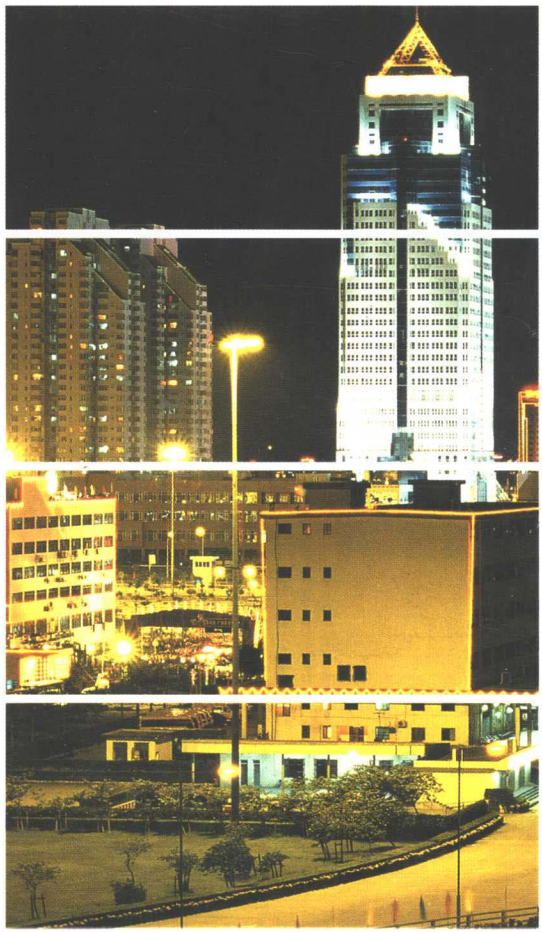
21世纪高校计算机应用技术系列规划教材
谭浩强 主编

# C++面向对象程序设计习题解答与上机指导

林小茶 陈维兴 编著

★本书是《C++面向对象程序设计》的配套教材，
也可单独作为学习C++语言的辅导书。
★本书给出了主教材中所有习题的参考答案，
以及每个实验题的上机步骤和参考程序，
供教师和学生参考。
★本书以应用为目的，
注重培养应用能力。
★适合于高校学生学习C++程序设计课程的教材，
也可作为C++语言自学者的参考书。





中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE


谭浩强　主编

21世纪高校计算机应用技术系列规划教材

# C++面向对象程序设计习题解答与上机指导

林小茶　陈维兴　编著

中国铁道出版社

2004·北京

## 内 容 简 介

本书是《C++面向对象程序设计》的配套教材，书中给出了主教材中所有习题的参考答案以及每个实验题的上机步骤和参考程序，供教师和学生参考。本书内容主要分为3篇，分别为《C++面向对象程序设计》习题和参考答案； C++语言上机实验环境介绍；上机实验题与参考答案。另外书后附录给出了常用的错误信息注释，以供学生上机实验时使用。

本书可作为高校学生学习C++程序设计课程的辅导教材，也可作为C++语言自学者的参考书。

**图书在版编目（CIP）数据**

C++面向对象程序设计习题解答与上机指导/林小茶，陈维兴编著. －北京：中国铁道出版社，2004.4
（21世纪高校计算机应用技术系列规划教材）
ISBN 7－113－05899－X
Ⅰ.C… Ⅱ.①林…②陈… Ⅲ.C语言-程序设计-高等学校－教学参考资料 Ⅳ.TP312

中国版本图书馆CIP数据核字(2004)第033132号

**书　　名**：C++面向对象程序设计习题解答与上机指导
**作　　者**：林小茶　陈维兴
**出版发行**：中国铁道出版社（100054，北京市宣武区右安门西街8号）
**策划编辑**：严晓舟　魏　春
**责任编辑**：苏　茜　黄园园　秦绪好
**封面设计**：白　雪
**印　　刷**：河北省遵化市胶印厂
**开　　本**：787×1092　1/16　印张：13　字数：312千
**版　　本**：2004年5月第1版　　2004年5月第1次印刷
**印　　数**：1～5000册
**书　　号**：ISBN 7-113-05899-X/TP・1201
**定　　价**：18.00元

