普通高等教育“十一五”国家级规划教材
北京高等教育精品教材

配套用书

C++面向对象程序设计教程（第4版）

习题解答与上机指导

陈维兴 林小茶 陈昕 编著

清华大学出版社

C++
面向对象程序设计教程（第4版）习题解答与上机指导

陈维兴 林小茶 陈昕 编著

清华大学出版社
北京

内 容 简 介

本书是《C++面向对象程序设计教程(第4版)》(陈维兴、林小茶编著,清华大学出版社2018年出版)的配套用书。书中内容分为两部分:第1部分是《C++面向对象程序设计教程(第4版)》习题与参考解答,给出了教材中所有习题的参考答案;第2部分是C++上机实验指导,详细介绍了C++上机操作方法,包括对Visual C++ 6.0和Visual C++ 2010两个环境的简单介绍,同时精心设计了与教材内容配套的7组实验题,每组实验题目都包括"实验目的和要求""实验内容和步骤",供上机实验参考。在本书的最后一章给出了各组上机实验题的参考解答,供读者参考和借鉴,以帮助读者更好地掌握C++面向对象程序设计的基本概念和编程方法。

本书可作为学习《C++面向对象程序设计教程(第4版)》的辅助教材,也可供学习C++的其他读者参考。

图书在版编目(CIP)数据

C++面向对象程序设计教程(第4版)习题解答与上机指导/陈维兴,林小茶,陈昕编著.—4版.—北京:清华大学出版社,2018(2019.2重印)

ISBN 978-7-302-50370-5

Ⅰ.①C… Ⅱ.①陈… ②林… ③陈… Ⅲ.①C++语言—程序设计—高等学校—教学参考资料 Ⅳ.①TP312.8

中国版本图书馆CIP数据核字(2018)第118075号

责任编辑:柳　萍
封面设计:何凤霞
责任校对:赵丽敏
责任印制:丛怀宇

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
印 装 者:三河市铭诚印务有限公司
经　　销:全国新华书店
开　　本:185mm×260mm　　**印　　张**:12　　**字　　数**:291千字
版　　次:2003年5月第1版　2018年10月第4版　　**印　　次**:2019年2月第2次印刷
定　　价:32.00元

产品编号:077008-01

前　言

学过程序设计的人，都有一个体会，看别人编写的程序，好像挺明白的，但是一旦要自己编写一个程序，就感觉无从下手。这是因为程序设计是一门对实践环节要求很高的课程，初学者要想真正学会 C++ 面向对象程序设计，最重要的是抓住两个关键环节：一个是多做习题多编程；另一个就是多上机，写在纸上的程序是否正确，最好的办法就是上机验证。为此，我们编写了这本习题解答与上机指导书，以期帮助读者尽快地掌握 C++ 语言程序设计的基本规则与编程技巧，并能够熟练运用这些规则与技巧，编制出具有良好风格的应用程序，最终能够顺利地通过上机调试。

本书的主要内容分为两部分：第 1 部分是《C++ 面向对象程序设计教程(第 4 版)》(陈维兴、林小茶编著，清华大学出版社 2018 年出版)习题与参考解答，详细解答了教材中的所有习题；第 2 部分是 C++ 上机实验指导，详细介绍了 C++ 上机操作方法，并精心设计了与教材内容配套的 7 组实验题，每组实验题目都包括“实验目的和要求”“实验内容和步骤”，供上机实验时参考。在本书的最后一章给出了各组上机实验题的参考解答，帮助初学者掌握实验内容和理解具体实现步骤，以更好地掌握 C++ 面向对象程序设计的基本概念和编程方法。

提供习题参考解答和实验题参考解答的主要目的是供读者参考和借鉴，作者在这里要强调一点，程序设计是创作的过程，解决一个实际问题的程序肯定不是唯一的，因此，在阅读本书的参考解答之前，希望读者已经独立思考过教材中的习题及实验题目，这样才有助于程序设计水平的提高，不要把本书的参考解答作为唯一的答案。本书中所有程序都经作者在 Visual C++ 6.0 及 Visual C++ 2010 上调试通过(注意，在两个环境下调试会有一点区别，请读者参考主教材的相关内容)。

