◉ 梁 鸿　孙晓燕　编著

# 面向对象程序设计与visualc++实验教程

中国石油大学出版社

TEACHING MATERIALS FOR COLLEGE STUDENTS

高 等 学 校 教 材

# 面向对象程序设计与Visual C++实验教程

◎ 梁 鸿
◎ 孙晓燕
编 著

中国石油大学出版社

**图书在版编目(CIP)数据**

面向对象程序设计与 Visual C++实验教程/梁鸿,孙晓燕编著.—东营:中国石油大学出版社,2006.8
ISBN 7-5636-1887-2

Ⅰ.面... Ⅱ.①梁... ②孙... 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.TP312

中国版本图书馆 CIP 数据核字(2006)第 089124 号

中国石油大学(华东)规划教材
书　　名:面向对象程序设计与 Visual C++实验教程
作　　者:梁　鸿　孙晓燕

**责任编辑**:袁超红　高　颖(电话 0546－8395745)
**封面设计**:TRANSMART

**出 版 者**:中国石油大学出版社(山东　东营　邮编 257061)
**网　　址**:http://www.uppbook.com.cn
**电子信箱**:shiyoujiaoyu@126.com
**排 版 者**:海讯科技有限公司
**印 刷 者**:东营市新华印刷厂
**发 行 者**:中国石油大学出版社(电话 0546－8391797,8392791)
**开　　本**:180×235　印张:9　字数:181 千字
**版　　次**:2006 年 8 月第 1 版第 1 次印刷
**定　　价**:13.00 元

## 内 容 简 介

本书内容可分为基础知识和上机实验两部分。基础知识部分以介绍面向对象程序设计和Windows编程知识为主。由于Visual C++环境内容丰富，为了能与有限的上机实验学时相配合，上机实验部分以几个大作业的形式由浅入深地进行讲解。学生完成本书的学习之后，可以初步掌握面向对象程序设计与Visual C++编程环境，从而为进一步深入学习或自学Visual C++奠定良好的基础。

本书可作为面向对象程序设计、程序设计实习等课程的实验教程，也可用作编程爱好者的参考资料。

# Preface
# 前 言

面向对象程序设计是目前主流的程序设计方法，Visual C++是目前广泛使用的一种基于Windows图形用户界面的程序设计集成开发环境。借助Visual C++提供的工具，可以比较容易地进行面向对象程序的开发，设计具有图形界面的应用程序。本书作为《面向对象程序设计》、《面向对象引论与Visual C++》、《程序设计实习》、《软件综合实习》等课程的配套实验教程，可以起到学习Visual C++和练习面向对象编程的双重作用。

由于Visual C++环境本身包含的内容丰富，需要较多的时间来熟悉与运用，为了能与课程有限的上机实验学时相配合，在编写这本实验教程指导书时，以完成几个大作业的形式来由浅入深地讲解。主要的一个大作业名为SmallCAD，是一个小型的、功能相对简单的图形绘制程序。在此程序中，读者可以学习到Visual C++的数据处理、图形绘制、鼠标消息响应、MFC常用类的使用、文档存取、菜单与对话框等资源的设计使用、程序的调试、联机帮助的使用，还可以练习面向对象的知识，例如类的设计、类继承层次的设计、类的特殊成员的使用、抽象类与虚函数等。

另外一个大作业是针对目前数据库广泛应用的现状，设计实现一个使用数据库的学生教学管理程序。在常用的数据库中建立一个简单的教学信息数据库，实现在Visual C++环境中使用此数据库，浏览数据库的内容，对其中的数据进行查询、修改、打印输出等操作。

完成本书的学习之后，学生可以初步掌握面向对象程序设计知识与Microsoft Visual C++编程环境，并且为今后进一步深入的学习或自学Visual C++打下良好的基础。课程内容以讲义形式安排，每讲对应2个学时。前面1至4讲为基础知识部分，可用于自学或课堂授课学时，后面5至15讲用于上机实验。其中，程序调试与联机帮助功能两讲的内容相对比较独立，可以依据上机实验的实际情况提前讲解。

