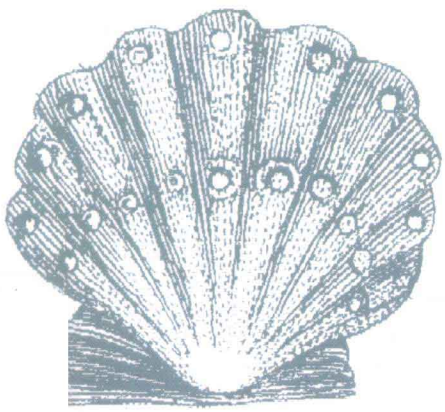
工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

面向对象程序设计及C++（第2版）

Object-Oriented Programming and C++ (2nd Edition)

朱立华 俞琼 主编
郭剑 朱建 副主编

- 强调C++实践技能，重点突出面向对象
- 经典例程贯穿始终，全面诠释编程方法
- 一题多解前后对比，深入启发读者探究



高校系列

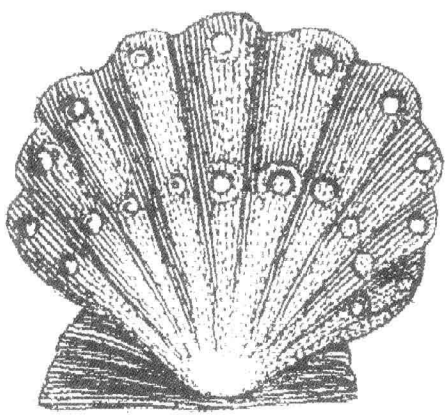
人民邮电出版社
POSTS & TELECOM PRESS

工
21
21s
教育“十二五”规划教材立项项目
划教材
nputer Science

面向对象程序设计及C++（第2版）

Object-Oriented Programming and C++ (2nd Edition)

朱立华 俞琼 主编
郭剑 朱建 副主编



高校系列

人民邮电出版社
·北京

图书在版编目（CIP）数据

面向对象程序设计及C++ / 朱立华，俞琼主编. — 2版. -- 北京 : 人民邮电出版社，2012.2
21世纪高等学校计算机规划教材
ISBN 978-7-115-26906-5

Ⅰ. ①面… Ⅱ. ①朱… ②俞… Ⅲ. ①面向对象语言－程序设计－高等学校－教材②C语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2011)第281725号

内 容 提 要

本书是为已经掌握C语言知识，需要学习C++语言的读者编写的一本C++语言入门教材。

全书共分9章。第1章通过与面向过程的程序设计方法的简单对比，初步介绍面向对象程序设计方法；第2章介绍在面向过程的程序设计方面C++语言对C语言的改进及扩充；第3章至第6章以面向对象程序设计的封装性、继承性和多态性这3大主要特征为主线组织内容，系统而全面地介绍了面向对象程序设计的基本概念和方法，是本书最核心的内容；第7章简单介绍模板的知识及STL初步；第8章介绍输入/输出控制及文件的读写；第9章通过一个综合实例初步介绍了面向对象的程序设计过程。

本书注重可读性、启发性和可用性。每章开头的内容提要简明扼要地对本章内容进行总体描述；在每章结尾有本章小结，对本章的主要内容作归纳总结；每章最后还配有一定数量的习题帮助读者巩固知识。每章通过大量典型的实例解析新的知识点。书中还通过大量图、表对知识点作总结或从本质上进行分析，便于读者记忆和理解。另外，本书还为授课教师提供配套的电子课件、全部例题源代码以及习题源代码。

本书可作为高等院校计算机相关专业程序设计课程的教材，也可作为工程技术人员的参考用书。

工业和信息化普通高等教育“十二五”规划教材立项项目

21世纪高等学校计算机规划教材

面向对象程序设计及C++（第2版）

◆ 主　　编　朱立华　俞　琼
副 主 编　郭　剑　朱　建
责任编辑　武恩玉

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京铭成印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：18.5　　2012年2月第2版
字数：491千字　　2012年2月北京第1次印刷

ISBN 978-7-115-26906-5

定价：38.00元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154
广告经营许可证：京崇工商广字第0021号

前　言

目前，许多高等院校在 C 语言课程结束之后开设了 C++语言课程，同时，也有很多读者希望掌握 C 语言之后继续学习 C++语言，尤其希望掌握 C++面向对象的程序设计，本书的编写及修订正是顺应了这些需求。本书作者长期从事计算机语言的教学，具有丰富的教学经验和独到的见解，这些经验和见解都已融入到本书的内容中。

C++语言是 C 语言的超集，既支持面向过程的程序设计，又支持面向对象的程序设计，但后者是其主要特色和应用。本书第 1 章比较了这两种不同的程序设计方法，通过浅显的实例使读者较容易地初步理解面向对象的相关概念。

为使读者能用 C++语言写出更好的结构化程序，本书第 2 章详细讲解了 C++语言在支持面向过程的程序设计方面对 C 语言的改进及扩充，便于读者使用 C++语言实现结构化程序设计时更好地发挥 C++语言的优势。

