高等院校实验课教材

C++与数据结构实验教程

苏京霞　高　飞　编著

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

高等院校实验课教材

C++与数据结构实验教程

苏京霞　高　飞　编著

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

内 容 简 介

本书是和《C++与数据结构》(北京理工大学出版社出版)一书配套使用的参考用书。

本书分两大部分共15章。第1部分,共7章。第1章到第6章与《C++与数据结构》的面向对象的概念及C++程序设计基础相对应,其程序例题和实验题目供实验教学参考;第7章简单介绍Visual C++ 6.0集成开发环境,供读者在其上编写、调试和运行自己开发的程序。第2部分从第8章到第15章,对应《C++与数据结构》的数据结构部分。该部分在有些章节中所提供的程序例题和实验题目没有采用模板机制,有些章节的则采用了模板机制,可根据学习的需要对模板和非模板进行转换。

本书可作为高等院校电子信息类以及其他相关专业本科生学习《C++与数据结构》的辅助用书,也可供从事程序设计的工程技术人员参考使用。

图书在版编目(CIP)数据

C++与数据结构实验教程/苏京霞,高飞编著 .—北京:北京理工大学出版社,2006.8

高等院校实验课教材

ISBN 7-5640-0854-7

Ⅰ.C... Ⅱ.①苏...②高... Ⅲ.①C语言-程序设计-高等学校-教材②数据结构-高等学校-教材 Ⅳ.①TP312②TP311.12

中国版本图书馆CIP数据核字(2006)第096708号

出版发行 / 北京理工大学出版社
社 址 / 北京市海淀区中关村南大街5号
邮 编 / 100081
电 话 / (010)68914775(办公室) 68944990(批销中心) 68911084(读者服务部)
网 址 / http://www.bitpress.com.cn
经 销 / 全国各地新华书店
印 刷 / 北京国马印刷厂
开 本 / 787毫米×1092毫米 1/16
印 张 / 15.25
字 数 / 359千字
版 次 / 2006年8月1版 2006年8月第1次印刷
印 数 / 1~5000册 责任校对 / 陈玉梅
定 价 / 23.00元 责任印制 / 刘京凤

前　言

本书是和《C++与数据结构》（北京理工大学出版社出版）一书配套使用的参考用书。关于《C++与数据结构》一书的特点和编写该书的初衷，已在《C++与数据结构》的前言中做了比较详细的说明。

《C++与数据结构》一书中既包括基于对象和面向对象的 C++程序设计，也包括数据结构与算法设计。C++是为了解决大型软件开发过程中的问题而产生的，然而在学习的初级阶段不可能就接触大型程序，或编写大型程序，所以必须从简单的小程序开始，循序渐进，逐步深入。《C++与数据结构》课程力求以算法为中介，使学习程序语言和学习数据结构并进，本书的程序例题和实验题目正是为这个目的设计的。

本书内容包括两个部分，共 15 章。

第 1 部分从第 1 章到第 6 章，对应《C++与数据结构》的面向对象的程序设计基础。第 1 章到第 6 章所提供的程序例题和实验题目仅供教师安排实验参考。教材每一章内容的课时不同，所以相应的实验课时应该有所不同。不同的学校、不同的专业可适当地选择实验的内容或增加其他的实验内容。第 7 章简单介绍了 Visual C++ 6.0 集成开发环境，供读者在其上编写、调试和运行自己开发的程序。如果需要对 Visual C++ 6.0 开发环境做进一步的了解，可参阅相关文献。

第 2 部分从第 8 章到第 15 章，对应《C++与数据结构》的数据结构部分。该部分在有些章节中所提供的程序例题和实验题目没有采用模板机制，有些章节的则采用了模板机制。可根据学习的需要对模板和非模板进行转换。

本书由苏京霞、高飞编著。在本书的编写过程中，课程组的聂青、薛艳明、李慧芳提出了许多宝贵的建议；本书的出版，得到了北京理工大学出版社的大力支持和热心帮助，在此一并表示衷心的感谢。