本书共包含15讲。1至4讲的面向对象和Windows编程基础知识部分由梁鸿编写；5至15讲的上机实验内容设计与上机作业设计由孙晓燕编写。特别感谢计算机与通信工程学院的庄长淞教授。庄老师退休前为研究生讲授的《面向对象程序设计》课程使作者受益匪浅，SmallCAD大作业的创意也来自庄老师，在此向庄老师表示衷心的感谢。

由于编写时间紧、经验不足，书中难免存在错误与不足，敬请读者批评指正。

作者地址：山东省东营市中国石油大学（华东）计算机与通信工程学院 257061

邮件地址：liangh@hdpu.edu.cn sun_xiaoyan@sina.com

作 者

2005年12月5日

# 目录 CONTENTS

# 第1讲　面向对象程序设计基础

近年来，在计算机软件业发展中，面向对象程序设计的思想已经被越来越多的软件设计人员所接受。这不仅因为它是一种先进、新颖的程序设计思想，更主要的是由于这种程序设计思想更接近人的思维活动，更自然客观。人们利用这种思想进行程序设计时，编程能力在很大程度得到改善，编写的程序也清晰易懂、易维护，软件的可扩充性和可重用性都有较大提高。在这样的形势下，计算机教学思想也都相应地转向以面向对象为中心来开展。作为一个计算机专业从业人员，越早接受面向对象的程序设计思想，掌握面向对象的编程技术，就越有利于程序设计能力的培养。

面向对象程序设计思想与以往普遍使用的面向过程的程序设计思想有较大的不同，新的概念较多，初学者理解上会有一些难点。因此，本讲将让读者初步了解面向对象与传统方法的不同，并向读者介绍面向对象的主要优点。

## 1.1　传统的面向过程程序设计方法

计算机和软件产业近年来取得了飞速的发展，计算机过去的主要功能——计算功能——已经逐渐被管理、通信、娱乐等功能替代。计算机软件产业也相应发生了巨大的变化，软件的规模越来越大、功能越来越强、界面越来越美观和友好。这些都对计算机程序设计提出了更高的要求：要求软件要有更高的正确性、健壮性、可维护性、可复用性等特征。传统程序设计方法的局限性也逐渐体现：软件开发的方法与人类认识世界的方法不一致，导致复杂性、自然（客观）性两大难题无法解决。

### 1.1.1　面向过程程序设计方法的核心

传统的面向过程程序设计方法从计算机诞生一直使用至今，它是通过在程序中模拟问题求解的过程来解题（编制相应的程序）的，代表性的语言有：汇编，BASIC，PASCAL，C，FORTRAN，COBOL等。

面向过程思想的核心就是功能的分解。拿到一个待求解的问题后，将其划分为几个大的步骤，每一步为一个功能模块，相互之间一般是顺序关系。每个功能模块由一个过程或函数来实现。过程或函数是建立大型复杂软件的核心机制。当求解的问题较为复杂时，为了控制各个过程的复杂性，可以将一个复杂的过程不断分解细化，直到程序设计人员可以理解每个过程或步骤为止。面向过程思想求解问题的步骤如图1-1所示。