面向对象的程序设计是本书的重点。

本书的第 3 章至第 8 章围绕面向对象程序设计的概念与方法展开。第 3 章介绍了类、对象、构造函数与析构函数、友元等知识，这些知识是面向对象程序设计的基础。在此基础上，第 4 章对类中数据的共享与保护的实现措施和方法作了详细介绍，本章中的静态数据成员、静态成员函数、常数据成员、常成员函数以及常对象等知识都是在编程中经常要用到的。第 5 章从类与类之间的关系上进行分析，介绍了类的组合、依赖、继承与派生等关系，重点是类的继承与派生关系。派生类的定义、构造函数与析构函数的定义及调用顺序、二义性的消除、虚基类、赋值兼容等知识都是这一部分的重点。第 6 章介绍的多态性方便了用户编程，静态多态性与动态多态性的概念及实现方法是本章的重点，通过函数重载和运算符重载实现静态多态性，通过继承、虚函数、基类指针或引用实现动态多态性。第 7 章介绍了关于模板及 STL 编程的相关知识，C++支持的函数模板和类模板使得抽象可以在类型层次上完成，从而提供了一种更高层次的代码重用方法，C++语言提供的标准版模板库（Standard Template Library，STL）是面向对象程序设计与泛型程序设计思想相结合的一个良好典范。第 8 章详细介绍了 C++语言的流类库、格式控制方法、文件操作方法等非常实用的编程技术。第 9 章通过一个综合实例全面应用 C++面向对象编程的相关知识，帮助读者更好地理解面向对象编程。

与第 1 版相比，本版具有几个明显的特点。

（1）一例贯穿思想。体现在 3 处：第 3 章的各个小例题之间有前后相关性，用比较和层层深入的方式介绍新内容；第 3 章至第 8 章的综合实例是一例贯穿下来，但每章的侧重点不同，前后章中作一些对比；第 9 章专门介绍了关于面向对象编程的基本知识，并通过对前几章贯穿下来的实例的具体设计与实现体现面向对象编程的基本方法。这种做法改变了第 1 版教材中小例题无相关性，缺乏大例题综合运用相关知识的缺点。

（2）一半以上的小例题给出相应的思考题，启发读者进一步思考，对这些较深入问题的分析和解答放到实验教材的第 1 部分。这样做的目的便于分层次教学，主教材中强调的是最基本的知识，适合一般水平的读者，降低学习难度；而一些有一

定深度或不是常用的做法放到思考与练习中，方便水平较高的读者进一步深入学习。

（3）大段的总结和描述分散到各个例题中分开讲述，使知识点不至于太过密集，降低学习难度，读者读起来会比较轻松。

（4）章节的设置更为合理。本版增加了泛型程序设计及 STL 的知识，将对象成员的内容从第 1 版类与对象这一知识单元调整到类与类之间的关系中，并将第 1 版中类与对象知识单元中关于类内数据的共享与保护这一部分内容单独成一章介绍，使现有第 3 章内容比较集中体现类与对象最基本最核心的内容，再逐步深入，降低了学习难度。

本版继续保持了第 1 版中的一些做法和特点。

（1）每个新知识点的引出都以前面已有的知识作基础，并且结合实际例程阐述各要点，循序渐进地介绍新内容。

（2）每个实例程序的关键语句及运行结果后都有详细注解，帮助读者更好地理解。

（3）注重编程风格，源代码的书写格式规范。

（4）每章结束有本章小结帮助读者回顾本章内容，最后配有一定量的习题帮助读者巩固知识。

下表给出了学时安排建议，各院校可根据实际情况对教学内容合理取舍及分配学时。

章　节	讲授学时数	上机学时数
第 1 章	2	0
第 2 章	4	2
第 3 章	4	4
第 4 章	2	2
第 5 章	4	2
第 6 章	4	2
第 7 章	4	2
第 8 章	2	2
第 9 章	2	4
合计	28	20

本书的所有程序都已在 Visual C++6.0 集成开发环境下调试通过，电子教案在人民邮电出版社教学服务与资源网的网站（www.ptpedu.com.cn）下载区中，本书配套的教材《面向对象程序设计及 C++实验指导（第 2 版）》不仅包含实验部分，还包括了主教材思考与练习的解析、主教材课后习题解析与答案，并提供了每章补充习题及答案。

本书第 1 章、第 2 章、第 4 章由朱立华编写；第 3 章、第 7 章由俞琼编写；第 5 章、第 9 章由郭剑编写；第 6 章、第 8 章由朱建编写；全书由朱立华统稿。南京邮电大学计算机学院的张伟老师、程序设计教学课程组全体老师，以及西华大学陈红红老师等兄弟学校的同行们为教材的改版提出了很多宝贵的意见和建议，在此深表感谢。

由于编者水平有限，书中难免存在一些缺点和错误，恳请读者批评指正。

编　者

2011 年 10 月于南京邮电大学计算机学院

目 录

第 1 章 面向对象程序设计及 C++语言概述

内容提要：

本章通过与面向过程的程序设计方法的简单对比，向读者初步介绍面向对象程序设计方法。对其中的一些主要概念：类、对象、封装、继承、多态作简单介绍。

C++是众多支持面向对象程序设计的高级语言之一，它是在 C 语言的基础上发展起来的，既支持面向过程的程序设计又支持面向对象的程序设计。

本章将对 C++语言的基本特性作简单介绍，并以 Visual C++6.0 为集成开发环境介绍 C++程序的完整开发过程。

1.1 面向过程与面向对象

在使用面向对象程序设计方法之前，软件开发过程中人们广泛使用的是面向过程的程序设计方法。该方法以功能为基础，将数据与对数据的操作相分离，其优点是结构清晰，模块化强，但缺点是代码的可重用性差，不利于代码的维护和扩充。因此，面向过程的程序设计方法较适合于小型的程序和算法。

