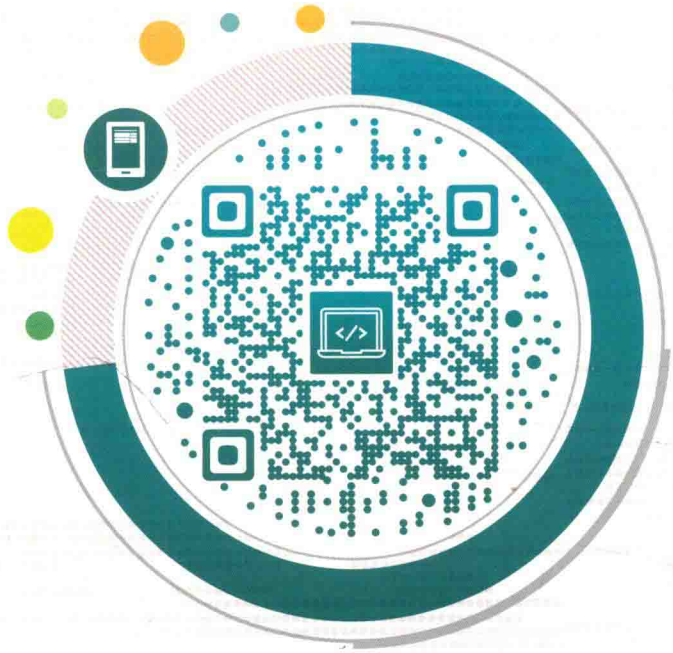
21世纪高等学校计算机类课程创新规划教材 · 微课版



# 面向对象程序设计教程（C++语言描述）（第3版）

微课版

◎ 马石安 魏文平 编著

900分钟 视频讲解

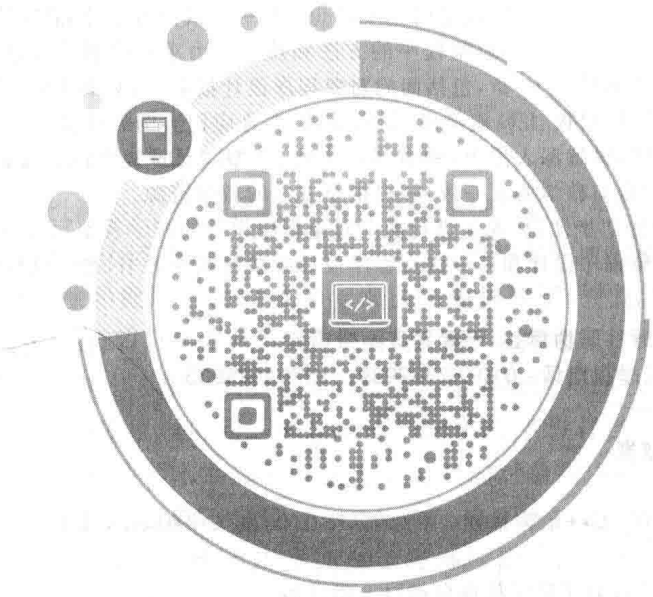
104个 实例

269道 练习题

10个 上机实验

清华大学出版社

21世纪高等学校计算机类课程创新规划教材 · 微课版



# 面向对象程序设计教程

## （C++语言描述）（第3版）

微课版

◎ 马石安 魏文平 编著

清华大学出版社

北京

## 内 容 简 介

本书以面向对象程序设计(Object-Oriented Programming,OOP)方法为核心,并选用C++语言作为工具。

本书浓缩了作者多年来软件开发经验和教学实践体会,围绕两条主线进行编写:一条主线以通俗易懂的语言围绕类与对象,介绍面向对象程序构造的基本思想;另一主线设计了丰富的实用程序,通过实践引导读者快速掌握使用C++语言开发面向对象程序的方法和技巧。力求使读者不仅会使用C++语言编程,而且可以理解这些机制。本书共分10章,包括面向对象程序设计概论、从C到C++、类与对象、继承机制、多态性和虚函数、运算符重载、模板、I/O流类库、异常处理、综合应用实例等内容。

本书内容安排循序渐进,讲解深入浅出,列举实例丰富、典型。每章提供的二维码可观看相应章节的视频讲解,练习题和实验内容与教学要求一致,并提供全方位的教学资源。

本书是为已有C语言的初步知识,准备进行面向对象程序设计的初学者编写的,可作为高等院校计算机及相关专业学习面向对象程序设计和C++语言程序设计的教材或参考书,也可供自学者使用。