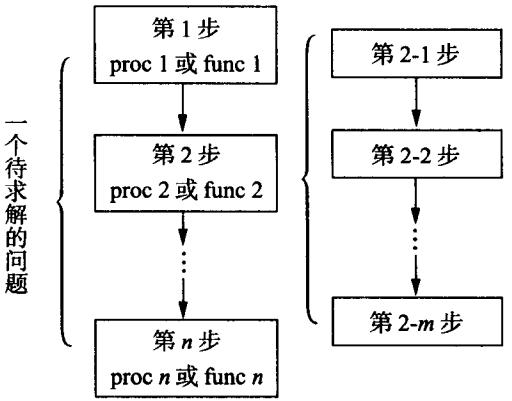


**图 1-1　面向过程程序设计求解问题的步骤**

### 1.1.2　面向过程方法示例

下面以一个简单的程序来说明面向过程程序设计方法。面向过程方法的具体解题步骤基本如下：

（1）将所求问题分解为若干个功能块（模块）。

（2）根据模块功能来设计数据结构以存储数据。

（3）编写若干过程或函数来操作这些数据，构成各功能模块。

【例 1】用面向过程的方法设计程序——设置并显示当前日期。

下面分析本题目要求，按照上述步骤得到解题思路。

（1）模拟解题步骤：第 1 步询问当前日期；第 2 步接收用户输入的当前日期；第 3 步显示当前日期。

（2）设计数据格式：用结构体存放日期数据。

```
struct Date                    //定义数据
{   unsigned int year;         //年
    unsigned int month;        //月
    unsigned int day;          //日
};
```

（3）编写各个功能模块：复杂的功能模块需要用过程或函数来实现，简单的步骤只用若干条语句即可实现。

实现方法 1：

```
#include <iostream.h>          //输入输出库函数文件
struct Date                    //定义日期数据
{   unsigned int year;         //年
    unsigned int month;        //月
```

```
    unsigned int day;        // 日
};
void SetDate(Date *d, int y1, int m1, int d1)      //设置日期函数
{   d->year=y1; d->month=m1; d->day=d1;   }
void ShowDate(Date *d)      //显示日期函数
{   cout<<"Today is:";
    cout<<d->year <<'/'<<d->month<<'/'<<d->day <<endl;
}
void main()      //主函数——解题过程
{   struct Date today;
    unsigned int year, month, day;
    cout<<" 请输入今天的日期:年 月 日 ";
    cin>>year>>month>>day;
    SetDate(&today, year, month, day);
    ShowDate(&today);
}
```

实现方法 2：由于此题目非常简单，也可以不编制小功能的函数。

```
#include <iostream.h>
int main()
{   unsigned int year, month, day;
    cout<<" 请输入今天的日期:年 月 日 ";
    cin>>year>>month>>day;
    cout<<" 今天是:"<<year<<" 年 "<<month<<" 月 "<<day<<" 日 "<<endl;
    return 0;
}
```

### 1.1.3 面向过程方法的主要缺点分析

面向过程的主要缺点可以归结为一句：过程与数据结构的分离。对程序设计者而言，过程是重点，设计人员主要考虑功能要如何分解，在每个过程中着重安排语句序列；数据结构为辅，数据结构的定义与使用它们的语句在位置上分离，而程序员编程时又必须时时处处考虑所使用的数据结构，在语句中使用数据结构、修改数据的取值。

正是这种矛盾导致了面向过程程序设计的后果——软件设计困难。具体表现在以下几个方面：

1. 软件的可维护性差

程序设计过程是一个瀑布式的模型，从分析、设计到实现，每一步都以上一步为基础并细化，导致修改困难。用户需求的微小变化，一般都会引起细节实现的大量修改。

以前面【例1】设置并显示日期程序为例，由于中、美、英几国常用的日期表示习惯不同，要求将上例中适合中国习惯的设置和显示方法修改为适合美国习惯。针对这个修改要求，需要修改数据结构定义，还要修改各过程相关的语句。可以说是“牵一发而动全身”，修改工作涉及太大，效率较差。

2. 软件的可重用性差

如果软件的可重用性好，前人的工作就可以被方便地大量借鉴。【例1】中设置并显示日期功能应该属于一个比较常见的功能，但这个功能是分散在数据定义及主函数对它实施的操作中，不是一个完整的模块，只能提取程序段重用。而真正的重用性必须由文件来保证，如C语言中的库函数stdio.h，用#include <stdio.h>预编译命令将整个文件复制来使用。然而，【例1】中使用日期的代码在main函数中是无法直接重用整个文件的。

3. 数据的安全性与正确性难以保证

仍然以【例1】设置并显示日期程序为例：

（1）程序中任何地方都可以修改数据内容。如输出日期之前将年赋值修改，编译不会出错，但显示结果有误。

（2）数据取值范围与数据含义难以保证一致。日期的年、月、日都定义为整数，但月的取值只能是1～12，日的取值则根据月份和是否是闰年而变化。这就需要编码控制数据范围与数据含义的对应关系。

