计算机基础程序设计题典丛书

C++语言程序设计题典

李春葆　曾 平　刘 斌　　编著

清 华 大 学 出 版 社

http://www.tup.tsinghua.edu.cn

（京）新登字 158 号

内 容 简 介

C++语言是计算机科学及相关专业的重要基础课程。本书以程序的方式介绍 C++语言的基本内容和特点，涵盖了 C++面向对象程序设计部分。很多难以用文字描述的复杂概念通过一个简单的程序就得以清楚地说明。

全书分为 10 章，从第 1 章到第 9 章以专题的方式介绍了 C++语言的各组成成分，每章均由内容要点、例题解析、习题实践和参考答案 4 部分组成。第 10 章是综合应用，给出了 5 个典型例子，全面涵盖了 C++语言中面向对象程序设计的概念和技术。本书所有的程序均在 Visual C++ 6.0 环境中调试通过。

本书可作为计算机科学及相关专业的本、专科学生学习 C++语言课程的参考书，也适合考研和计算机等级水平考试者。

图书在版编目（CIP）数据

书　　名：C++语言程序设计题典
作　　者：李春葆　曾 平　刘 斌
出 版 者：清华大学出版社（北京清华大学学研大厦.邮编 100084）
http://www.tup.tsinghua.edu.cn
责任编辑：朱起飞
印 刷 者：北京市耀华印刷有限责任公司
发 行 者：新华书店总店北京发行所
开　　本：787×1092　1/16　　印张：19.375　　字数：471 千字
版　　次：2002 年 6 第 1 版　　2002 年 6 月第 1 次印刷
书　　号：ISBN 7－302－ 05686-2
印　　数：0001~5000
定　　价：28.00 元

《计算机基础程序设计题典丛书》序

计算机软件产业是IT行业最前沿和最有前途的产业之一，也最富有竞争性和挑战性。随着WTO的加入，我国越来越重视软件产业的发展。为了应对挑战，关键是加强软件人才的培养，目前全国有33所大学设立了教授软件开发的专业学院。

“软件＝程序＋文档”，程序设计是软件开发的基础。一个优秀的软件开发人才必须首先从程序设计开始培养和训练起步。试想一下，如果没有基本的程序设计思想和技术，怎么可能成为一个优秀的软件开发工程师呢？

本丛书就是面向培养软件开发人才而提供的基础性程序设计教程，它以程序设计为核心，涵盖了一些重要的基础性程序设计科目。本套丛书包括：

1. 《C语言程序设计题典》
2. 《C＋＋语言程序设计题典》
3. 《数据结构程序设计题典》
4. 《Visual Basic程序设计题典》
5. 《Visual C++程序设计题典》

本丛书具有以下特点：

○ 强化程序设计的实现过程

要想成为一个程序设计工程师，必须深入领会程序设计的奥秘，这样才能做到触类旁通。所以本丛书不同于一般的软件系统使用手册，仅列出了命令的使用语法，也不同于一般的语言教科书，多注重于抽象的理论方法，而是经过仔细的挑选，从程序实现的角度精心设计出一系列富有代表性的程序设计题目，并给予完整的解析。

○ 内容叙述力求通俗易懂

本丛书不仅面向计算机专业的本、专科学生，更立足于希望通过基础程序设计训练，继而成为计算机程序设计工程师的有志青年。因此，在文字叙述和内容的安排上尽量通俗易懂，力求讲出题目设计思想的来龙去脉，把程序设计的整个“过程”讲透。

○ 结构布局完整、合理，便于自学

丛书采用统一的结构，以章为单位，每章讨论一个专题，均由内容要点、例题解析、习题和参考答案等构成。所有习题都附有参考答案。例题的求解过程详尽，习题参考答案简略，但可以从相应的例题中找到程序设计思路，因而适合通过自学提高程序设计能力。

