媒体录放有关的设备，如录音机、录像放像机、VCD机、随身听等等，都具有一组相同或类似的控制键：▶键用来播放，●键用来录音或录像，■键用来停止录放等等。这些媒体录放设备的内部结构和实现原理有很大差别，但其操纵方法的基本语义却十分相近，因此可以用统一的接口来操纵它们。多态性在软件中也早已存在，

如在一般高级语言中，通过“+”操作，既可以完成两个整数相加，也可以完成两个实数的相加。再如，利用复制—粘贴操作，既可以完成一段文字数据的复制，也可完成一个图形的复制。多态性可大大简化系统的界面，使得不同的但又具有某种共同属性的对象不但能在一定程度上共享代码，而且还能共享接口。这就大大提高了系统的一致性、灵活性和可维护性。

在程序设计中采用面向对象方法的目的是使程序设计者能更好地理解和管理庞大而复杂的程序，它在结构化程序设计的基础上完成进一步的抽象，这种在设计方法上更高层次的抽象适应目前软件开发的特点。

使用面向对象的程序设计方法绝不是要摒弃结构化程序设计方法，相反，它充分吸收了结构化程序设计优点，并在此基础上引进了一系列新的、强有力的概念，从而开创了程序设计工作的新天地。面向对象的程序设计方法把可重复使用性视为软件开发的中心问题，通过装配可重用的部件来生产软件；但在对象内部的实现上，我们仍然要使用结构化程序设计方法，仍然要调用 C/C++函数库中的很多有用的函数。

面向对象的程序设计可提高系统开发的效率，增强系统的可靠性。由于面向对象编程的可重用性，可以在应用程序中大量采用成熟的类库，从而缩短开发时间。由于继承和封装使应用程序修改所带来的影响更加局部化，因而面向对象的程序设计可使所开发的应用系统更易于维护，更易于升级。

1.2 为什么要学习 C++

C++是由 C 语言扩充而成的，它在 C 的基础上增加了常值（const）数据、inline 函数、引用（reference）函数及操作符重载等新的成分以及新的更严格的类型检查机制，同时全面支持面向对象概念，如对象、类、属性和方法、派生类与继承等等。

C++的早期版本叫做“带类的 C”，是 1980 年由贝尔实验室发明的。发明 C++的重要目标就是在保留 C 原有精华的基础上提供全面的面向对象的编程支持，使得程序的结构更加清晰、更容易维护和扩充，同时又不丧失其高效性。从实际效果看，C++达到了它的目标。

C++并不是惟一的面向对象程序语言，但 C++较为全面地反映了面向对象方法中的各种概念，因此通过学习 C++语言，有利于全面、系统地掌握面向对象的程序设计方法。作为 C++前身的 C 已经是广为应用的语言，为很多人所掌握。这些人更具备了进一步学习 C++的有利条件。

C++也跟其前身 C 一样，已被广泛地应用。例如，Visual C++和 C++Builder 这两种可视化开发工具都是以 C++为脚本语言的。因此，学习和掌握 C++也具有十分现实的意义。

1.3 一个简单的 C++程序

例 1.1 就是一个典型的 C++程序。

例 1.1 清单（area_t.cpp）

```
1: //---------------------------------------------------------------
2: #pragma hdrstop
3: #include <condefs.h>
4: #include<iostream.h>
```

```
5: #define PI 3.1416
6: double area_of_round(double);
7:
8: //---------------------------------------------------------------
9: #pragma argsused
10: int main(int argc, char* argv[]){
11:  double radius,area;
12:  do{
13:    cout<<endl<<"radius=";
14:    cin>>radius;
15:    if(radius<0.0){
16:      cout<<endl<<"  error: radius can't be negative!";
17:      continue;
18:    }
19:    area=area_of_round(radius);
20:    cout<<endl<<"area="<<area<<" when radius="<<radius<<endl;
21:  }while(radius!=0);
22:  return 0;
23: }
24:
25: double area_of_round(double r){
26:  return r*r*PI;
27: }
```

本书中所有清单中的程序行号是为了便于说明而加上去的，不是程序内容的一部分。

1. 程序要做什么

程序从第 10 行（main 函数入口处）开始执行。程序的主体是一个 do…while 循环：首先从键盘输入圆的半径（第 13, 14 行），并判断输入的数据是否有误，如有误则返回到循环的开始处重新输入半径（第 15~18 行）；如输入的半径无误就调用 area_of_round 函数计算圆的面积，并将这个结果赋给变量 area（第 19 行）；最后显示输出 area 的值（第 20 行），这样就完成了一次循环。每循环一次就重复一次这样的计算圆面积的工作，直到输入的半径是 0 为止。

2. 字母大小写有区分意义

对于 C++语言，字母大小写是有区分意义的，因此 area, Area 和 AERA 是 3 个不同的标识符，不能相混，这一点与大多数计算机语言不同。

3. 常量和预处理命令#define

例 1.1 的头几行是几个预处理命令，#是预处理命令的标志。#define 是定义宏的预处理命令，常被用来定义符号常量。如例 1.1 的第 5 行，符号 PI 被定义为 3.141 6，这

样所有出现在程序中的 PI 在编译时都将被替换为 3.141 6，这比直接用 3.141 6 要好得多。用符号来代替常数可提高程序的可读性和可维护性。例如，通过测试发现圆周率 PI 取 3.141 6 精度不够，则不管程序中有多少处引用 PI，我们只需把这个#define 命令中的 3.141 6 更换成一个更精确的数值，比如 3.141 592 7，就可以了。