本书内容是作者多年教学实践的总结，虽然得到了读者的肯定，但由于编者水平有限，错误和不当之处在所难免，在此恳请广大读者批评指正。

编　者

2018 年 4 月

目　录

第 1 部分　《C ++ 面向对象程序设计教程(第 4 版)》习题与参考解答

第 2 部分　C ++ 上机实验指导

第1部分

《C++面向对象程序设计教程(第4版)》习题与参考解答

第1部分

《C++面向对象程序设计教程（第4版）》习题与参考答案

第 1 章　面向对象程序设计概述

【1.1】　什么是面向对象程序设计？

【解】　面向对象程序设计是一种新的程序设计范型。这种范型的主要特征是：

程序 = 对象 + 消息

面向对象程序的基本元素是对象，面向对象程序的主要结构特点是：第一，程序一般由类的定义和类的使用两部分组成；第二，程序中的一切操作都是通过向对象发送消息来实现的，对象接收到消息后，启动有关方法完成相应的操作。

面向对象程序设计方法模拟人类习惯的解题方法，代表了计算机程序设计新颖的思维方式。这种方法的提出是对软件开发方法的一场革命，是目前解决软件开发面临困难的最有希望、最有前途的方法之一。

【1.2】　什么是对象？什么是类？对象与类的关系是什么？

【解】　在现实世界中，任何事物都是对象。它可以是一个有形的具体存在的事物，例如一张桌子、一个学生、一辆汽车，甚至一个地球；它也可以是一个无形的、抽象的事件，例如一次演出、一场球赛、一次出差等。对象既可以很简单，也可以很复杂，复杂的对象可以由若干简单的对象构成，整个世界都可以认为是一个非常复杂的对象。在现实世界中，对象一般可以表示为：属性+行为，一个对象往往是由一组属性和一组行为构成的。

在面向对象程序设计中，对象是描述其属性的数据以及对这些数据施加的一组操作封装在一起构成的统一体。在C++ 中每个对象都是由数据和操作代码（通常用函数来实现）两部分组成的。

在现实世界中，“类”是一组具有相同属性和行为的对象的抽象。类和对象之间的关系是抽象和具体的关系。类是对多个对象进行综合抽象的结果，对象又是类的个体实物，一个对象是类的一个实例。

在面向对象程序设计中，“类”就是具有相同的数据和相同的操作（函数）的一组对象的集合，也就是说，类是对具有相同数据结构和相同操作的一类对象的描述。

类和对象之间的关系是抽象和具体的关系。类是多个对象进行综合抽象的结果，一个对象是类的一个实例。例如“学生”是一个类，它是由许多具体的学生抽象而来的一般概念。同理，桌子、教师、计算机等都是类。

【1.3】　现实世界中的对象有哪些特征？请举例说明。

【解】　现实世界中的对象，具有以下特性：

（1）每一个对象必须有一个名字以区别于其他对象；

（2）用属性来描述它的某些特征；

（3）有一组操作，每组操作决定对象的一种行为；

（4）对象的行为可以分为两类：一类是作用于自身的行为；另一类是作用于其他对象的行为。

例如，雇员刘明是一个对象。

对象名：

刘明

对象的属性：

年龄：36

生日：1966.10.30

工资：20000

部门：人事部

对象的操作：

吃饭

开车

【1.4】 什么是消息？消息具有什么性质？

【解】 在面向对象程序设计中，一个对象向另一个对象发出的请求被称为“消息”。当对象接收到发向它的消息时，就调用有关的方法，执行相应的操作。例如，有一个教师对象张三和一个学生对象李四，对象李四可以发出消息，请求对象张三演示一个实验，当对象张三接收到这个消息后，确定应完成的操作并执行之。

一般情况下，我们称发送消息的对象为发送者或请求者，接收消息的对象为接收者或目标对象。对象中的联系只能通过消息传递来进行。接收对象只有在接收到消息时，才能被激活，被激活的对象会根据消息的要求完成相应的功能。

消息具有以下三个性质：

(1) 同一个对象可以接收不同形式的多个消息，作出不同的响应；

(2) 相同形式的消息可以传递给不同的对象，所作出的响应可以是不同的；

(3) 对消息的响应并不是必需的，对象可以响应消息，也可以不响应。

【1.5】 什么是抽象和封装？请举例说明。

【解】 抽象是将有关事物的共性归纳、集中的过程。抽象是对复杂世界的简单表示，抽象并不打算了解全部问题，而只强调感兴趣的信息，忽略了与主题无关的信息。例如，在设计一个成绩管理程序的过程中，只关心学生的姓名、学号、成绩等，而对他的身高、体重等信息就可以忽略。而在学生健康信息管理系统中，身高、体重等信息必须抽象出来，而成绩则可以忽略。

抽象是通过特定的实例(对象)抽取共同性质后形成概念的过程。面向对象程序设计中的抽象包括两个方面：数据抽象和代码抽象(或称为行为抽象)。前者描述某类对象的属性或状态，也就是此类对象区别于彼类对象的特征物理量；后者描述了某类对象的共同行为特征或具有的共同功能。