面向对象程序设计方法既吸取了面向过程程序设计方法的优点，又考虑了现实世界与面向对象空间的映射关系，该方法的提出和运用是软件开发史上的一个里程碑。面向对象程序设计方法将数据与对数据的操作作为一个整体，并且数据本身对外界常常是隐藏的。该方法所具有的封装性、继承性和多态性为提高代码的可重用性、可扩充性和可维护性提供了有力的技术保障。面向对象程序设计方法适用于大型程序的开发和维护，是目前主流的程序设计方法之一。

1.1.1 面向过程的程序设计

面向过程程序设计思想的核心是**功能分解**，通常采用自顶向下、逐步求精的方法进行：将一个大规模的、具有复杂系统的设计任务按功能逐步分解为若干小规模的、易于控制和处理的子任务，这些子任务都是可以独立编程的子程序模块。每个子程序功能单一，调用方便。C 语言用函数来实现各子程序模块，最后在 main( )函数中，通过合理的流程控制，将这些函数有机地组织成完整的程序。

由于采用了模块分解与功能抽象，因此分解出来的功能模块可以作为功能复用的基本单元，相似功能可以组成子程序库以供复用。C 语言就提供了很多标准函数库供程序开发人员使用。

面向过程的程序设计具有直观、条理性强、结构清晰的特点。但是，面向过程的程序设计方

法以功能为核心，将数据和对数据的操作分离，功能要求一旦发生改变，就可能需要重新定义数据结构，从而使很多代码需要重新开发。因为一旦据结构发生改变，与之相关的所有操作都必须改变，对应于这些操作的函数代码将会改变。有的时候，即使功能类似，但由于用于不同的数据结构之上，所以代码都需要重新编写而无法复用。因此，面向过程的程序设计存在着代码可重用性和可维护性差的弱点，不适合大型程序的开发和维护。

下面通过实例进一步讨论面向过程的程序设计方法。

一个简单的学生成绩管理系统用来管理若干个学生的信息。每个学生的信息包括学号、姓名、某门课的平时成绩、期末成绩、总评成绩和名次，其中总评成绩=平时成绩×0.3+期末成绩×0.7。为实现成绩的录入、计算、排名、输出等操作，需要定义一个合适的数据结构。这里定义如下简单的结构体类型 SS 表示每个学生的信息。

```
typedef  struct  studentScore
{   char number[10];                    //学号
    char name[10];                      //姓名
    int dailyScore ;                    //平时成绩
    int finalScore;                     //期末成绩
    float generalScore;                 //总评成绩
    int place;                          //名次
}SS;
```

根据功能要求，定义以下函数：

```
void readData(SS stu[ ],int n);         //输入 n 个学生的学号、姓名、平时及期末成绩
void calcuScore(SS stu[ ],int n);       //计算 n 个学生的总评成绩
void sortScore(SS stu[ ],int n);        //根据总评成绩排名，得出每个学生的名次
void printOut(SS stu[ ],int n);         //按一定的格式输出 n 个学生的完整信息
```

假设上述结构用来统计 C 语言成绩，但是在 C 语言课程中，需要增加实验成绩这一项，学生的总评成绩计算方法可修改为：总评成绩=平时成绩×0.2+实验成绩×0.2+期末成绩×0.6。显然，类型 SS 中需要增加一个数据项 int experiScore; 表示实验成绩。数据结构改变了，部分功能也需要调整，因此以上 4 个函数的代码都要改变，可见，面向过程的程序设计可复用性较差，不利于软件维护。

面向过程的程序设计的范型是“**程序=算法+数据结构**”，数据与对数据操作的分离导致软件维护（包括软件的测试、调试和升级）的困难。随着软件业的繁荣，面向对象的程序设计方法应运而生，它很好地解决了以上问题。

1.1.2 面向对象的程序设计

与面向过程的程序设计不同，面向对象的程序设计将数据以及对这些数据的操作以**类（class）**的形式**封装（encapsulate）**为一个整体，以类的**对象（object）**作为程序的基本元素，通过向对象发送**消息（message）**，进而由对象启动相关的方法完成各种功能。同时，数据本身不能被类外的程序和过程直接存取，这种机制增强了数据的安全性和软件的可靠性。

类与对象是抽象与具体的关系。以属于类的对象作为程序的基本元素，符合人们习惯的思维方式，也符合现实世界的组成和运作规律。现实世界就是由一个个对象组成。人、动物、物品、事情都是属于不同类的对象，每一个对象有一些属性，也有一些行为，并且各有一个名字。在面向

对象程序设计中，属性作为类中的数据项称为**数据成员（data member）**，表示一类对象所共有的静态特性；行为作为类中的操作称为**成员函数（member function）**，表示一类对象所共有的动态特性。