由于编写者水平有限，不足之处在所难免，诚恳地希望读者批评指正。

编　者

目　录

第 1 部分　面向对象的 C++程序设计基础

第 1 部分　面向对象的 C++程序设计基础

第 1 章　C++类及其对象的封装性

1.1　实验目的和要求

1. 了解 C++类的构成；
2. 掌握声明类的方法、类和类的成员的概念以及定义对象的方法；
3. 掌握成员函数的定义；
4. 熟悉构造函数和析构函数；
5. 了解 C++的封装性。

1.2　C++ 概 述

C++是从 C 语言发展演变而来的，并在 C 语言基础上增加了面向对象程序设计的功能，它适合用于编制复杂的大型软件系统。

1.2.1　C++基本概念

C++语言包括过程性语言部分和类部分。过程性语言部分与 C 并无本质的差别，只是功能增强了。类部分是 C 中所没有的，它是面向对象程序设计的主体。

1. 对象

对象是 C++面向对象程序设计的核心，在面向对象方法中，对象是指系统中用来描述客观事物的一个实体，是对象程序设计的基本单元。对象由一组属性和一组行为构成。

2. 类

C++中的类是指在所要处理问题中的若干对象的共同行为和不同状态的集合。一个类的所有对象都有相同的数据结构，并且共享相同的实现操作的代码。

3. 消息

消息是对象之间相互请求或相互协作的途径，是要求某个对象执行其中某个功能操作的规格说明。对象之间的联系只能通过相互传递消息来进行。消息的具体表现为函数。

4. 封装

封装就是把对象的属性和服务结合成一个独立的系统单位，并尽可能隐藏对象的内部细节。封装的特性如下。

(1) 对于对象的所有私有数据和内部程序外界是不可直接访问的。

(2) 具有外部接口，这个接口描述了对象之间相互请求和响应的消息格式和功能。

(3) 对象内部的代码实现是隐藏的，即外部对象不能直接修改有关的代码细节。

5. 继承性

继承是面向对象系统的一个重要特性，若一个特殊类的对象拥有其一般类的全部属性与服务，则称特殊类是对一般类的继承。

6. 多态性

多态性是指在一般类中定义的属性或行为被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为。

1.2.2 C++程序开发过程

C++程序开发过程与其他高级语言源程序开发过程一样，都必须先经过编辑、编译和连接过程，最后生成可执行的代码后才能运行。程序开发过程的基本步骤如下。

(1) 编辑：编辑是将编写好的 C++源程序输入到计算机中生成磁盘文件的过程。磁盘文件名要用扩展名.cpp，例如：File.cpp。

(2) 编译：编译是通过编译器将程序的源代码（.cpp）转换成为机器代码的形式，称为目标代码，即机器语言指令，这时产生的文件扩展名为. obj。如：将源文件 File.cpp 编译后，生成目标文件 File.obj。但这仍然不是一个可执行的程序，因为目标代码只是一个个的程序块，需要将其连接成为一个适应操作系统环境的程序整体。为了把它转换为可执行程序，必须进行连接。

(3) 连接：连接是编译的最后一个过程，它将用户程序生成的多个目标代码文件和系统提供的库文件（.lib）中的某些代码连接在一起，经过这种连接处理，生成一个可执行文件，存储这个可执行文件的扩展名为.exe。如：File.obj 文件经连接处理后，生成可执行文件 File.exe。

(4) 运行：最后是运行程序。运行可执行文件的方法很多，一般在编译系统下有运行功能，可通过选择菜单项的方式实现。

程序在被编译、连接的过程中，难免会出现错误，这时系统会给出致命错误和警告错误的提示，这时需要返回上一步，对源程序进行修改，直至没有错误提示为止。对于程序中存在的致命错误必须要进行修改，否则，系统不能生成可执行文件。程序中存在警告错误时，虽然系统也会生成可执行文件，但是一般要求修改警告错误后再去运行可执行文件，因为有的警告错误会导致结果的错误。