在现实世界中，所谓封装就是把某个事物包围起来，使外界不知道该事物的具体内容。在面向对象程序设计中，封装是指把数据和实现操作的代码集中起来放在对象内部，并尽可能隐蔽对象的内部细节。

下面以一台洗衣机为例，说明对象的封装特征。首先，每一台洗衣机有一些区别于其他洗衣机的静态属性，例如出厂日期、机器编号等。另外，洗衣机上有一些按键，如“启动”“暂

停”“选择”等，当人们使用洗衣机时，只要根据需要按下“选择(洗衣的方式)”“启动”或“暂停”等按键，洗衣机就会完成相应的工作。这些按键安装在洗衣机的表面，人们通过它们与洗衣机交流，告诉洗衣机应该做什么。我们无法(当然也没必要)操作洗衣机的内部电路和机械控制部件，因为它们被装在洗衣机里面，这对于用户来说是隐蔽的，不可见的。

【1.6】 什么是继承？请举例说明。

【解】 继承所表达的是类之间的相关关系，这种关系使得某类对象可以继承另外一类对象的特征和能力。现实生活中，继承是很普遍和容易理解的。例如我们继承了我们父母的一些特征，如种族、血型、眼睛的颜色等，父母是我们所具有的属性的基础。

图 1.1 所示是一个继承的典型例子：汽车继承的层次。

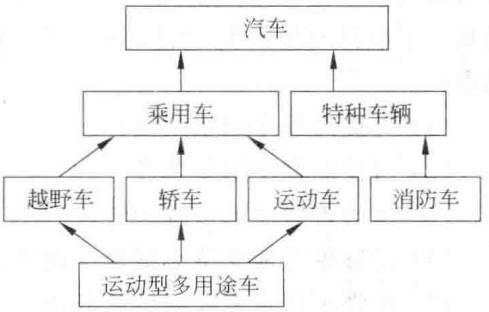

图 1.1

以面向对象程序设计的观点，继承所表达的是类之间相关的关系。这种关系使得某一类可以继承另外一个类的特征和能力。

【1.7】 若类之间具有继承关系，则它们之间具有什么特征？

【解】 若类之间具有继承关系，则它们之间具有下列几个特性：

(1) 类间具有共享特征(包括数据和操作代码的共享)；

(2) 类间具有差别或新增部分(包括非共享的数据和操作代码)；

(3) 类间具有层次结构。

假设有两个类 A 和 B，若类 B 继承类 A，则类 B 包含了类 A 的特征(包括数据和操作)，同时也可以加入自己所特有的新特性。这时，我们称被继承类 A 为基类或父类；而称继承类 B 为类 A 的派生类或子类。同时，我们还可以说，类 B 是从类 A 中派生出来的。

【1.8】 什么是单继承、多继承？请举例说明。

【解】 从继承源上分，继承分为单继承和多继承。

单继承是指每个派生类只直接继承了一个基类的特征。图 1.2 表示了一种单继承关系。它表示 Windows 操作系统的窗口之间的继承关系。

多继承是指多个基类派生出一个派生类的继承关系。多继承的派生类直接继承了不止一个基类的特征。例如，小孩喜欢的玩具车即继承了车的一些特征，还继承了玩具的一些特征。如图 1.3 所示。

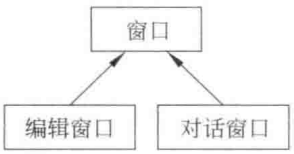

图 1.2

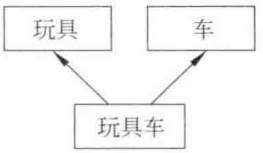

图 1.3

【1.9】 什么是多态性？请举例说明。

【解】 面向对象系统的多态性是指不同的对象收到相同的消息时执行不同的操作。例

如，有一个窗口(Window)类对象，还有一个棋子(Piece)类对象，当我们对它们发出“移动”的消息时，“移动”操作在 Window 类对象和 Piece 类对象上可以有不同的行为。

C++ 语言支持两种多态性，即编译时的多态性和运行时的多态性。编译时的多态性是通过函数重载(包括运算符重载)来实现的，运行时的多态性是通过虚函数来实现的。

【1.10】 面向对象程序设计的主要优点是什么？

【解】 面向对象程序设计本质上改变了人们以往设计软件的思维方式，从而使程序设计者摆脱了具体的数据格式和过程的束缚，将精力集中于要处理对象的设计和研究上，极大地减少了软件开发的复杂性，提高了软件开发的效率。面向对象程序设计主要具有以下优点：