例如，人类具有姓名、性别、年龄、身高、体重、身份等属性，同时还有吃、穿、住、行、学习、工作等行为。如果定义一个 Person 类来表示人类，则该类应该有 name、sex、age、height、weight、identity 等数据成员，同时还有 eat( )、dress( )、reside( )、act( )、study( )、work( )等成员函数。如果 teacher1 是人类的一个具体对象，teacher1 就是该对象的名字，这个对象有具体的数据成员的值，例如，张虹、女、25 岁、1.70m、55kg、教师，同时也有吃、穿、住、行、学习、工作等行为即成员函数。

实际上，类就是一种类型，与一般的类型不同，该类型不仅有数据成员，同时还包含对数据成员操作的成员函数；而类的对象就是属于该类型的变量，显然每一个对象都有一个对象名，各个对象拥有自己独特的数据成员值。

对于上面成绩管理的例子，在面向对象的程序设计中，将结构体改造成一个类，类中包括数据成员及操作数据的成员函数，类 SS 的定义如下：

```
class  SS                              //定义一个类 SS
{                                      //以下几项是数据项，作为类内的数据成员
private:                               //数据成员是私有的，即对外部函数不公开
    char number[10];                   //学号
    char name[10];                     //姓名
    int dailyScore;                    //平时成绩
    int finalScore ;                   //期末成绩
    float generalScore;                //总评成绩
    int place ;                        //名次
public :                               //以下是类的公有成员函数，是类的对外接口
    void readData() ;                  //输入当前学生的学号、姓名、平时及期末成绩
    void calcuScore();                 //计算当前学生的总评成绩
    void printOut();                   //按一定的格式输出当前学生的完整信息
    …                                  // 其他函数，略
};
⋮                                      //类内函数及友元函数的具体实现代码省略
```

主函数中，通过定义类的对象，再向对象发送消息完成程序，例如：

```
int main( )
{
    SS  s,t;                           //定义 SS 类型的两个学生对象，对象名为 s 和 t
    s.readData( );                     //通过向对象发送消息，对象接收消息后调用成员函数
    t.readData( );                     //readData()实现读入每个学生的信息
    s.calcuScore( );                   //对象调用成员函数 calcuScore()计算总评成绩
    t.calcuScore( );
    s.printOut();                      //对象调用成员函数 printOut()输出各自的信息
    t.printOut();
    return 0;
}
```

在以上的类定义中，所有的数据成员以及成员函数的具体实现代码都被封装和信息隐藏，只有成员函数的原型，即**接口（interface）**对外公开，在主函数中只能通过类的对象调用类的接

口函数，间接地修改数据成员的值，而不能通过对象直接操作被封装的数据成员，这是面向对象程序所具有的封装性，这种特性增强了代码的安全性。

在这个例子中，如果在类 SS 中增加一个数据项 int experiScore; 表示实验成绩，总评成绩的计算方法改变，则类内的几个函数代码需要做相应的修改，而主函数中的代码不需做任何变动。

通过这个简单示例，可以得出以下结论。

（1）面向对象的程序一般由类的定义和类的使用两部分组成，类的使用表现为类对象的定义与功能调用。

（2）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。

在实际的程序开发过程中所涉及的类，可以由程序设计者自己开发，也可以使用已有的类（包括类库中提供的类和其他人开发完成的类）。程序设计者最好使用已有的类，这样可提高代码的可重用性，缩短开发周期，提高效率，这也是面向对象程序设计所提倡的方法。例如，在上面成绩管理的例子中，如果类 SS 是已有的类，则程序设计者只要完成主函数中一些简单的代码，定义类 SS 的对象并掌握接口函数的调用方法，向对象发送正确的消息即可，至于类 SS 内部细节则不必关心。当类内的数据成员发生变化时，只有开发类库的程序员需要了解底层细节，对操作类内数据成员的成员函数代码进行修改，尽量保持接口不变，而直接使用 SS 类的程序员就不必修改自己的程序代码。因此，面向对象的程序设计在代码的可重用性和可维护性上具有面向过程的程序设计无法比拟的优势。

面向对象的程序最终是通过定义各个类的对象，向对象发送消息，对象在接收到消息后启动相应的方法完成各种操作。向对象发送消息的形式如下。

对象名.成员函数名（实际参数表）

面向对象的程序设计是一种新型的程序设计范型，其特征为“**程序=对象/类+对象/类+…**”以及“**对象/类=数据+操作**”。类的使用除了用来定义对象完成各种功能以外，还可以在已有类的基础上再增加一些其他属性和行为，派生出新的类，即利用**继承（inheritance）**机制形成父类与子类的类层次关系；也可以在定义一个新类时将已定义过的类的对象作为新类的数据成员，这是类的**聚合关系**。此外，向对象发送同样的消息，有可能执行不同的代码产生不同的效果，这就是**多态性（polymorphism）**。

因此，面向对象程序设计中最突出的特征是**“封装性、继承性和多态性”**，最重要的概念是**类和对象**，面向对象的程序设计就是围绕**类的定义**和**类的使用**展开的。

1.2 面向对象的基本概念及特征

面向对象的程序设计方法是一种利用抽象、封装等机制，借助于对象、类、继承、消息传递和多态等概念进行程序设计的方法。

1.2.1 类与对象

类与对象是抽象与具体的关系，是一对相互依存的概念。**类**是具有相同属性和操作的一组对象的集合，它为属于该类的全部对象提供了统一的抽象描述。而**对象**是对问题域中客观存在事物