本丛书的作者都是长期从事高校计算机基础教学工作的教授和副教授，他们集教学和程序设计经验于一炉，精心编写了这套丛书，相信会为我国软件人才培养起促进作用。

李春葆

2002.1.5

前　言

C＋＋语言是计算机科学及相关专业的重要基础课程。如果说C语言作为编程入门课程，介绍过程式程序设计方法的话，C＋＋语言则提供了面向对象程序设计风格，更接近于常用的软件开发平台（如VC＋＋）的软件开发思想。目前有很多C＋＋语言教材，其中相当一部分篇幅介绍C语言程序设计，考虑到本丛书中已出版了《C语言程序设计题典》，为了避免重复，本书只涵盖了C＋＋面向对象程序设计部分。本书的主要特点是以程序的方式介绍C＋＋语言的基本内容和特点，很多难以用文字描述的复杂概念通过一个简单的程序就得以清楚地说明。

全书分为10章，从第1章到第9章以专题的方式介绍了C＋＋语言的各组成成分，每章分为内容要点、例题解析、习题实践和参考答案4小节。第1章介绍类和对象的概念；第2章介绍引用；第3章介绍友元；第4章介绍重载函数和重载运算符；第5章介绍模板；第6章介绍继承与派生；第7章介绍多态性；第8章介绍I／O流库的使用；第9章介绍异常处理方法；第10章是综合应用，给出5个典型例子，全面涵盖了C++语言中面向对象程序设计的概念和技术。

本书所有的程序均在Visual C++ 6.0环境中调试通过，在其他版本如5.0中一般也都可以运行。

本书的编写得到了武汉大学计算机学院的大力支持，北京科海培训中心的夏非彼高级工程师提出了许多富有启发的建议。作者在此表示衷心的感谢。

本书可作为计算机科学及相关专业的本、专科学生学习C＋＋语言课程的参考书，也适合考研和计算机等级水平考试者。

由于水平所限，尽管编者不遗余力，仍可能存在错误和不足之处，敬请读者批评指正。

编 者

2002.2.15

目　录

第1章　类和对象 1
1.1　内容要点 2
1.1.1　知识点 2
1.1.2　内容概要 3
1.2　例题解析 6
1.3　习题实践 17
1.4　参考答案 28
第2章　引用 54
2.1　内容要点 55
2.1.1　知识点 55
2.1.2　内容概要 55
2.2　例题解析 56
2.3　习题实践 64
2.4　参考答案 68
第3章　友元 75
3.1　内容要点 76
3.1.1 知识点 76
3.1.2　内容概要 76
3.2　例题解析 77
3.3　习题实践 87
3.4　参考答案 90
第4章　重载 99
4.1　内容要点 100
4.1.1　知识点 100
4.1.2　内容概要 100
4.2　例题解析 102
4.3　习题实践 111
4.4　参考答案 115

第5章　模板 123
5.1　内容要点 124
5.1.1　知识点 124
5.1.2　内容概要 124
5.2　例题解析 126
5.3　习题实践 130
5.4　参考答案 132
第6章　继承和派生 143
6.1　内容要点 144
6.1.1　知识点 144
6.1.2　内容概要 144
6.2　例题解析 146
6.3　习题实践 158
6.4　参考答案 165
第7章　多态性 193
7.1　内容要点 194
7.1.1　知识点 194
7.1.2　内容概要 194
7.2　例题解析 196
7.3　习题实践 204
7.4　参考答案 209
第8章　I／O流库 222
8.1　内容要点 223
8.1.1　知识点 223
8.1.2　内容概要 223
8.2　例题解析 225
8.3　习题实践 234
8.4　参考答案 236
第9章　异常处理 256
9.1　内容要点 257
9.1.1　知识点 257
9.1.2　内容概要 257
9.2　例题解析 259
9.3　习题实践 261