(1) 可提高程序的重用性；

(2) 可控制程序的复杂性；

(3) 可改善程序的可维护性；

(4) 能够更好地支持大型程序设计；

(5) 增强了计算机处理信息的范围；

(6) 能很好地适应新的硬件环境。

面向对象程序设计是目前解决软件开发面临难题的最有希望、最有前途的方法之一。

第2章　C++概述

【2.1】 简述C++的主要特点。

【解】 C++语言的主要特点表现在两个方面，一是全面兼容C，并对C的功能作了不少扩充，二是增加了面向对象的机制，具体表现为：

(1) C++是C的超集，C++保持与C的兼容，这就使许多C代码不经修改就可以为C++所用，用C编写的众多的库函数和实用软件可以用于C++中。

(2) C++是一个更好的C，它保持了C的简洁、高效和接近汇编语言等特点，并对C的功能作了不少扩充。用C++编写的程序比C更安全，可读性更好，代码结构更为合理，C++的编译系统能够检查出更多的类型错误。

(3) 用C++编写的程序质量高，从开发时间、费用到形成的软件的可重用性、可扩充性、可维护性和可靠性等方面有了很大的提高，使得大中型的程序开发变得更加容易。

(4) 增加了面向对象的机制，几乎支持所有的面向对象程序设计特征，体现了近20年来在程序设计和软件开发领域出现的新思想和新技术，这主要包括：

① 抽象数据类型；

② 封装与信息隐藏；

③ 以继承方式实现程序的重用；

④ 以函数重载、运算符重载和虚函数来实现多态性；

⑤ 以模板来实现类型的参数化。

C++语言最有意义的方面是支持面向对象的特征，然而，由于C++与C保持兼容，使得C++不是一个纯正的面向对象的语言，C++既可用于面向过程的结构化程序设计，也可用于面向对象的程序设计。

【2.2】 下面是一个C程序，改写它，使它采用C++风格的I/O语句。

```
#include<stdio.h>
int main()
{ int a,b,d,min;
  printf("Enter two numbers:");
  scanf("%d%d",&a,&b);
  min=a>b?b:a;
  for(d=2; d<min; d++)
    if(((a%d)==0)&&((b%d)==0))break;
  if(d==min)
  { printf("No common denominators\n");
```

```
    return 0;
  }
  printf("The lowest common denominator is %d\n",d);
  return 0;
}
```

【解】 修改后的程序如下：

```
#include<iostream>
using namespace std;
int main()
{ int a,b,d,min;
  cout<<"Enter two numbers:";
  cin>>a;
  cin>>b;
  min=a>b?b:a;
  for(d=2; d<min; d++)
    if(((a%d)==0)&&((b%d)==0))break;
  if(d==min)
  { cout<<"No common denominators\n";
    return 0;
  }
  cout<<"The lowest common denominator is"<<endl<<d;
  return 0;
}
```

【2.3】 测试下面的注释(它在C++ 风格的单行注释中套入了类似于 C 的注释)是否有效：

```
//this is a strange /* way to do a comment */
```

【解】 此注释有效，单行注释中可以嵌套/*……*/方式的注释。

【2.4】 以下这个简短的C++ 程序不可能编译通过，为什么？

```
#include<iostream>
using namespace std;
int main()
{ int a,b,c;
  cout<<"Enter two numbers: ";
  cin>>a>> b;
  c=sum(a,b);
  cout<<"sum is:"<<c;
  return 0;
}
int sum(int a,int b)
{ return a+b;
}
```

【解】 不可能通过编译的原因是：在程序中，当一个函数的定义在后，而对它的调用在前时，必须将该函数的原型写在调用语句之前，而在本程序中缺少函数原型语句。在语句“using namespace std;”后加上函数原型语句“int sum(int a,int b);”就可通过编译。

【2.5】 回答问题。

(1) 以下两个函数原型是否等价：

```
float fun(int a,float b,char * c);
float fun(int,float,char * );
```

(2) 以下两个函数的第 1 行是否等价：

```
float fun(int a,float b,char * c)
float fun(int,float,char * )
```

【解】

(1) 这两个函数原型是等价的，因为函数原型中的参数名可以缺省。

(2) 这两个函数的第 1 行是不等价的，因为这个函数的第 1 行中必须包含参数名。

【2.6】 下列语句中错误的是(　　)。

A. int * p=new int(10);　　B. int * p=new int[10];

C. int * p=new int;　　D. int * p=new int[40](0);

【解】 D

说明：“int * p=new int(10);”表示动态分配 1 个整型内存空间，初值为 10；

“int * p=new int[10];”表示动态分配 10 个整型内存空间；

“int * p=new int;”表示动态分配 1 个整型内存空间；

“int * p=new int[40](0)”想给一个数组分配内存空间时，对数组进行初始化，这是不允许的。

【2.7】 假设已经有定义“const char * const name="chen";”下面的语句中正确的是(　　)。

A. name[3]='a';　　B. name="lin";

C. name=new char[5];　　D. cout<<name[3];

【解】 D

说明：name 被定义为指向常量的常指针，所以它所指的内容和本身的内容都不能修改，而“name[3]='a';”修改了 name 所指的常量，“name="lin";”和“name=new char[5];”修改了常指针，只有 D 输出一个字符是正确的。