（3）数据允许的操作范围难以控制。如整数允许做加、减、乘、除等操作，但对日期中的整数年、月、日，乘、除操作显然是没有意义的。

4. 不适应软件规模的扩大

软件产业的一个显著特征就是软件规模越来越大。以Word为代表的字处理系统设计为例，其中包括很多操作，如文字输入、文字修改、插入图表、格式设置（文字、段落、页）、排版、打印等。用面向过程方法设计会遇到一些困难，因为复杂的系统很难设计各个过程，很难限定各个过程的先后顺序，很难设计各个过程的相互调用关系等。

## 1.2 先进的面向对象程序设计方法

从20世纪80年代起，软件业之中开始逐渐探索并发展出一种新的程序设计思想——面向对象的程序设计方法（Object-oriented Programming）。这是一种新的程

序设计范型，它将数据和定义在其上的用户需要的操作构成一个整体——对象。

这种方法的主要特点是：

（1）避免了面向过程的主要缺点（过程与数据结构的分离），把数据结构的定义和对数据的操作构成一个统一的整体——类和对象，程序一般由类的定义和类（用户自定义类型）的变量——对象——的使用构成。

（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，启动相关的方法来完成，而不是通过过程调用实现。

下面采用面向对象的方法来设计上一节【例1】中的程序：设置并显示当前日期。

面向对象方法的解题思路是：首先分析题目要求，设计相关的类（本例中只有一个日期类）。每个类都由数据和操作构成，日期类的数据包括年、月、日共3个数据项，对日期做的操作有设置日期、显示日期。第2步完成问题要求的功能，即主函数的实现。先定义变量——日期类对象，然后操作这个对象——向其发送消息，由其执行相应的方法来完成程序要求的功能。

### 1.2.1 面向对象方法示例

【例2】用面向对象的方法设计程序：设置并显示当前日期。

（1）日期类的定义：

```
class Date      // 定义类——数据 + 操作
{  private:
   unsigned int year, month, day;      // 定义数据，年、月、日共3个数据项
   public:     // 定义操作
   void SetDate(unsigned yy, unsigned mm, unsigned dd)      // 方法1：设置日期
   {   year=yy; month=mm; day=dd;  }
   void ShowDate()      // 方法2：显示日期
   {   cout<<"Today is:"<<year<<'/'<<month<<'/'<<day<<endl;  }
};     // 类定义结束
```

（2）程序功能的实现：编制主函数，实现题目所要求的功能。

```
int main()
{
  Date today;      // 建立Date类的变量——today对象
  unsigned int year, month, day;
  cout<<" 请输入今天的日期:年 月 日";
  cin>>year>>month>>day;
  today.SetDate(year, month, day );      // 给对象发消息，启动对应方法
```

```
    today.ShowDate();      // 给对象发消息，启动对应方法
    return 0;
}
```

从上面的给出的代码看，面向对象方法与面向过程方法编制的代码似乎没有多大差别。那么，面向对象方法是如何克服传统方法的缺点，提高程序设计能力的呢？这个问题通过对这 2 个简单程序的细节对比分析就能得到结论。

### 1.2.2 面向对象方法的优点

采用面向对象方法设计程序主要有以下优点：

1. 有助于降低编程复杂度

虽然总的代码量没有太大的变化，但通过比较可以看出，面向过程方法的重要代码在 main 函数或用户自定义的函数中，而面向对象方法的重要代码则集中在类的定义中，main 函数比较简单。这个代码重心转移的意义重大，类成为整个程序设计的中心。类本身既包含数据又包含操作，将两者结合为一个统一的整体。这将更符合客观世界的特点，也避免了面向过程的主要缺点（过程与数据结构的分离）和由此导致的程序设计困难的后果。一个问题中涉及到的类个数一般不会太多，只要将类设计好，就可以定义任意多个类对象。类对象的使用代码很简单，可以说是“一劳永逸”。