的抽象，是类的具体的个体，也称为类的一个实例。例如，教师是一个类，李老师则是教师类的一个对象。

在面向对象的程序设计方法中，类实质上就是一种类型，但这种类型与一般类型不同。类包括数据成员和成员函数。类类型体现的是在面向对象的程序设计中以数据为中心，将数据与对数据的操作捆绑在一起的思想。

在上一节的类 SS 中，number、name、dailyScore、finalScore、generalScore、place 都是数据成员，分别表示学生的学号、姓名、平时成绩、期末成绩、总评成绩、名次。注意：每一个数据成员必须属于某一特定的类型，这种类型既可能是简单类型，也可能是其他类型。类 SS 中的成员函数有 void readData( )、void calcuScore( )和 void printOut( )，分别实现对类内数据成员的输入、计算和输出功能。

与类是一种类型相对应，对象实际上属于类类型的一个变量，这种变量与一般变量不同，对象由一组具体的属性（即数据成员）的值来标识，可以执行类所定义的行为（即成员函数），数据成员是描述对象静态特性的数据项，成员函数是描述对象动态特性的操作。

例如，上面的类 SS，主函数中定义了两个对象，对象名分别为 s 和 t，每个对象都是通过调用成员函数 readData()读入数据，使对象的一些数据成员获得值，通过调用成员函数 printOut()输出每个对象数据成员的值。

由此可见，**每个对象都具有以下特征**。

（1）必须属于某一个类，必须有一个区别于同类型其他对象的对象名。

（2）对象有自己的属性值，即每个对象的数据成员有特定的值来标识该对象的静态特性。

（3）对象还可以有一组类所规定的操作，每一个操作决定对象的一种动态行为，通过“**对象名.成员函数名（实际参数表）**”的形式实施这种行为。

由于对象属于类，对象可以有哪些数据成员来表达其静态特性，可以执行哪些成员函数以实现其动态特性，实际上都取决于类的设计，因此在面向对象的程序设计中，最富挑战性和创造性的工作就是类的设计。同时，面向对象编程也是以类的设计为基础的，只有设计好了类，才能通过类的对象展示面向对象程序设计的魅力。

1.2.2　封装性

封装，顾名思义，就是将某事物包装起来，使外界不了解它的详细内情。封装性使得面向对象程序设计具有面向过程程序设计无法比拟的安全性和可靠性。

封装在生活中无处不在。例如，用户使用 MP4 时，可以通过其面板上的控制按钮打开电源，进行菜单选择、选歌、调音量、播放视频等。这些控制按钮相当于这台 MP4 对象提供给用户的接口，用户只要知道这些按钮的功能并且会使用就可以了，至于 MP4 里面有哪些元器件以及用户按下某一个按钮时其内部的元器件状态如何改变、如何参与工作等细节对用户是隐藏的，通过机壳将这些都封装在里面。这样封装使 MP4 不容易损坏，且使 MP4 的制造技术保密。

在面向对象的程序设计中，封装主要是针对对象而言的，对象就是一个数据和操作的封装体。在程序设计语言中，对象的 private 或 protected 成员被封装和信息隐藏，而对象的 public 属性成员呈现少量的对外接口，但具体实现细节（即成员函数的实现代码）都对外隐藏。

在 1.1.2 小节的类 SS 中，数据成员 number、name、dailyScore、finalScore、generalScore、place 以及成员函数 void readData()、void calcuScore()及 void printOut()的实现细节都被封装起来，因而这些信息对于外界是隐藏的。但是，3 个成员函数的原型对外公开。这样，在 main 函数中，通过

类 SS 的对象接收这些接口消息，从而间接地修改对象的数据成员值，否则，如果数据成员公开，则在主函数中可以直接修改某对象的总评成绩值，数据的安全性得不到保证。

由于封装的单位是对象，而对象总是属于某一个类，因此，在封装之前需要仔细做好数据抽象和功能抽象的工作，明确一个类中有哪些数据成员和成员函数，哪些成员需要信息隐藏，哪些成员应该对外公开，以便在封装时决定提供哪些对外接口。

封装机制使对象将非 public 成员以及接口函数实现的内部细节隐藏起来，并能管理自己的内部状态。外部只能从对象所表示的具体概念、对象提供的服务和对象提供的外部接口来认识对象，通过向对象发送消息来激活对象的自身动作，实现一定的功能。

1.2.3 继承性

继承是面向对象的程序设计提高代码重用性的重要措施。继承表现了特殊类与一般类之间的上下分层关系，这种机制为程序员提供了一种组织、构造和重用类的手段。继承使一个类（称为**基类**或**父类**）的数据成员和成员函数能被另一个类（称为**派生类**或**子类**）重用。在派生类中只需**增加**一些基类中没有的数据成员和成员函数，或是对基类的某些成员进行**改造**，这样可以避免公共代码的重复开发，减少代码和数据冗余。