编译过程主要是进行词法分析和语法分析的过程。程序的运行是对算法的验证。程序运行后，一般会在屏幕上显示出运行结果。用户可以根据运行结果来判断程序是否有算法上的错误。

C++程序开发过程如图 1.1 所示。

1.2.3 C++程序的组成

一个 C++程序由以下基本部分组成。

(1) 预处理命令：C++程序的预处理命令以“#”开头。C++提供了 3 类预处理命令：宏

定义命令、文件包含命令和条件编译命令。如：

```
#include"iostream.h"          //文件包含命令
#define MaxSize 100           //宏定义命令
#ifdef DEBUG                  //条件编译命令
  -----
#endif
```

(2) 输入和输出：C++程序中总是少不了输入和输出的语句，以实现与程序内部的信息交流。特别是屏幕输出的功能，几乎每个程序都要使用。C++语言除了保留了 C 语言标准库中的各种输入和输出函数外，还提供了一套新的输入和输出函数——流式输入输出。例如：

```
cout <<"Hello，How are you！";
cin >> name;
```

(3) 函数：在 C++程序中，每个程序是由若干个函数组成的，在组成一个程序的若干个函数中，必须有一个并且只能有一个是主函数 main（）。执行程序时，系统先找主函数，并且从主函数开始执行，其他函数只能通过主函数或被主函数调用的函数执行。

C++中函数原型应该包括：函数的返回值类型、函数名和函数的各个参数的类型，希望读者在程序设计中使用完整的函数原型，养成良好的习惯。例如：

```
void main（）          //函数没有返回值
int Max（int，int）    //函数返回值为整型
```

(4) 语句：语句是组成程序的基本单元。函数是由若干条语句组成的。语句由单词组成，单词间用空格符分隔，以分号结束。

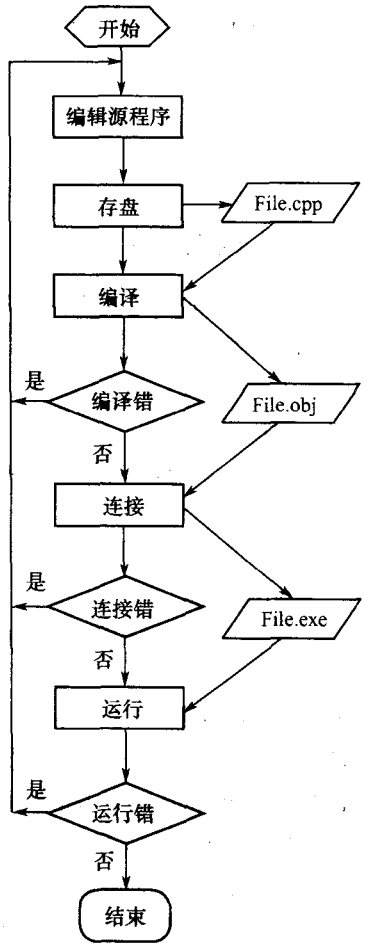

图1.1　C++程序开发过程

C++语句有表达式语句，复合语句、分支语句、循环语句、转向语句和空语句等。

(5) 变量：多数程序都需要说明和使用变量。在 C 语言中，变量的定义必须出现在函数或复合语句的最前面，而在 C++程序中，变量的定义可以出现在任何位置，例如：可以用下面的方式写循环语句。

```
for（int i = 0; i < n; i++）
    for（int j = 0; j < i; j++）{...; }
```

下面是一个完整的可执行程序。

```
#include < iostream.h >
void main（）
  {
    int x，y;
    cout <<"请输入数据：";
```

```
    cin >> x >> y;
    cout <<"x+y ="<< x+y << endl;
}
```