9.4 参考答案 262
第10章 综合应用 266
参考文献 301

第 1 章　类和对象

1.1 内容要点

1.1.1 知识点

1. 类和对象的概念

- 类的定义
- 对象的定义
- 成员的访问权限

2. 构造函数和析构函数

- 构造函数的定义
- 重载构造函数
- 构造函数的调用方式
- 析构函数的定义
- 析构函数的调用方式

3. 对象指针和对象数组

- 对象指针的定义和使用
- 对象数组的定义和使用

4. 成员指针

- 数据成员指针的定义和使用
- 成员函数指针的定义和使用

5. 静态成员

- 静态数据成员的定义和使用
- 静态成员函数的定义和使用

6. 嵌套类

- 嵌套类的定义和使用
- 嵌套类中构造函数的调用顺序

7. 类和对象的应用

1.1.2　内容概要

1. 类的定义

类是C++封装的基本单元，它把数据和函数封装在一起。其一般定义格式如下：

```
class  <类名>
{
private:
   <私有数据成员和成员函数>;
protected:
   <保护数据成员和成员函数>;
public:
   <公有数据成员和成员函数>;
};
<各个成员函数的实现>;
```

其中，class是定义类的关键字。<类名>是一个标识符，用于惟一标识一个类。一对大括号内是类的说明部分，说明该类的所有成员。类的成员包含数据成员和成员函数两部分。类的成员从访问权限上分有以下三类：公有（public）、私有（private）和保护（protected），其中默认为private权限。

2. 访问权限

类成员有三类访问权限：公有（public）、私有（private）和保护（protected）。说明为公有的成员可以被程序中的任何代码访问；说明为私有的成员只能被类本身的成员函数及友元类的成员函数访问，其他类的成员函数，包括其派生类的成员函数都不能访问它们；说明为保护的成员与私有成员类似，只是除了类本身的成员函数和说明为友元类的成员函数可以访问保护成员外，该类的派生类的成员也可以访问。

3. 对象的定义格式

定义对象的格式如下：

```
<类名> <对象名表>;
```

其中，<类名>是指定对象所属的类的名字，即所定义的对象是该类的对象。<对象名表>中可以有一个或多个对象名，多个对象名用逗号分隔。在<对象名表>中，可以是一般

的对象名，还可以是指向对象的指针名或引用名，也可以是对象数组名。

4. 构造函数

构造函数是类的一个特殊的成员函数，它与类同名，并且没有返回值。C++在创建一个对象时，会自动调用类的“构造函数”，在构造函数中可以执行初始化成员变量的操作。

构造函数可以像普通函数一样被重载，C++根据说明中的参数个数和类型选择合适的构造函数。

5. 析构函数

一个类可能在构造函数里为对象分配资源，这些资源需要在对象不复存在以前被释放。例如，如果构造函数中为对象分配了内存，这块内存在对象消失之前必须被释放。

与构造函数对应的是析构函数。当一个对象消失，或用new创建的对象用delete删除时，由系统自动调用类的析构函数。析构函数名字为在构造函数前加符号“~”，析构函数没有参数和返回值。一个类中只可能定义一个析构函数，所以析构函数不能重载。

在析构函数中一般做一些清除工作，在C++中，清除就像初始化一样重要。通过析构函数来保证清除的执行。

当对象超出其定义范围时（即释放该对象时），编译器自动调用析构函数。在以下情况下，析构函数也会被自动调用：

- 如果一个对象被定义在一个函数体内，则当这个函数结束时，该对象的析构函数被自动调用。
- 当一个对象是使用new运算符动态创建的，在使用delete运算符删除它时，delete将会自动调用析构函数。

6. 类成员指针

类成员指针有类数据成员指针和类成员函数指针。

（1）类数据成员指针

类数据成员指针的定义格式如下：

```
<类型><类名>::<指针名>
```

在使用类数据成员指针时，需要首先指定类的一个数据成员，然后，通过类的对象来引用指针所指向的成员。

（2）类成员函数指针

类成员函数指针的定义格式如下：

```
<类型>(<类名>::*<指针名>)(<参数表>)
```

给类成员函数指针赋值的格式如下：

<指向函数的指针名>=<函数名>

程序中使用指向函数的指针调用函数的格式如下：

(*<指向函数的指针名>)(<实参表>)