4. 基本语句与复合语句

像例 1.1 中的第 13, 14, 16, 17, 19, 20, 22, 26 等行中的语句都是基本语句，分别完成表达式计算（19, 26 行）、输入输出（13, 14, 16 行）和程序流程控制（17, 22, 26 行）等任务。每个基本语句都以分号（;）作为结束符，分界十分明确，因此一行中出现多个基本语句，或一个基本语句占用多行都是可以的。

复合语句是用{}括起来的语句序列。例 1.1 中有 4 个复合语句：作为函数定义中的函数体的两个复合语句（10~23 行和 25~27 行）、作为 do 循环的循环体的复合语句（12~21 行）和作为 if 条件分支的分支程序的复合语句（15~18 行）。

5. 函数定义和函数原形

函数的定义性声明，简称函数定义，是程序文件的主要内容。例 1.1 中从第 10 行开始和从第 25 行开始的两段程序就是对 main 和 area_of_round 这两个函数的定义性声明，或者说是函数 main 和函数 area_of_round 的定义。函数定义的简单格式（常见格式，不一定完整严格，以下提到的简单格式都是这个含义）如下：

类型修饰符 *函数名*(*形式参数表*) *函数体*

格式中带下划线的斜体字表示在实际程序中要用实际内容（实际的类型修饰符、实际的函数名等）去替代。

常见函数的定义由*类型修饰符*、*函数名*、*形式参数表*、*函数体*四部分组成。*形式参数表*必须用圆括号括起来，当表中有多个参数时，参数之间要用逗号隔开。空的参数表可表示为“()”。*函数体*是一个复合语句，即一个用“{}”括起来的语句序列。

函数的参考性声明，主要是指函数原形（prototype），即不含*函数体*的函数声明，其简单格式是：

类型修饰符 *函数名*(*形式参数表*);

如果被调用的函数的定义位于调用处的后面（如例中的 area_of_round），则必须以函数原形的形式提前进行参考性声明。例 1.1 中，由于对函数 area_of_round 第一次调用在第 19 行，而它的定义却在第 25 行，位于调用处之后，因此在第 6 行用函数原形的形式提前对 area_of_round 进行了参考性声明。

6. 头文件和预处理命令#include

例 1.1 中的第 13, 14, 16, 20 行用 C++流完成对终端的输入、输出，其中的>>和<<是两个作为操作符的特殊函数。还有一个称为操纵符（manipulator）的东西：endl，是一种特殊的函数。这些函数虽然是系统提供的，但也必须遵循先声明后使用的原则，而它们的参考性声明（原形）都在系统提供的头文件中。例 1.1 的第 4 行用#include 命令把头文件 iostream.h 插入到该#include 命令处，实际上也就是把>>，<<，endl 等的声明插入到程序的开始处。

一般来说，如果程序文件中引用到的函数、变量、常量、数据类型等是由别的文件提供的，就必须在程序文件的开始部分用#include命令把有关的头文件包含进来。所谓“有关的头文件”，就是指含有那些函数、变量、常量、数据类型等声明的头文件。头文件以 h（header 的字头）为扩展名。

7. 主函数 main()：应用程序的入口

构成 C++应用系统的程序文件可以是一个，也可以是多个，但其中必有一个程序文件包含一个名为 main 的特殊函数，称为主函数。它是整个系统的入口，应用系统将从该函数开始运行。由于没有任何其他函数要调用主函数，因此无须对之进行参考性声明。

对于 Windows 应用系统，主函数的名字是 WinMain。

8. 变量及变量的数据类型

变量的定义性声明，简称变量定义，如例 1.1 中第 11 行。变量定义可采用下面的简单格式：

类型修饰符 变量名表;

其中的*类型修饰符*主要包括 char, int, long, float, double 等，分别表示字符型、整型、长整型、单精度和双精度。例如，

```
int a,b,xy;
```

就定义了三个整型变量：a, b 和 xy。

如果程序中使用的变量是在同一程序文件中定义的，则必须定义在前、使用在后；如果是在另一文件中，则必须在使用处之前用保留字 extern 将该变量声明为外部，例如

```
extern double var;
```

就告诉编译系统：本文件中使用的双精度变量 var 是在另一个文件中定义的。

9. 赋值

像很多程序设计语言一样，C++也是用操作符“=”构成赋值语句的。例如，例 1.1 中的第 19 行的：

```
area=area_of_round(radius);
```

就是一个赋值语句。

C++的赋值语句的一般格式是：

变量=表达式;

与其他语言不同的是，在 C++中，*变量=表达式*本身也是一个表达式，称为赋值表达式；加上语句结束符（即分号）后才成为赋值语句。表达式都是有值的，赋值表达式也不例外：它的值就是等号左边的*变量*所获得的值，例如 z=5 的值就是 5。赋值表达式的值当然也可以赋给一个变量，因此语句：

```
x=y=z=5;
```

在 C++中是合法的。它首先完成对 z 的赋值，使 z 获得值 5，这个 5 也同时就是表达式 z=5 的值；随后完成对 y 的赋值，赋给 y 的值就是等号右边的表达式 z=5 的值，即 5，这个 5 也同时就是表达式 y=z=5 的值；最后完成对 x 的赋值，赋给 x 的值就是等号右边的表达式 y=z=5 的值，即 5，这个 5 也同时就是表达式 x=y=z=5 的值，但被舍弃不用。简而言之，