**图书在版编目(CIP)数据**

面向对象程序设计教程:C++语言描述:微课版/马石安,魏文平编著. —3版. —北京:清华大学出版社,2018

(21世纪高等学校计算机类课程创新规划教材·微课版)

ISBN 978-7-302-51062-8

Ⅰ. ①面… Ⅱ. ①马… ②魏… Ⅲ. ①C++语言—程序设计—教材 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2018)第192417号

**责任编辑**:魏江江
**封面设计**:刘　键
**责任校对**:胡伟民
**责任印制**:董　瑾

**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 装 者**:北京国马印刷厂
**经　　销**:全国新华书店
**开　　本**:185mm×260mm　　**印　　张**:19.75　　**字　　数**:481千字
**版　　次**:2007年8月第1版　2018年10月第3版　　**印　　次**:2018年10月第1次印刷
**印　　数**:27501～29000
**定　　价**:49.50元

产品编号:077671-01

# 前言

自从第一台计算机诞生以来，程序设计方法与程序设计语言不断发展。面向对象的程序设计使计算机解决问题的方式更符合人类的思维方式，更能直接地描述客观世界，通过增加代码的可重用性、可扩充性和程序自动生成功能来提高编程效率，并且大大减少软件维护的开销，从而被越来越多的软件设计人员所接受。“面向对象”不再是软件开发中的一个时髦名词，而是对软件开发人员的基本要求。面向对象程序设计已经成为程序设计领域的主流技术。

目前，在教学实践中还很难找到一本合适面向对象程序设计的入门教材能够兼顾到理论应用和编程实践。我们编写本书的目的是为了给面向对象程序设计初学者提供一本清晰的入门教材，该教材以面向对象程序设计（Object-Oriented Programming，OOP）方法为核心，并选用 C++语言作为工具。本书围绕两条主线进行编写：一条主线以通俗易懂的语言围绕类与对象，介绍面向对象程序构造的基本思想；另一条主线设计了丰富的实用程序，通过实践引导学生快速掌握使用 C++语言开发面向对象程序的方法和技巧。

本书浓缩了作者多年来软件开发和教学实践的经验和体会，通过多次讲授面向对象程序设计，作者能够深刻理解面向对象程序设计编程的基本学习要求，与其他面向对象程序设计教材相比，本书有以下特色：

（1）以循序渐进、深入浅出的方式引导读者学习面向对象程序设计的基本思想。

本书在章节的安排上是由易到难。在讲解每章的过程中，尽量用一个实例，从满足基本要求开始，一步一步融入新的思想和方法。每章最后设计了一个应用实例，围绕一个专用系统来开发，重点对本章内容进行综合运用，同时与前面章节相呼应。

为了突出教学重点，本书实例中没有用到 C++语言的复杂结构，这样既使程序具有可读性，又避免了喧宾夺主。

（2）以面向对象程序设计方法为核心，以 C++语言为工具。

面向对象程序设计作为一种程序设计方法，应该是独立于程序设计语言的。本书在讲解面向对象程序设计的每一个新机制时，首先介绍为什么要引入这些机制，然后说明这些机制在 C++内部是如何实现的。我们力求使读者不仅学会使用，而且可以理解这些机制。只有这样读者才可能很容易地转向其他程序设计语言。

当然，在面向对象程序设计语言环境中进行程序设计，可以使面向对象思想得到更好的支持。所以，在学习面向对象程序设计的过程中，掌握程序设计语言的特征固然是重要的，但掌握面向对象程序设计思想却是更本质的要求。

（3）不需要先有扎实的 C 语言基础。

一是 C++语言对 C 语言最主要的扩充是引入了面向对象的概念及相应的处理机制。本书第 2 章介绍了 C++语言的新特性，且重点介绍了它在后续章节中要用到的部分。二是

没有设计复杂的算法,这与本书的教学目标是一致的。

(4) 类是构造面向对象程序的基本单元。

时下流行的一个观点是,学习C++应该先从类学起。从第3章开始,书中的实例程序基本上都是由主函数加上类组成的,类是构造面向对象程序的基本单元。这样有助于初学者采用面向对象思维方式而不是传统结构化的思维方式来解决实际问题,有助于构造良好的程序结构,为日后处理大型程序打好基础。

(5) 每个关键概念都配以完整的C++测试实例。

本书针对所讲述的知识点提供便于理解的实例,避免枯燥无味的讲解,给读者以直观的感受。每章后面提供一个综合实例,如此环环紧扣,帮助读者完成从了解、熟练到深入理解的学习过程。为了确保正确性,每个实例均已在Visual C++ 6.0环境下调试通过。