（3）this指针

类的每一个成员函数都有一个隐含定义的常量指针，我们把它称为this指针。this指针的类型就是成员函数所属的类的类型。当调用成员函数时，它被初始化为被调用函数所在的类实例的地址。

this指针只能在类的成员函数中使用，它指向该成员函数被调用的对象。this指针一般用于返回当前对象自身。

7. 对象数组

对象数组是指数组元素为对象的数组。该数组中若干个元素必须是同一个类的若干个对象。对象数组的定义、赋值和引用与普通数组一样，只是数组的元素与普通数组不同，它是同类的若干个对象。

对象数组定义格式如下：

<类名><数组名>[<大小>]...

其中，<类名>指出该数组元素是属于该类的对象，方括号内的<大小>给出某一维的元素个数。一维对象数组只有一个方括号，二维对象数组要有两个方括号，等等。例如：

```
Sample Array[5];
```

表明Array是一维对象数组名，该数组有5个元素，每个元素都是类Sample的对象。其元素从Array[0]到Array[4]。

8. 静态成员

静态成员的提出是为了解决数据共享的问题。在类中，静态成员分为静态数据成员和静态成员函数。

（1）静态数据成员

静态数据成员是类的所有对象共享的成员，而不是某个对象的成员。使用静态数据成员可以节省内存，因为它是所有对象所公有的，因此，对多个对象来说，静态数据成员只存储在一处，供所有对象共用。静态数据成员的值对每个对象都是一样的，但它的值是可以更新的。

静态数据成员的使用方法和注意事项如下：

- 静态数据成员在定义或说明时前面加关键字static。
- 静态数据成员初始化与一般数据成员初始化不同。静态数据成员初始化的格式如下：

<数据类型><类名>::<静态数据成员名>=<值>;

- 静态数据成员必须进行初始化。
- 引用静态数据成员时，采用如下格式：

<类名>::<静态数据成员名>

如果静态数据成员的访问权限允许的话(即public的成员)，可在程序中按上述格式来引用静态数据成员。

（2）静态成员函数

静态成员函数和静态数据成员一样，它们都属于类的静态成员，它们都不是对象成员。因此，对静态成员的引用不需要用对象名。在静态成员函数的实现中不能直接引用类中说明的非静态成员，可以引用类中说明的静态成员。如果静态成员函数中要引用非静态成员时，可通过对象来引用。

调用静态成员函数使用如下格式：

<类名>::<静态成员函数名>(<参数表>)

9. 嵌套类

在一个类中定义的类称为嵌套类，定义嵌套类的类称为外围类。

定义嵌套类的目的在于隐藏类名，减少全局标识符，从而限制用户使用该类建立对象。这样可以提高类的抽象能力，并且强调了两个类(外围类和嵌套类)之间的主从关系。

1.2 例题解析

【例1.1】 分析以下程序的执行结果。

```
#include <iostream.h>
#include <stdlib.h>
class Sample
{
  public:
     int x,y;
     Sample() { x=y=0; }
     Sample(int a,int b) { x=a;y=b; }
     void disp()
     {
         cout << "x=" << x << ",y=" << y << endl;
```

```
    }
};
void main()
{
    Sample s1(2,3);
    s1.disp();
}
```

【解】　本题说明了重载构造函数的定义方法。首先定义了一个类Sample，在main()中定义了它的一个对象，定义s1对象时调用其重载构造函数（x=2,y=3），然后，调用其成员函数输出数据成员。所以输出为：x=2,y=3。