【2.8】 假设已经有定义“char * const name="chen";”下面的语句中正确的是(　　)。

A. name[3]='q';　　B. name="lin";

C. name=new char[5];　　D. name=new char('q');

【解】 A

说明：name 被定义常指针，所以它所指的内容能改变，但指针本身的内容不可以修改，“name[3]='q';”修改了 name 所指的内容，是正确的。而“name="lin";”“name=new char[5];”和“name=new char('q');”以不同的方法修改了常指针，都是错误的。

【2.9】 假设已经有定义“const char * name="chen";”，下面的语句中错误的是(　　)。

A. name[3]='q';　　B. name="lin";

C. name=new char[5];　　D. name=new char('q');

【解】 A

说明：name被定义指向常量的指针，所以它所指的内容不能改变，但指针本身的内容可以修改，而“name[3]='q';”修改了name所指的内容，是错误的。“name=="lin";”“name=new char[5];”和“name=new char('q')”以不同的方法修改了常指针，都是正确的。

【2.10】 重载函数在调用时选择的依据中，(　　)是错误的。

A. 函数名字　　B. 函数的返回类型

C. 参数个数　　D. 参数的类型

【解】 B

【2.11】 在(　　)情况下适宜采用内联函数。

A. 函数代码小，频繁调用　　B. 函数代码多，频繁调用

C. 函数体含有递归语句　　D. 函数体含有循环语句

【解】 A

【2.12】 下列描述中，(　　)是错误的。

A. 内联函数主要解决程序的运行效率问题

B. 内联函数的定义必须出现在内联函数第一次被调用之前

C. 内联函数中可以包括各种语句

D. 对内联函数不可以进行异常接口声明

【解】 C

【2.13】 在C++中，关于下列设置默认参数值的描述中，(　　)是正确的。

A. 不允许设置默认参数值

B. 在指定了默认值的参数右边，不能出现没有指定默认值的参数

C. 只能在函数的定义性声明中指定参数的默认值

D. 设置默认参数值时，必须全部都设置

【解】 B

【2.14】 下面的类型声明中正确的是(　　)。

A. int & a[4];　　B. int & * p;

C. int && q;　　D. int i, * p=&i;

【解】 D

说明：C++中不能建立引用数组和指向引用的指针，也不能建立引用的引用。所以A、B、C是错误的，D是正确的。

【2.15】 下面有关重载函数的说法中正确的是(　　)。

A. 重载函数必须具有不同的返回值类型　　B. 重载函数形参个数必须不同

C. 重载函数必须有不同的形参列表　　D. 重载函数名可以不同

【解】 C

【2.16】 关于 new 运算符的下列描述中,(　　)是错误的。

A. 它可以用来动态创建对象和对象数组

B. 使用它创建的对象或对象数组可以使用运算符 delete 删除

C. 使用它创建对象时要调用构造函数

D. 使用它创建对象数组时必须指定初始值

【解】 D

【2.17】 关于 delete 运算符的下列描述中,(　　)是错误的。

A. 它必须用于 new 返回的指针

B. 使用它删除对象时要调用析构函数

C. 对一个指针可以使用多次该运算符

D. 指针名前只有一对方括号符号,不管所删除数组的维数

【解】 C

【2.18】 写出下列程序的运行结果。

```
#include<iostream>
using namespace std;
int i=15;
int main()
{ int i;
  i=100;
  ::i=i+1;
  cout<<::i<<endl;
  return 0;
}
```

【解】 本程序的运行结果如下:

```
101
```

说明:在语句"::i=i+1;"中赋值号左边"::i"的中 i 是全局变量,赋值号右边的 i 是局部变量。所以执行该语句的结果是将局部变量 i 的值加 1(即 101)后赋值给全局变量 i。

【2.19】 写出下列程序的运行结果。

```
#include<iostream>
using namespace std;
void f(int &m,int n)
{ int temp;
  temp=m;
  m=n;
  n=temp;
}
int main()
{ int a=5,b=10;
  f(a,b);
  cout<<a<<" "<<b<<endl;
  return 0;
}
```