例如，图 1.1 描述了类的继承关系。如果已经定义了学生类，其中，有表示就读学校、姓名、学号、成绩等的数据成员，还有表示上课、考试的成员函数。现在，定义一个大学生类，就只要再增加专业、学分这 2 个数据成员，增加毕业设计等成员函数，对学生类中已有的所有成员直接使用。这样，在定义大学生类时需要书写的代码相对较少，缩短了开发周期。

如果没有继承机制，每次的程序开发都要从“零”开始，使得系统的开发和维护开销都很大，面向过程的程序设计就缺乏继承机制。

从图 1.1 中可以看到，类的继承可以多次进行，从一般的学生类到它的子类大学生类再到更下一层的在职大学生类，越往下层越具体。最下层的在职大学生类不仅继承了直接基类大学生类的所有特性，还继承了间接基类学生类的特性。

从继承源上看，继承分为单一继承和多重继承两种。单一继承指派生类只从一个基类继承而来，多重继承指派生类同时从多个基类继承而来。显然，在图 1.1 中，大学生类对学生类的继承就是单一继承，而在职大学生类是从大学生类和职工类多重继承得到的。因此，在职大学生类将同时具有大学生类和职工类的特性，另外还具有自己本类中新增加的特性。

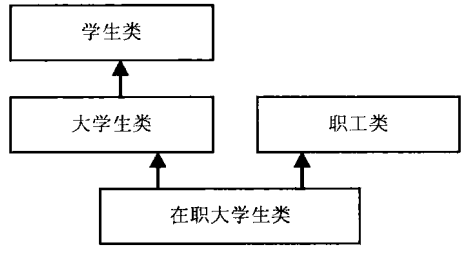


图 1.1 类继承关系示意图

1.2.4 多态性

多态性源自面向对象编程语言，而并不是面向对象方法的基本特征。在自然语言和日常工作生活中，同一个词经常在不同的场合下代表不同的意义，这就是多态性。例如“发动”这个动词，“发动汽车”与“发动火车”所执行的动作不一样，“发动公民义务献血”中其含义又不一样。

面向对象程序设计的多态性指一种行为对应着多种不同的实现方法。多态性包括静态多态性（也称为编译时的多态性）和动态多态性（也称运行时的多态性）两种。

在同一个类、无继承关系的平行类、不属于任何类的普通函数中，同一个函数名对应着不同的实现代码，这种多态性称为静态多态性。例如，学生类中定义了“考试”这一成员函数，但是

考试形式可以有多种，不同的课程可能考试的具体过程也不一样，这时根据学生对象接收到的考试这一消息中所传过来的课程信息，产生不同的考试行为。

在一般类和它的各个派生子类中，同一个函数名可以对应着不同的实现代码，这种多态性称为动态多态性。例如，在图形类中说明了一个函数 getArea()，用来求当前图形的面积，但是在图形类中没有给出具体代码，因为具体图形未确定。利用继承机制定义了几个派生类：三角形类、圆形类、正方形类、长方形类，在每一个派生类中都重新定义了函数 getArea()。显然，为求相应图形的面积，实现的代码不一样。这些知识在第 6 章中将会详细介绍。

多态性的意义在于用同一个接口实现不同的操作，这样，直接使用类来进行程序开发就很方便。例如，为求面积，如果在每一个类中函数名都不一样，则既不科学，又会给使用者增加记忆难度。

1.3　C++语言概述

C++语言的研发始于 1980 年，贝尔实验室的 Bjarne Stroustrup 对 C 语言进行了改进和扩充，增加了对面向对象程序设计的支持。最初的成果称为“带类的 C”，1983 年正式取名为 C++，在经历了 3 次修订后，于 1994 年制定了 ANSI C++标准的草案，以后又经过不断完善，成为目前的 C++语言，并仍在不断地发展。C++语言是同时支持面向过程程序设计和面向对象程序设计的混合型语言，是目前应用最为广泛的高级程序设计语言之一。

1.3.1　C++语言支持面向对象的程序设计

进行面向对象的程序设计，必须使用面向对象的程序设计语言。面向对象的程序设计语言应该具有以下几个特点。

（1）支持对象的概念并拥有对象所有的特点。

（2）对象属于类。

（3）提供类的继承机制。

C++语言是在传统 C 语言的基础上进行改造和扩充并引入了面向对象的概念和方法，支持面向对象的程序设计，具体表现在以下 3 个方面。

1. 支持封装性

C++语言允许使用类和对象。类是支持封装的工具，对象是封装的实体，是封装的具体实现。类的成员具有不同的访问权限，类的私有成员仅由该类体内的成员函数访问，因此私有成员具有信息隐藏性，在类体外不可见。类的公有成员是类体与外界的接口，类体外的函数可以访问类的公有成员。类中还有一种保护成员，它具有公有成员和私有成员的双重特性，多用于类继承机制中。

2. 支持继承性

C++语言支持面向对象程序设计中的继承，它同时支持单一继承和多重继承。继承性给 C++语言的编程带来了方便，提高了代码的可重用性，增强了程序的可扩展性，提高了软件的开发效率。继承是两个类之间的关系，基类和派生类是继承中的重要概念。派生类继承了基类的所有成员，并且可以增添自己特有的新成员，改造从基类继承来的成员。继承实现了抽象和共享机制。