# 21世纪高校计算机应用技术系列规划教材

# 编委会名单

# 丛书序言

21世纪是信息技术高度发展并且得到广泛应用的时代，信息技术深刻地改变了人类的生活、工作和思维方式。每一个人都应当学习信息技术、应用信息技术。人们平常习惯说的计算机教育其内涵实际上已经发展为信息技术教育，内容主要包括计算机和网络的基本知识和应用。

对多数人来说，学习计算机的目的是为了利用计算机这个现代化工具去处理工作和面临的各种问题，使自己能够跟上时代前进的步伐，同时要在学习的过程中努力培养自己的信息素养，使自己具有信息时代所要求的科学素质，站在信息技术发展和应用的前列，推动我国信息技术的发展。

学习计算机课程，有两种不同的方法，一是从理论入手；一是从实际应用入手。不同的人有不同的学习内容和学习方法。大学生中的多数人将来是各行各业中的计算机应用人才。对他们来说，不仅需要解决**知道什么**，更重要的是**会做什么**。因此要以应用为目的，注重培养应用能力，大力加强实践环节，激励创新意识。

根据实际教学的需要，我们组织编写这套“**21世纪高校计算机应用技术系列规划教材**”。顾名思义，这套丛书的特点是突出应用技术，面向实际应用。在选材上，根据实际应用的需要决定内容的取舍，坚决舍弃那些现在用不到、将来也用不到的内容。在叙述方法上，采取**“提出问题——介绍解决问题的方法——归纳结论和概念”**的三部曲，这种从实际到理论、从具体到抽象、从个别到一般的方法，符合人们的认识规律，实践证明已取得了很好的效果。

本丛书采取模块化的结构，根据需要确定一批书目，也就是提供一个课程菜单供各校选用，以后根据信息技术的发展和教学的需要，不断地补充和调整。只要教学有需要，我们就组织编写新的教材，不受任何框框的限制。我们的指导思想是面向实际，面向应用，面向对象。这样比较灵活，能满足不同学校、不同专业的需要。希望各校的老师把你们的要求反映给我们，我们将会尽最大努力满足大家的要求。

本丛书可以作为大学计算机应用技术课程教材以及高职高专、成人高校和面向社会的培训班的教材，也可作为学习计算机的自学教材。

参加本丛书策划和编写工作的专家和老师有：谭浩强、陈维兴、严晓舟、薛淑斌、秦建中、安淑芝、安志远、赵乃真、李雁翎、宋红、周永恒、熊伟建、宋金珂、陈元春、冯继生、姚怡、沈洪、沈添、李尊朝、王晓敏、侯冬梅、訾秀玲、魏善沛、孙中胜、王内义、程爱民、史秀璋、李振银、刘涛、李宁等。此外参加本丛书编辑和其他工作的还有：魏春、秦绪好、张艳芳、戴薇、郭晓溪、马建、姜淑静、杨东晓、于静等。对于他们的智慧、奉献和劳动表示深切的谢意。中国铁道出版社以很高的热情和效率组织了丛书的出版工作。在组织编写出版的过程中，得到全国高等院校计算机基础教育研究会和各高等院校老师的热情鼓励和支持，对此谨表衷心的感谢。

本丛书如有不足之处，请各位专家、老师和广大读者不吝指正。

谭浩强谨识

2003年2月于清华园

# 前　言

学过程序设计的人，都有一个体会，看别人编写的程序，好象挺明白的，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++语言程序设计，最重要的要抓住两个关键环节：一个是多做程序设计的习题，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一下。为此，我们编写了这本习题解答与实验指导。本书在对教材中的习题进行解答的同时，也对一些基本的程序算法和规则进行了详细的分析，希望能帮助学习者尽快掌握C++语言程序设计的基本规则与编程规律，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3篇：第一篇“《C++面向对象程序设计》习题和参考答案”是对教材中的习题的详细解答；第二篇“C++语言上机实验环境介绍”介绍了C++程序设计调试环境Visual C++ 6.0和Turbo C++；第三篇“上机实验题与参考答案”安排了10套精心设计的实验，每个实验都给出了详细的实验目的、实验要求和实验步骤，帮助学习者掌握C++程序设计的方法，并进一步加深对课程相关内容的理解与掌握。考虑到学生在上机操作中，经常遇到一些错误信息，但看不懂，不知如何修改，因此我们把一些常用的错误信息注释放在附录中，以供学生上机实验时使用。