2. 数据的安全性与正确性得到良好保证

类是数据与操作结合的整体，数据与操作的一致性由类来保证。类中的所有成员均有访问权限限制，数据成员一般定义成私有（private），类之外无法直接访问和修改私有成员；类的成员函数一般定义成公有（public），作为类对外使用的接口。外界只能通过类提供的接口来完成工作，类提供什么函数，外界才能做什么操作，不能随意修改数据，从而保证了数据的安全性。如整数允许加减乘除，而日期类允许的操作只有 SetDate 和 ShowDate，日期数据不会进行加减乘除操作，因此保证了数据操作的合法性。

控制数据的正确性也完全可以在类的定义中保证。如数据的取值范围问题，由于类的功能都是在成员函数中实现的，所以可以在相关的成员函数中添加代码来控制数据的正确性。例如，以下代码可在日期类中控制月和日数据的正确性：

```
void SetDate(unsigned yy,unsigned mm,unsigned dd)
{   if( mm<1 || mm>12 || dd<1 || dd>31 )
      { cout<<"Wrong Data.";   return;   }
    year=yy; month=mm; day=dd;
}
```

完整的数据正确性控制应该是按照闰年与否及具体月份来确定日的范围，此处不详细讨论。

3. 软件的可重用性好

类是一个整体，其文件组织方法固定，包括类的头文件和实现文件。需要使用某类时，用#include命令将类的头文件包含到新程序中即可。C语言的库函数、Visual C++ 中的 MFC 类库都是软件重用的典型例子。

以上日期类应以如下 2 个文件方式定义：

```
//date.h   类的头文件，只包含类体，类体中有数据成员定义和成员函数首部
class Date
{   private:
    unsigned int year, month, day;      //定义数据，年、月、日共3个数据项
    public:
    void SetDate(unsigned yy, unsigned mm, unsigned dd);
    void ShowDate();
};
//date.cpp   类的实现文件，包含成员函数的体外实现
#include "date.h"
#include "iostream.h"
void Date::SetDate(unsigned yy, unsigned mm,unsigned dd)
//方法 1：设置日期
{   year=yy; month=mm; day=dd;   }
void Date::ShowDate()
//方法 2：显示日期
{   cout<<"Today is:"<<year<<'/'<<month<<'/'<<day<<endl;   }
```

若有其他程序想使用日期类，只需要将2个日期类文件拷贝过去，并在程序中加入#include "date.h" 即可。面向对象的软件可重用性主要就体现在类的重用上，只要前人设计出良好的类，就可以任意重复使用，避免了重复代码的开发。

4. 软件易维护

当用户需求变化时，程序的修改可以主要集中在类的内部(修改数据和函数)。只要类的接口形式保持不变，程序其他部分就不必修改。这对大型软件的意义十分重大。因为软件规模越来越大，会使后期维护难度和人力、财力消耗增大，而面向对象设计方法是有效降低维护难度的好方法。例如，日期类的修改——中、美、英方式变换：

```
void SetDate(unsigned mm,unsigned dd,unsigned yy)      //只变换参数顺序
```

```
void ShowDate()      //修改输出顺序
{ cout<<"Today is:"<< month<<'/'<<day <<'/'<<year<<endl;   }
```

可以看到修改集于类中，主函数甚至可以不用变动。另外，由于Visual C++是增量编译（只编译修改的文件），因此，程序修改后重新编译程序的工作量也很小。

从以上分析可见，面向对象方法的优点与类的引入是分不开的。

### 1.2.3 面向对象方法的特性

面向对象最突出的特性是封装性、继承性和多态性。

1. 封装性（Encapsulation）

现实世界中，家用电器的使用就是一个典型的“封装”的例子。以录音机为例，它的功能以按键的方式展示在机器表面，人们通过按键来使用录音机。录音机内部电路是如何设计完成这些功能的，用户无法（也没必要）了解，因为它们是封装在机壳内部的，对用户而言是隐蔽的、不可见的。这就是所谓封装的原理。那么用户如何知道使用哪个按键呢？看录音机的说明书即可。说明书中只介绍了录音机可以做什么（what to do），并不需要告诉用户录音机是如何做的（how to do）。