1.3 C++类的构成

1.3.1 类的定义

在C++语言中，定义类的一般形式是：

```
class   类名
  {
    private:
            //私有类型数据成员和成员函数
    public:
            //公有类型数据成员和成员函数
    protected:
            //保护类型数据成员和成员函数
  };
<各个成员函数的实现>;
```

C++语言的类对于不同的成员（数据成员和函数成员）定义了 3 个可视的层次，分别是 private 段、protected 段和 public 段。

在 private 段中，只有类中的成员函数可以访问该段的成员，当最先说明它时为缺省；在 protected 段中，只有类中的成员函数和它的派生类可访问段中的成员；在 public 段中，类中的成员函数、类中的对象、派生类的成员函数及派生类中的对象都可访问该段的成员。

例：

```
class  CArea                          //声明一个类
    {
      private:                        //私有部分
        int x，y，area;               //数据成员
      public:                         //公有部分
        void squarea（int vx，int vy）; //成员函数
        void print（）;
    };
```

1.3.2 成员函数的定义

定义成员函数的一般形式如下：

```
返回值类型   类名::成员函数名（参数表）
  {
    函数体
  }
```

其中，“::”是作用域运算符。

例：void CArea:: square（int vx，int vy）

```
{
  x = vx; y = vy;
  area = x*y;
}
```

其中，“CArea:: square”表示 square 函数是 CArea 类的成员。

1.3.3　构造函数与析构函数

1. 构造函数

构造函数的作用就是在对象被创建时利用特定的值构造对象，将对象初始化为一个特定的状态，使此对象具有区别于其他对象的特征。构造函数在对象被创建时由系统自动调用。

构造函数应具有以下特点，并遵循下列规则。

(1) 构造函数名与类的名称必须相同，一个类的构造函数可以重载。

(2) 构造函数没有返回值，因此也没有函数类型的说明。

(3) 类中既可包含多个构造函数，也可没有构造函数，此时编译系统会自动使用缺省值。定义多个构造函数时，在参数的个数或类型上必须存在差异，否则系统调用时会出现二义性。

(4) 复制构造函数可利用已有的对象来建立一个类中的新对象。

(5) 构造函数是由系统自动调用来给对象进行初始化的。

例: 类中声明的各种构造函数。

```
class CArea
  {
      //...
      public:
          CArea（）{x = 0; y = 0; }          //声明无参数的构造函数
          CArea（int rx，int ry）            //声明带两个整型数据的构造函数
          CArea（float rr）{r = rr; }        //声明带一个浮点参数的构造函数
          CArea（float rr，char ra[ ]）      //声明带两个参数的构造函数
  };
```

2. 析构函数

在 C++语言中，析构函数是用来释放一个对象的。在一般情况下，当一个对象的生存期结束时，系统将自动调用析构函数将它释放。析构函数应具有以下特点，并遵循下列规则。

(1) 析构函数的名称必须是“~”符号加上类名称。

(2) 析构函数没有返回值，因此也没有函数类型说明。

(3) 一个类仅有一个析构函数，此外，如果类中未定义析构函数，那么系统会自动加上一个完全不做任何事的析构函数。

(4) 析构函数没有参数。

析构函数也是成员函数，它在取消某个对象时起作用。

1.3.4 对象定义格式

对象定义的一般形式为：

<类名> <对象名>（<初值表>）;

对象成员表示方法为：

<对象名>.<成员名>（<参数表>）或者 <对象指针名>→<成员名>（<参数表>）

例：定义一个日期类 CDate，再创建一个生日对象。

```
class  CDate
  {
        int Year;
        int Month;
        int Day;
     public:
        void SetDate (int, int, int);
        void ShowDate ();
    }myBirthday;                    //定义类的同时创建对象 myBirthday
```