提供习题解答和实验参考源程序的主要目的是给学习者一个参考和借鉴，作者在这里要强调一点，程序设计是一个创作的过程，解决一个实际问题的程序肯定不是惟一的，因此，在阅读本书的参考答案之前，希望读者已经独立思考过教材中的习题以及实验题目，这样才有助于程序设计水平的提高，并且不要把本书的参考源程序作为唯一的答案。本书中所有程序都经作者在Visual C++6.0或Turbo C++3.0上调试通过。

本书第一篇的内容由陈维兴教授和林小茶副教授共同编写，附录由陈维兴编写，第二篇和第三篇由林小茶编写。

在本书编写和出版过程中，全国计算机基础教育研究会理事长谭浩强教授给予了指导和把关，在此表示最衷心的感谢。

由于水平有限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

编　者

2004年4月

# 目　录

## 第一篇　《C++面向对象程序设计》习题和参考答案



## 第二篇　C++语言上机实验环境介绍



## 第三篇　上机实验题与参考答案

# 第一篇 《C++面向对象程序设计》习题和参考答案

## 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型的程序设计范型。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，在主程序中定义各对象并规定它们之间传递消息的规律。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】 什么是类?什么是对象？对象与类的关系是什么？

【解】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。对象可以认为是：数据+操作。

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。

在面向对象程序设计中，总是先声明类，再由类生成其对象。类是建立对象的“模板”，按照这个模板所建立的一个个具体的对象，就是类的实际例子，通常称为实例。

【1-3】 现实世界中的对象有哪些特征？请举例说明。

【解】对象是现实世界中的一个实体，其具有以下一些特征：

（1）每一个对象必须有一个名字以区别于其他对象。

（2）需要用属性来描述它的某些特征。

（3）有一组操作，每一个操作决定了对象的一种行为。

（4）对象的操作可以分为两类：一类是自身所承受的操作，一类是施加于其他对象的操作。

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。消息是一个对象要求另一个对象执行某个操作的规格的说明，通过消息传递才能完成对象之间的相互请求或相互协作。

消息具有以下 3 个性质：

（1）同一个对象可以接收不同形式的多个消息，做出不同的响应。

（2）相同形式的消息可以传递给不同的对象，所做出的响应可以是不同的。

（3）消息的发送可以不考虑具体的接收者，对象可以响应消息，也可以不响应。

**【1-5】** 什么是方法？消息和方法的关系是什么？

**【解】**在面向对象程序设计中，要求某一对象作某一操作时，就向该对象发送一个相应的消息，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。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。方法的界面也就是消息的模式，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。在 C++ 语言中方法是通过函数来实现的，称为成员函数。消息和方法的关系是：对象根据接收到的消息，调用相应的方法；反过来，有了方法，对象才能响应相应的消息。

**【1-6】** 什么是封装和抽象？请举例说明。

**【解】**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。对象好像是一个不透明的黑盒子，表示对象属性的数据和实现各个操作的代码都被封装在黑盒子里，从外面是看不见的，更不能从外面直接访问或修改这些数据及代码。使用一个对象的时候，只需知道它向外界提供的接口形式而无需知道它的数据结构细节和实现操作的算法。封装机制可以将对象的使用者与设计者分开，使用者不必知道对象行为实现的细节，只需要使用设计者提供的接口让对象去做。

抽象是人类认识问题的最基本的手段之一。它忽略了一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象是对复杂世界的简单表示，抽象强调感兴趣的信息，忽略了不重要的信息。例如，在设计一个学籍管理程序的过程中，考察某个学生对象时，只关心他的姓名、学号、成绩等，而对他的身高、体重等信息就可以

忽略。

以一般观点而言，抽象是通过特定的实例（对象）抽取共同性质以后形成概念的过程。抽象是对系统的简化描述或规范说明，它强调了系统中的一部分细节和特性，而忽略了其他部分。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况，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操作。