面向对象程序设计的封装性是由类来体现的。类的定义中，私有数据（个数和类型）和各个成员函数的内部实现代码是被封装起来的内容；类展示给外界的是它的接口，包括类名和成员函数首部。外界可以用类名来定义类对象，通过成员函数首部定义代码来调用此函数。

类和对象封装的意义是：一方面，划定一个清楚的边界，把对象的属性（私有数据）及对属性的操作包装在这个范围内，对内部实现施加严格的保护，对象的行为能力如何实现是保密的（用户只能按规定格式使用函数，不知道数据如何表示，也不知道函数如何处理数据），保证了数据与函数的安全；另一方面，规定一个明确的外部接口，让用户明确类对象的功能并能正确使用它。

封装使模块独立性强（模块之间依赖性小，保证了数据的安全性，模块修改不影响外界，重用性好，易维护），而且封装隐藏了编程复杂度，降低了软件开发的难度。

2. 继承性（Inheritance）

自然界中每个人都从父母那里继承了一些特征，如种族、血型等，每个人又都与父母有所不同。面向对象程序设计中的继承是一种在父类与子类之间共享数据和方法的机制。类之间具有共享的特征，子类继承了父类的数据成员和成员函数；子类又不完全等同于父类，它可以新增数据和函数，还可以修改父类的函数。通过继承性，能表达类之间的一种特定关系——层次关系，从而将多个类有序地组织起来。

继承的主要作用是避免公用代码的重复开发，减少代码和数据冗余，使程序员

有了组织、构造、重用类的手段。Visual C++ 中的MFC类库包含200多个类，是典型的继承示例。另外，继承机制以相关性来组织事物，将相关的类概括成高一级的共性的类，这样可减少对相似事物进行说明或记忆的规模，相当于提供了一种简化手段。以计算机绘图为例，其中涉及多个图形类，就可以按图1-2所示的方式组织起来，上层类抽象，下层类具体。各类都有一些相同的数据和操作（如绘图、旋转等），可以使用相同的名称，便于理解掌握。

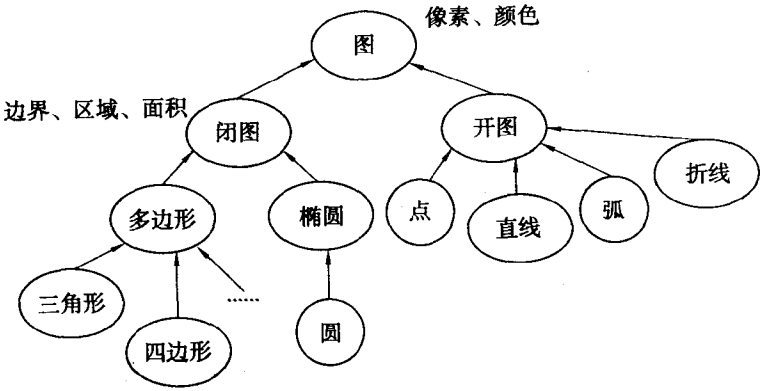


图1-2 计算机中“图”的继承关系图

3. 多态性（Polymorphism）

现实世界中自然语言就体现了多态性。如“请把门打开”、“请把电视机打开”、“请把水龙头打开”3句话中，都有“打开”，但作用在不同的事物上有不同的动作和效果。多态性的基本含义就是不同的对象收到相同名称的消息，产生不同的行为方式（执行不同的操作）。一般是用同一个名字定义不同但又类似的操作，实现“一个接口、多种方法”。

面向对象程序设计支持多种多态性的表现为：

（1）函数重载（同名函数参数个数或类型不同）；

（2）运算符重载（运算符在不同的类中有不同的运算含义）；

（3）派生类重载基类成员函数（函数名相同，函数体不同）；

（4）动态多态性（在运行时刻可以指向或引用不同类型的对象来实现调用同名的不同函数）：基类的指针或引用等动态类型 + 公有继承 + 虚函数；

（5）模板（类模板和函数模板）。

多态性符合人的思维方式，可以更好地表达行为共享，减少程序员记忆操作名称的负担。用户只需了解操作的一般含义，从而使编程简化，更易于理解，程序的能力也更强。