3. 支持多态性

多态性是在继承性基础上的面向对象程序设计的重要特性之一。不同编程语言支持多态性的方式有所不同。C++语言同时支持静态多态性和动态多态性，主要表现在以下两个方面。

（1）支持**静态联编**以实现静态多态性。静态联编指程序运行之前就已经可以完全确定该同名接口调用的是哪一个版本。C++语言通过函数重载和运算符重载实现静态联编。重载指同一个函数名可以有多种不同的实现代码，即同一个行为对应不同的实现，在重载时要求形式参数的个数、类型或顺序有所区别。

（2）支持**动态联编**以实现动态多态性。动态联编是在程序运行时才可以确定该同名接口究竟来自于基类还是某一个派生的子类。动态联编是在公有继承的前提下通过虚函数、基类的指针或引用实现的。动态联编虽然运行效率不如静态联编，但是它可以通过高度抽象提高程序的灵活性和可扩充性。

1.3.2 C++语言与 C 语言的关系

C++语言由 C 语言发展而来，兼容 C 语言，并对 C 语言作了改进和扩充。作为面向过程的程序设计语言，C++语言与 C 语言的关系可以用**继承**和**改进**来概括。

1. C++语言继承了 C 语言

C 语言是 C++语言的一个子集。C 语言的词法、语法等绝大多数都可以直接用到 C++语言中。例如，C 语言中的类型、运算符和表达式在 C++语言中都可以使用；C 语言的语句也是 C++语言的语句；C 语言中的函数定义及调用在 C++语言中也合法；C 语言的预处理命令也可用于 C++语言；C 语言中的构造类型，如数组、结构体和联合体类型在 C++语言中也可以使用，但是 C++语言提供了更简洁灵活的用法；C 语言的指针在 C++语言中一样使用，但是 C++语言在动态内存空间的管理上引入了更方便的方式，C++语言通过增加“引用”大大减少了不安全指针的使用；C 语言中关于作用域的规则、存储类别的规定在 C++语言中也都适用。

由于 C++语言继承了 C 语言，所以 C++语言保持了 C 语言简练明了的风格，也保留了 C 语言面向过程的特性。在利用 C++语言进行面向过程的程序设计时，可以有多种方案，可以完全使用 C 语言风格，但是使用 C++语言风格更为方便。掌握 C 语言的读者学习用 C++语言进行面向过程的程序设计相当容易，只要重点学习 C++语言对 C 语言的改进部分就可以。

2. C++语言改进了 C 语言

C++语言虽然保留了 C 语言的风格和特点，但又针对 C 语言的某些不足做了改进。下面简单列举一些 C++语言对 C 语言的改进内容，更详细的介绍见第 2 章。

（1）C++语言提供了与 C 语言不同的 I/O 流类库，方便了输入/输出操作。

（2）C++语言新增加了行注释符（//），为单行注释提供了方便。

（3）C++语言引入了名字空间，防止同名的冲突问题。

（4）C++语言建议使用 const 关键字来定义符号常量，使常量具有类型，解决了宏定义下的符号常量无类型说明的问题。

（5）新增加了专用于处理逻辑值的 bool 类型，增加了 string 类型方便字符串的处理，允许定义无名联合、无名枚举类型，并且有新的用法，对结构体类型也进行了扩充，结构体中可以有成员函数。

（6）C++语言规定，凡是从高类型向低类型转换都需要加强制类型转换，并且增加了类似于函数调用形式的强制类型转换方式。

（7）C++语言规定函数说明必须使用原型声明方式，不允许使用简单说明方式。

（8）C++语言允许局部变量随用随定义，不必限定在语句块的开头定义，更加灵活。

（9）C++语言允许函数的形式参数带有默认值，这给函数调用带来方便。

（10）C++语言引进了函数重载和运算符重载机制，方便了编程。

（11）C++语言引进了内联函数，建议使用内联函数代替带参数的宏定义，增加了对参数类型

的说明。

（12）C++语言引进了引用的概念，使得可以通过变量的别名直接操作变量本身，而不必通过指向变量的指针间接操作变量，这样大大减少了指针的使用，提高了安全性。

（13）C++语言利用指针使用 new 和 delete 运算符代替函数更方便地进行动态内存空间的分配与释放。

（14）C++语言提供了异常的检查、处理机制，提高了程序的健壮性。

1.3.3　其他面向对象的程序设计语言

高级语言层出不穷，面向对象的程序设计语言作为高级语言的一种，研发开始于 20 世纪 60 年代，先后出现了 Simula、Smalltalk、Object-C、Eiffel、Ada、C++和 Java 等面向对象的程序设计语言，每种语言各有其优势和应用领域。

20 世纪 60 年代开发的 Simula 67 语言被誉为面向对象程序设计语言的鼻祖，因为它提出了对象、类、继承的概念和面向对象的术语，面向对象程序设计的许多原始思想都来源于 Simula 语言。

Smalltalk 语言是 20 世纪 70 年代开始开发的，它完整体现了来自 Simula 以及其他早期原型语言中面向对象的概念，历经 Smalltalk-72、Smalltalk-76 和 Smalltalk-80 几个版本，现在一般用 Smalltalk-80。

Object-C 语言是 1983 年左右开发的，它在 C 语言的基础上进行了扩充，通过新引入的构造和运算符来完成类定义和消息传递，其语法更像 Smalltalk 语言。