在面向对象程序设计方法中，对一个具体问题的抽象分析的结果，是通过类来描述和实现的。

现在以学生管理程序为例，通过对学生进行归纳、分析，抽取出其中的共性，可以得到如下的抽象描述：

共同的属性：姓名、学号、成绩等，它们组成了学生数据抽象部分。用 C++语言的数据成员来表示，可以是：

```
char* name; int number; float score;
```

共同的行为：数据录入、数据修改和数据输出等，这构成了学生的行为抽象部分，用 C++语言的成员函数表示，可以是：

```
input(); modify(); print();
```

如果我们开发一个学生健康档案程序，所关心的特征就有所不同了。可见，即使对同一个研究对象，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。

**【1-7】** 什么是继承性？请举例说明。

**【解】** 继承所表达的是对象类之间的相关关系，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。

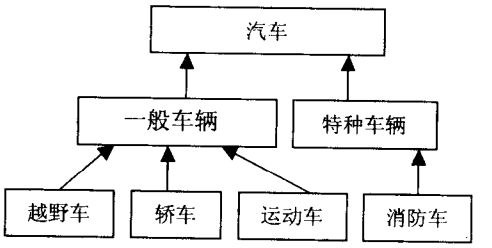



图 1-1　汽车继承的层次图

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

**【1-8】** 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 为基类或父类或超类；而称继承类 B 为 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。

**【1-9】** 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，小孩喜欢的玩具车既继承了车的一些特性，还继承了玩具的一些特征。如图 1-3 所示。

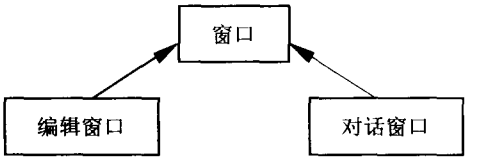



图 1-2　单继承关系


图 1-3　多继承关系

**【1-10】** 什么是多态性？请举例说明。

**【解】**多态性也是面向对象程序的重要特征。它是指不同的对象收到相同的消息时产生不同的行为方式。例如我们同样双击 Windows 系统桌面上的图标时，有的是打开多媒体播放器，有的是打开资源管理器。

利用多态性，用户只需发送一般形式的消息，而将所有的实现留给接收消息的对象。对象根据所收到的消息做出相应的动作。

**【1-11】** 什么是函数重载和运算符重载？为什么要使用重载？

**【解】**重载一般包括函数重载和运算符重载。函数重载是指一个标识符可同时用于为多个函数命名，而运算符重载是指一个运算符可同时用于多种运算。也就是说，相同名字的函数或运算符在不同的场合可以表现出不同的行为。

为什么要使用重载？使用重载的目的是为了更好地表达行为共享，这种行为共享就像将相似的操作划分在一起。使用重载可以使程序员在只知道操作的一般含义，而不知道操作的具体细节的情况下能正确地对某个对象使用一个操作。

另外，使用重载的直接益处是减少了程序员记忆操作的名字的负担。

**【1-12】** 传统程序设计方法的局限性主要有哪些?

**【解】**传统程序设计方法的局限性主要有 3 个方面：

（1）传统程序设计开发软件的生产效率低下

传统程序设计的生产方式仍是采用较原始的方式进行，程序设计基本上还是从语句一级开始。软件的生产中缺乏大粒度、可重用的构件，软件的重用问题没有得到很好地解决，从而导致软件生产的工程化和自动化屡屡受阻。传统程序设计的特点是数据与其操作分离，而且对同一数据的操作往往分散在程序的不同地方。这样，如果一个或多个数据的结构发生了变化，那么这种变化将波及程序的很多部分甚至遍及整个程序，致使许多函数和过程必须重写，严重时会导致整个软件结构的崩溃。传统程序设计是面向过程的，其数据和操作相分离的结构，使得维护数据和处理数据的操作过程要化费大量的精力和时间，严重地影响了软件的生产效率。