(6) 每章后面配有与教学要求一致的练习题。

每章后面的练习题内容全面,形式多样。包括问答题、选择题、判断题、分析程序输出结果题和编程题等。通过这些练习题,读者可以及时地检查和考核对本章内容学习和掌握的情况,教师也可以从中选出一些题作为作业题。

(7) 附录配有与教学要求一致的实验内容。

安排并指导学生上机实习,对学好本课程具有重要意义。对初学者来说,理解面向对象程序设计的基本思想需要一个循序渐进的过程。所以本书提供的实验内容既有验证性的,也有应用性的。每个实验中除了给出实验目的、实验内容外,还要求学生结合实验结果进行分析和讨论。

(8) 每章提供了教学视频。

为了让读者更轻松地完成本书的学习,我们精心制作了20小时的微课教学视频,全程语音讲解,让读者一学就会。

为方便教师教学和学生学习,我们还编写了配套的教学用书《面向对象程序设计(C++语言描述)题解及课程设计指导》,并提供书中所有源代码和课堂教学的课件等资源,构成一个完整的教学系列。

本书第3章~第10章由马石安编写,第1章~第2章以及附录由魏文平编写,全书由马石安统一修改、整理和定稿。

在编写过程中,本书参考和引用了大量书籍和文献资料,在此,向被引用文献的作者及给予本书帮助的所有人士表示衷心感谢,尤其感谢江汉大学领导和同事以及清华大学出版社领导和编辑的大力支持与帮助。

由于作者水平有限,加之时间仓促,书中难免存在缺点与疏漏之处,敬请读者及同行予以批评指正。

本书介绍

编　者

2018年3月

# 目 录

# 第1章 面向对象程序设计概论

自从第一台计算机诞生以来，程序设计方法与程序设计语言不断发展。面向对象的程序设计使计算机解决问题的方式更符合人类的思维方式，更能直接地描述客观世界，通过增加代码的可重用性、可扩充性和程序自动生成功能来提高编程效率，并且大大减少软件维护的开销，已经被越来越多的软件设计人员所接受。

本章首先介绍程序设计方法，重点比较面向对象程序设计方法与面向过程的结构化程序设计方法的区别。然后介绍面向对象程序设计的基本概念和目前流行的几种面向对象程序设计语言。最后强调C++语言对面向对象程序设计方法的支持。

## 1.1 程序设计方法


视频讲解

用计算机语言编写程序，解决某种问题，称为程序设计。程序设计需要一定的方法来指导，以便提高程序的扩充性、重用性、可读性、稳定性及编程效率。目前有两种重要的程序设计方法：面向过程的结构化程序设计方法和面向对象程序设计方法。

### 1.1.1 结构化程序设计方法

结构化程序设计(structured programming)的概念是由瑞士计算机科学家Niklaus Wirth于1971年首次提出来的，随之也出现了支持结构化程序设计方法的程序设计语言，例如Pascal、C、Ada等。程序设计语言开始向模块化、形式化方向发展。基于这些程序设计语言的支持，结构化程序设计方法逐渐成为了20世纪70～80年代十分流行的程序设计方法。

程序由模块(module)构成。结构化程序设计方法是面向过程的，一个模块就是一个过程。每一模块均是由顺序、选择和循环这3种基本结构组成的。模块之间的信息传递主要通过模块的接口(interface)来实现，这一机制隐藏了模块的内部细节，增强了模块的独立性。

结构化程序设计方法强调程序结构的规范性，强调程序设计的自顶向下、逐步求精的演化过程。在这种方法中，待解问题和程序设计语言中的子过程紧密相连。例如要开发一个成绩管理系统，由于问题较复杂，可以将待解的问题分解成若干子问题：

- 输入成绩。
- 处理成绩。
- 打印成绩。

每个子问题对应程序设计语言中的一个子过程。如果用C语言或当作过程语言使用的C++来解决上述问题，则待解问题将对应main()，每个子问题对应main()的调用函数。

当然每个子问题还可以继续分解,直到每个子问题都足够简单,相应的子过程也很容易处理(参见第2章的应用实例)。

可见,这种方法着眼于系统要实现的功能,从系统的输入和输出出发,分析系统要做哪些事情,以及如何做这些事情,自顶向下地对系统的功能进行分解,建立系统的功能结构和相应的程序模块结构,有效地将一个较复杂的程序系统设计任务分解成许多易于控制和处理的子任务,便于开发和维护。