Eiffel 语言从理论上讲是最好的面向对象的程序设计语言，它除了封装和继承，还集成了几个强有力的面向对象的特征，如参数化多态性、对方法实施前置条件和后置断言等。

Ada 语言的开发工作始于 1975 年，最初设计是为了构建长周期的、高度可靠的软件系统。Ada 语法严谨、书写优美、可读性强，它提供了一系列功能来定义相关的数据类型（type）、对象（object）和操作（operation）的程序包（package）。Ada 有 Ada 83 和 Ada 95 两个主要版本，Ada 一度被美国国防部强制指定为军用武器系统的唯一开发语言。

Java 语言是由 SUN 公司在 20 世纪 90 年代初开发的一种面向对象的程序设计语言，其优点是简单、面向对象、不依赖于硬件结构、可移植性强、安全性高、能最大限度地利用网络，因此被广泛用于网络编程。

1.4　C++程序及其开发

用 C++语言开发程序，必须遵循一定的步骤与方法。有功能强大的集成开发环境的支持，会使 C++程序的开发工作变得更轻松。

1.4.1　C++程序的开发过程

开发 C++程序与开发其他高级语言的程序一样，包括编辑、编译、链接和运行几个步骤。任何 C++程序的开发均遵循如图 1.2 所示的步骤。

1. 编辑

编辑是将编写好的 C++语言源程序通过输入设备录入到计算机中，生成扩展名为 .cpp 的源程序文件。编辑源程序的方法有两种：一种是选用 C++集成开发环境中的编辑器，这是最常用的方法；另一种是使用机器中的其他文本编辑器，如写字板、记事本等。

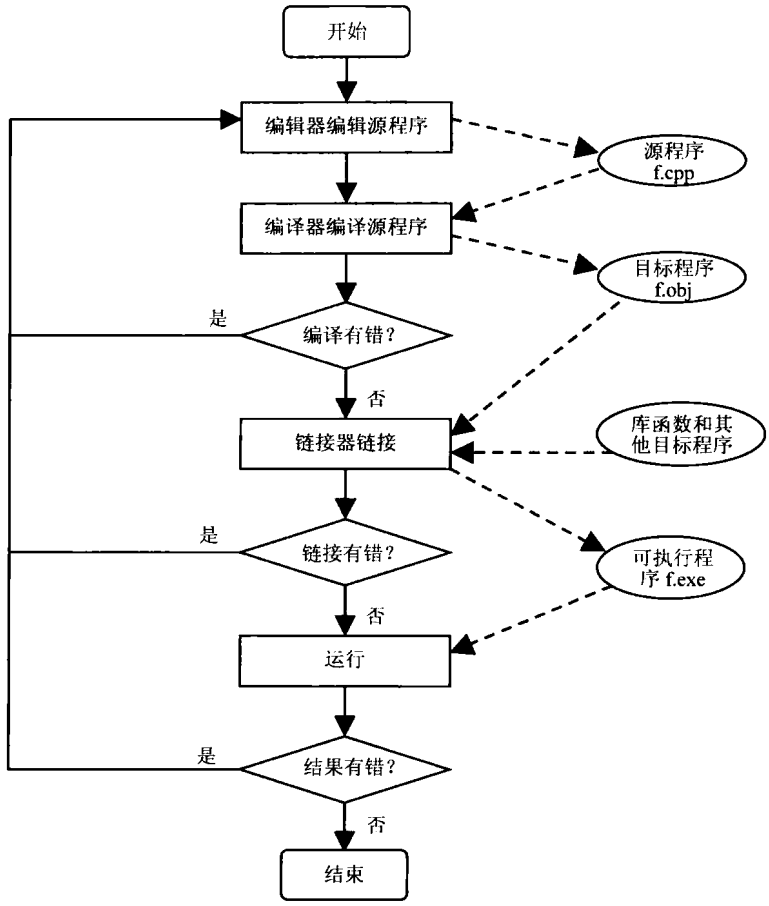


图 1.2　开发 C++程序的步骤

2. 编译

编译是将已生成的 C++语言源程序代码转换为机器可识别的目标代码（即二进制代码），生成相应的 .obj 文件。编译又包括预处理和编译两个子过程，先执行程序中的以“#”开头的预处理命令进行预处理，然后再进行正常的编译过程。在编译过程中主要进行词法和语法分析，发现有不符合的地方，及时显示错误或警告信息提示用户，用户必须重新修改源程序文件直至编译正确才能进行下面的过程。

3. 链接

链接是在编译生成的目标代码中加入某些系统提供的库文件代码，进行必要的地址链接，最后生成扩展名为 .exe 的可执行文件。

4. 运行

生成了可执行文件后就可以运行程序。运行程序的方法很多，最常用的是选择集成环境中的“运行”命令来运行可执行文件，另一种方法是在 MS-DOS 提示符后直接输入可执行文件名（如果主函数有形式参数，则应提供实际参数），按回车键确认。运行后在显示器上显示结果。

注意： 在以上 4 个过程中都有可能出错，无论是哪一个阶段出了错，都应该回到编辑步骤，因为如果源文件有错，就无法保证后面各步生成正确的文件。如果是运行出错，则程序存在逻辑上的错误，要借助调试器找出错误才能保证源程序的修改正确。