总之，要提高软件生产的效率，就必须很好地解决软件的重用性、复杂性和可维护性问题。但是传统的程序设计是难以解决这些问题的。

（2）传统程序设计难以应付日益庞大的信息量和多样的信息类型

当代计算机的应用领域已从数值计算扩展到了人类社会的各个方面，所处理的数据已从简单数字和字符，发展为具有多种格式的多媒体数据，如文本、图形、图像、影像、声音等，

描述的问题从单纯的计算问题到仿真复杂的自然现象和社会现象。随着计算机处理的信息量与信息类型的迅速增加，程序的规模日益庞大，复杂度不断增加，传统程序设计方法是无法应付的。

（3）传统的程序设计难以适应各种新环境

当前，并行处理、分布式、网络和多机系统等，已经或将是程序运行的主流方式和主流环境。这些环境的一个共同的特点是都具有一些有独立处理能力的节点，节点之间有通讯机制，即以消息传递进行联络。显然传统的程序设计技术很难适应这些新环境。

**【1-13】** 面向对象程序设计的优点主要有哪些?

**【解】**面向对象程序设计的优点主要包括以下几个方面：

（1）可提高程序的重用性

重用是提高软件开发效率的最主要的方法，面向对象程序设计能比较好地解决软件重用的问题。对象所固有的封装性和信息隐藏等机理，使得对象内部的实现与外界隔离，具有较强的独立性，它可以作为一个大粒度的程序构件，供同类程序直接使用。

（2）可控制程序的复杂性

面向对象程序设计采用了数据抽象和信息隐藏技术，把数据及对数据的操作放在一个个类中，作为相互依存、不可分割的整体来处理。这样，在程序中任何要访问这些数据的地方都只需简单地通过传递信息和调用方法来进行，这就有效地控制了程序的复杂性。

（3）改善程序的可维护性

在面向对象程序设计中，对对象的操作只能通过消息传递来实现，所以只要消息模式即对应的方法界面不变，方法体的任何修改不会导致发送消息的程序修改，这显然给程序的维护带来了方便。另外，类的封装和信息隐藏机制使得外界对其中的数据和程序代码的非法操作成为不可能，这也就大大地减少了程序的错误率。

（4）能够更好地支持大型程序设计

面向对象技术在数据抽象和抽象数据类型之上又引入了动态连接和继承性等机制，进一步发展了基于数据抽象的模块化设计，使其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

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的新颖的思维方法。用类来直接描述现实世界中的类型，可使计算机系统的描述和处理对象从数据扩展到现实世界和思维世界的各种事物，这实际上大大扩展了计算机系统处理的信息量和信息类型。

（6）很好地适应新的硬件环境

面向对象程序设计中的对象、消息传递等思想和机制，与分布式、并行处理、多机系统及网络等硬件环境也恰好相吻合。面向对象的思想也影响到计算机硬件的体系结构，现在已在研究直接支持对象概念的面向对象计算机。这样的计算机将会更适合于面向对象程序设计，更充分地发挥面向对象技术的威力。

# 第2章 C++基础

【2-1】 简述C++的主要特点。

【解】C++的主要特点有：

（1）C++保持与C的兼容，用C编写的软件可以用到C++中。

（2）用C++编写的程序可读性好，代码结构更合理，可直接地在程序中映射问题空间的结构。

（3）生成代码的质量高。

（4）软件的可重用性、可扩充性、可维护性和可靠性有了明显的提高，从而节省了开发费用和时间。

（5）支持面向对象的机制，可方便地构造出模拟现实问题的实体和操作。

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

【解】

```
#include <iostream.h>
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";
        retrun 0;
    }
    cout<<"The lowest common denominator is "<<endl<<d;
    retrun 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.h>
main()
{
    int a,b,c;
    cout <<"Enter two numbers:  ";
    cin >>a>>b;
    c=sum(a,b);
    cout <<"sum is:  "<<c;
    return 0;
}
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“#include<iostream.h>”后加上语句“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 *c);
```

（2）以下两个函数的第一行是否等价：

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

（3）以下两个函数原型是否等价：

```
int fun();
int fun(void);
```

【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）这两个函数的第一行是不等价的，函数的第一行中必须包含参数名。