要创建 CDate 类的对象 myBirthday，还可以像下面这样定义：

```
CDate  myBirthday;
```

1.3.5 类的封装性

类的封装性是指将数据和操作数据的行为结合起来，构成一个不可分割的整体，这种结合就是封装。在这个整体中，一些成员是受保护的，它们被有效地屏蔽，不能直接访问，另一些成员是公共的，作为对外部的接口。

下面来看一下时钟类的封装情况，如图 1.2 所示。

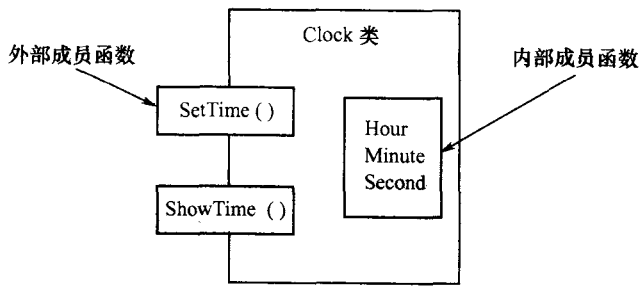

图1.2 Clock类的描述

内部数据成员与外部成员函数构成了 Clock 类。在该类中，时、分、秒被屏蔽，只能通过两个外部成员函数才能访问。

1.4 程 序 例 题

例题 1.1　显示日期。

程序实现：

```
#include   "iostream.h"
 class   tdate
   {
      private:                                    //成员函数声明
         int month;
         int day;
         int year;
       public:
         void set（int，int，int);                //设置日期值
         int   isleapyear（);                     //判定是否为闰年
         void print（);                           //输出日期值
   };
void   tdate:: set（int m，int d，int y）
  {
       month = m;
       day = d;
       year = y;
    }
int tadte:: isleapyear（）
  {
     return（year % 4 == 0 && year % 100!= 0 || year % 400 == 0);
      void print（）
         {cout << month <<"/"<< day <<"/"<< year << endl;  }
   }
void main（）
  {
       tdade a;                                   //定义类对象
       a.set（2，4，2006);
       a.print（);
   }
```

程序运行结果：

```
2/4/2006
```

例题 1.2　显示时钟。

程序实现：

```
#include  "iostream.h"
class  Clock                          //定义时钟类
  {
      private:
          int Hour，Minute，Second;
      public:
          Clock（）;                          //构造函数
          void SetTime（int newh，int newm，int news）;
          void ShowTime（）;
  };
Clock:: Clock（）                          //构造函数实现
  {
      Hour = 0;
      Minute = 0;
      Second = 0;
  }
void Clock:: SetTime（int newh，int newm，int news）
  {
      Hour = newh;
      Minute = newm;
      Second = news;
  }
void Clock:: ShowTime（）
  {
     cout << Hour <<": "<< Minute <<": "<< Second << endl;
  }
void main（）
  {
      Clock  myclock;
      cout <<"First time set and output: "<< endl;
      myclock.ShowTime（）;
      cout <<"Second time set and output: "<< endl;
      myclock.SetTime（8，30，30）;                    //设置时间为 8:30:30
      myclock.ShowTime（）;
  }
```

程序运行结果：

```
First time set and output:
0:0:0
Second time set and output:
8:30:30
```

例题 1.3　自定义拷贝构造函数。

程序实现：

```
#include < iostream.h >
#include < string.h >
class String                          //声明一个字符串类
  {
        int length;                   //字符串长度
        char *str;                    //指向字符串指针
    public:
        String（char *）;             //声明构造函数
        String（const String &p）;    //声明自定义拷贝构造函数
        void Show（）                 //显示字符串
         {
            cout << str << endl;
         }
  };
String:: String（char *s）
  {
      if（s）
        {
            length = strlen（s）;
            str = new char[length+1];
            strcpy（str，s）;
        }
      else
        {
            length = 0;
            str = 0;
        }
  }
String:: String（const String &p）
  {
      length = p.length;
      str = new char[length+1];
      strcpy（str，p.str）;
  }
void main（）
  {
      String str1（"This a constructing. "）;
```