到今天,结构化程序设计已无处不在,几乎每种程序设计语言都具备支持结构化程序设计的机制。然而,随着程序规模的扩大与复杂性的增加,这种面向过程的结构化程序设计方法已体现出明显的不足之处。首先是数据安全性问题,例如在上述成绩管理系统中,成绩数据被每个模块所共用,因此是不安全的,一旦出错,很难查明原因。其次是可维护性及可重用性差。它把数据结构和算法分离为相互独立的实体,一旦数据结构需要改变时,常常要涉及整个程序,修改工作量极大并容易产生新的错误。每一种针对老问题的新方法都要带来额外的开销。另外,图形用户界面的应用程序很难用过程来描述和实现,而且开发和维护也都很困难。一个好的软件应该随时响应用户的任何操作,而不是让用户严格按照既定的步骤使用。于是面向对象程序设计方法便应运而生了。

### 1.1.2 面向对象程序设计方法

面向对象程序设计方法建立在结构化程序设计方法基础上,完全避免了结构化程序设计方法中所存在的问题。

在结构化程序设计方法中,程序可表示为:

```
程序 = 数据结构 + 算法
```

即程序的要素是数据结构和算法。数据结构是指利用计算机的离散逻辑来量化表达需要解决的问题,而算法则研究如何高效而快捷地组织解决问题的具体过程。两者都是一个个独立的整体,互相之间没有必然的联系。

**【例1.1】** 用C++语言描述,用结构化程序设计方法计算矩形的面积。

```
// 程序 Li1_1.cpp
#include<iostream>
using namespace std;
int main()
{
    float length,width,area;              // 定义变量长、宽、面积
    cout<<"please input length and width"<<endl;
    cin>>length>>width;                   // 输入长、宽值
    area=length*width;                    // 计算面积
    cout<<area<<endl;                     // 输出面积
    return 0;
}
```

在该程序中的任何地方都可以对数据length、width、area进行访问。由此可见,在结构化程序设计方法中,数据结构和算法是分离的,数据结构属于整个程序,而且程序是从开始

至结束顺序执行的。

客观世界是由各种各样的对象组成的，对象可以是有形的，如汽车、房子，也可以是无形的，如火车的运行情况、学习计划等，但都具有属性和行为。与人们认识客观世界的规律一样，在面向对象的程序设计方法中，将程序设计为一组相互协作的对象(object)而不是一组相互协作的函数。在程序中，属性用数据表示，用来描述对象的静态特征；行为用程序代码实现，用来描述对象的动态特征。可见，在面向对象的程序设计方法中，对象是数据结构和算法的封装体。

根据这个定义，对象是计算机内存中的一块区域。在对象中，不但存有数据，而且存有代码，这使得每个对象在功能上相互之间保持相对独立。当然，对象之间存在各种联系，但它们之间只能通过消息进行通信。在面向对象程序设计方法中，程序可表示为：

程序 = 对象 + 消息

在高级程序设计语言中，一般用类来实现对象。类(class)是具有相同属性和行为的一组对象的集合，它是创建对象的模板。实际上，类是面向对象程序的唯一构造单位，面向对象程序看上去就是由一些类组成的。例如，计算矩形的面积，如果用面向对象程序设计方法，可以先定义矩形类，将数据 length、width 及 area 和对它们的操作封装在一起，再创建一个个相应的矩形对象，然后通过对对象执行相应的操作来实现程序的功能。例 1.2 给出了这个功能的实现。

**【例 1.2】** 用 C++语言描述，用面向对象程序设计方法计算矩形的面积。

```
// 程序 Li1_2.cpp
// 定义矩形类
#include <iostream>
using namespace std;
// 类的声明
class RectangleArea
{
    public:
        void SetData(float L,float W);          // 输入长、宽值
        float ComputeArea();                    // 计算面积
        void OutputArea();                      // 输出面积
    private:
        float length,width,area;                // 定义长、宽、面积
};
// 类的实现
void RectangleArea::SetData(float L,float W)
{
    length = L;
    width = W;
}
float RectangleArea::ComputeArea()
{
```

```
    area = length * width;
    return area;
}
void RectangleArea::OutputArea()
{
    cout <<"area = "<< area << endl;
}
// 主函数
int main()
{
    RectangleArea Rectangle;        // 声明对象
    Rectangle.SetData(8,9);
    Rectangle.ComputeArea();
    Rectangle.OutputArea();
    return 0;
}
```