（3）这两个函数原型是等价的，因为在函数原型中未注明参数，C++认为该函数的参数表为空（void）。

【2-6】 分析下面程序的输出结果：

```
#include <iostream.h>
int &f(int &i)
{
    i+=10;
    return i;
}
void main()
{
    int k=0;
    int &m=f(k);
    cout<<k<<endl;
    m=20;
    cout<<k<<endl;
}
```

【解】

```
10
20
```

【说明】f 函数的参数是引用，所以修改 k 的值有效。函数调用后，主函数中 k 的值变为 10。由于 m 是对函数的引用，当 m 被赋值为 20 时，k 的值也变为 20 。

【2-7】 举例说明可以使用 const 替代#define 以消除#define 的不安全性。

【解】例如，以下程序显示出#define 的不安全性：

```
#include <iostream.h>
#define A 2+4
#define B A*3
void main()
{
    cout<<B<<endl;
}
```

上面程序的运行结果是 14，而不是 18，但很容易被认为是 18。下面程序使用 const 替代了#define ，就可以消除#define 的不安全性：

```
#include <iostream.h>
const A=2+4;
const B=A*3;
void main()
{
    cout<<B<<endl;
}
```

使用 const 以后，运行结果是 18。

【2-8】 使用内联函数的优点是什么？

【解】使用内联函数的优点主要有两个：一是能加快代码的执行，减少调用开销；二是能消除宏定义的不安全性。

【2-9】 用动态分配空间的方法计算 Fibonacci 数列的前 20 项并存储到动态分配的空间中。

【解】

```
#include <iostream.h>
#include "stdio.h"
void main()
{
    int i,*p=new int[20];   //动态分配 20 个整型空间
    *p=1;
    *(p+1)=1;               //前面两个空间赋值 1
    cout<<*p<<"\t"<<*(p+1)<<"\t";
    p=p+2;                  //p 指向第三个空间
    for (i=3;i<=20;i++)
    {
        *p=*(p-1)+*(p-2);
        cout<<*p<<"\t";
        if (i%5==0) cout<<endl;
        p++;                //p 指向下一个空间;
    }
}
```

本程序的运行结果是：

```
1       1       2       3       5
8       13      21      34      55
89      144     233     377     610
987     1597    2584    4181    6765
```

【2-10】 建立一个被称为 sroot()的函数，返回其参数的二次方根。重载 sroot()三次，让它返回整数、长整数与双精度数的二次方根（计算二次方根时，可以使用标准库函数 sqrt()）。

【解】

```
double sroot(int i)
{
    return sqrt(i);
}
double sroot(long l)
{
    return sqrt(l);
}
double sroot(double d)
{
    return sqrt(d);
}
```

【2-11】 编写 C++风格的程序，解决百钱问题：将一元人民币兑换成 1、2、5 分的硬币，有多少种换法？

【解】

```
#include <iostream.h>
void main()
{
    int i,j,sum=0;;
```

```
    for (i=0;i<=20;i++)
        for (j=0;j<=50;j++)
            if (100-5*i-2*j>=0)
            {
                sum++;
                cout<<100-5*i-2*j<<"\t"<<j<<"\t"<<i<<endl;
            }
    cout<<"sum is "<<sum<<endl;
}
```

本程序的运行结果是：

（541 种组合情况，结果略）

```
sum is 541
```

**【2-12】** 编写 C++风格的程序，用二分法求解 f(x)=0 的根。

**【解】**

```
#include <iostream.h>
#include <math.h>
inline float f(float x)
{
    return 2*x*x*x-4*x*x+3*x-6;
}
void main()
{
    float left,right,middle,ym,yl,yr;
    out<<"please two number: "<<endl;   //接收输入，确定第一组数据区域
    cin>>left>>right;                   //(left, right)
    yl=f(left);
    yr=f(right);
    do
    {
        middle=(right+left)/2;
    ym=f(middle);
        if (yr*ym>0)
        {
            right=middle;
            yr=ym;
        }
        else
        {
            left=middle;
            yl=ym;
        }
    } while (fabs(ym)>=1e-6);
    cout<<"\nRoot is : "<<middle;
}
```

**【说明】**本例使用了内联函数 f(x)，因为在主程序中多次调用了它，这样可以加快代码执行的速度。

本程序的运行结果是：