注意：构造函数是惟一不能被显式调用的成员函数，它在定义类的对象时自动调用，也称为隐式调用。

【例1.2】　分析以下程序的执行结果。

```
#include <iostream.h>
class Sample
{
    int x,y;
    public:
    Sample() { x=y=0; }
    Sample(int a,int b) { x=a;y=b; }
    ~Sample()
    {
        if (x==y)
            cout << "x=y" << endl;
        else
            cout << "x!=y" << endl;
    }
    void disp()
    {
        cout << "x=" << x << ",y=" << y << endl;
    }
};
void main()
{
    Sample s1(2,3);
    s1.disp();
}
```

【解】 本题说明了析构函数的定义方法。首先定义了一个类Sample，在main()中定义了它的一个对象，定义s1对象时调用其重载构造函数（x=2,y=3），然后，调用其成员函数输出数据成员，最后在退出程序时自动调用析构函数。所以输出为：

```
x=2,y=3
x!=y
```

注意：析构函数在对象的作用域结束时被自动隐式调用。

【例1.3】 分析以下程序的执行结果。

```
#include <iostream.h>
class Sample
{
    int x;
    public:
    Sample(int a)
    {
        x=a;
        cout << "constructing object:x=" << x << endl;
    }
};
void func(int n)
{
    static Sample obj(n);
}
void main()
{
    func(1);
    func(10);
}
```

【解】 本题说明静态对象构造函数的调用情况。由于在func()函数中定义的对象obj是静态对象，故只被构造一次。所以输出为：

```
constructing object:x=1
```

注意：静态对象和静态变量一样，只被构造一次。块作用域的静态变量，在首次进入到定义该静态对象的函数时，构造该静态对象，以后进入该函数时不再构造静态对象。

【例1.4】　分析以下程序的执行结果。

```
#include <iostream.h>
class Sample
{
    int x,y;
    public:
    Sample() { x=y=0; }
    Sample(int a,int b) { x=a;y=b; }
    void disp()
    {
        cout << "x=" << x << ",y=" << y << endl;
    }
};
void main()
{
    Sample s(2,3),*p=&s;
    p->disp();
}
```

【解】　本题说明了对象指针的使用方法。这里通过指向对象的指针来调用对象的成员函数。对象指针p指向对象s，p->disp()等价于s.disp()。所以输出为：x=2,y=3。

注意：对象指针与结构体指针一样使用。在定义对象指针时不会调用构造函数。

【例1.5】　分析以下程序的执行结果。

```
#include <iostream.h>
class Sample
{
    public:
    int x;
    int y;
    void disp()
    {
        cout << "x=" << x << ",y=" << y << endl;
    }
};
void main()
```

```
{
    int Sample::*pc;
    Sample s;
    pc=&Sample::x;
    s.*pc=10;
    pc=&Sample::y;
    s.*pc=20;
    s.disp();
}
```

【解】 本题说明了类数据成员指针的使用方法。在main()中定义的pc是一个指向Sample类数据成员的指针。执行pc=&Sample∷x时，pc指向数据成员x，语句s.*pc=10等价于s.x=10（为了保证该语句正确执行，Sample类中的x必须是公共成员）；执行pc=&Sample∷y时，pc指向数据成员y，语句s.*pc=20等价于s.y=20（同样，Sample类中的y必须是公共成员）。所以输出为：x=10,y=20。

【例1.6】 下面是一个类的测试程序，设计出能使用如下测试程序的类。

```
void main()
{
    Test a;
    a.init(68,55);
    a.print();
}
```

其执行结果为：

```
测试结果:68-55=13
```

【解】 本题是要设计Test类，其设计方法很多，这里给出一种解法。Test类包括两个私有数据成员x、y，以及两个公共成员函数init()和print()，前者用于给数据成员赋值，后者用于x、y的减法运算和输出相应的结果。

```
#include <iostream.h>
class Test
{
    int x,y;
    public:
    void init(int,int);
    void print();
```