当然,我们现在还无法完全理解这个程序,但通过这个程序可以知道面向对象程序的基本结构。一般情况下,面向对象程序由3个部分构成:类的声明、类的成员的实现和主函数。

可见,面向对象程序设计着重于类的设计。类正是面向对象语言的基本程序模块,通过类的设计来完成实体的建模任务。如学籍管理系统,我们需要设计这样一些类:Teacher和Student,这些类分别与教师、学生实体相关联。每个类必须对与其相关的实体进行数据和操作的定义,类通过一个简单的外部接口,与外界发生关系。一个类中的操作不会影响到另一个类中的数据,这样程序模块的独立性、数据的安全性就有了良好的保障。程序的执行取决于事件发生的顺序,由顺序产生的消息来驱动程序的执行。不必预先确定消息产生的顺序,更符合客观世界的实际。

程序Li1_2比程序Li1_1看起来要烦琐一些。但是,如果以RectangleArea类为基础,通过继承,可以很方便地派生出长方体等新的几何体,从而实现代码重用。因为所举例子比较小,所以这3个部分都写在同一个文件中。在规模较大的项目中,往往需要多个源程序文件,每个源程序文件称为一个编译单元。这时,C++语法要求一个类的声明必须出现在所有使用该类的编译单元中。比较好的也是惯用的做法是将类的声明写在头文件中,使用该类的编译单元则包含这个头文件。通常一个项目至少划分为3个文件:类声明文件(*.h文件)、类实现文件(*.cpp文件)和类的使用文件(*.cpp,主函数文件)。对于更为复杂的程序,每一个类都有单独的声明和实现文件。采用这样的组织结构,可以对不同的文件进行单独编写、编译,最后再连接,同时可充分利用类的封装特性,在调试、修改程序时只对其中某一个类的声明和实现进行修改,而其余部分不用改动。这些都是结构化程序设计方法所做不到的。

软件工程专家Peter Codd和Edward Yourdon给面向对象下了一个简明的等式描述:

面向对象 = 对象 + 类 + 继承 + 消息 + 多态

也就是说,面向对象就是既使用对象又使用类、继承和多态等机制,而且对象之间通过消息

的传递实现通信。如果一个软件系统使用了这几个概念来设计，则可以说该软件系统是面向对象的。

面向对象程序设计方法提供了软件重用、解决大问题和复杂问题的有效途径，具有抽象性、封装性、继承性和多态性等特点，已经成为近年来主流的程序设计方法。

## 1.2 面向对象程序设计的基本概念

视频讲解

在1.1.2小节中涉及面向对象程序设计的许多概念，本节先了解这些基本概念，在后续章节中再详细讨论它们，以进一步加深对这些概念的理解和运用。

### 1.2.1 抽象

把客观世界的众多事物归纳、分类是人类在认识客观世界时经常采用的思维方法，分类所依据的原则就是抽象。抽象(abstract)就是忽略事物中与当前目标无关的非本质特征，而强调与当前目标有关的本质特征，从而找出事物的共性，并把具有共性的事物划为一类，得到一个抽象的概念。

所有的程序设计语言都提供抽象。面向对象方法中的抽象，是指对具体问题(对象)进行概括，找出一类对象的公共性质并加以描述的过程。它往往包括两个方面：数据抽象和行为抽象(或称为功能抽象、代码抽象)。其中，数据抽象描述某类对象共有的属性或状态，行为抽象描述某类对象共有的行为或功能特征。将这两方面抽象有机地结合，就形成了面向对象程序设计中的“对象”。还可以继续抽象：把众多相似的“对象”聚集起来，进一步抽象后就形成了“类”。

下面来分析程序清单Li1_2：通过对矩形图形的简单抽象分析，发现每一个矩形都具有一些相同的特征，比如都有长、宽和面积等数据，这就是对矩形进行数据抽象；另外矩形要具有设置长、宽数据和计算、显示面积等功能，这就是对矩形进行行为抽象。

数据抽象：

```
float length,width,area;
```

行为抽象：

```
SetData(float L,float W);
ComputeArea();
OutputArea();
```

如果不是计算矩形的面积，人们关注的特征可能是颜色、大小等。由此可见，对于同一个研究对象，由于所研究问题的侧重点不同，就可能产生不同的抽象结果。即使对于同一个问题，解决问题的要求不同，也可能产生不同的抽象结果。进一步对一个个矩形对象抽象就形成一个矩形类RectangleArea。

虽然只对某个研究对象(如矩形)进行抽象后即可得到一个对象(如矩形对象)，但为了代码重用，人们设计类而不是设计对象，类只需编码一次，就可以创建本类的所有对象。

抽象是面向对象程序设计的一个基本特征，类就是对一些问题和概念进行抽象的工具，类从客观世界的一组事物中抽取其共同的属性和行为，对象则是类的实例化、具体化。