HZ BOOKS
华章教育

高等院校计算机课程设计指导丛书

C++程序设计课程设计

刘燕君 刘振安 张一叶 编著

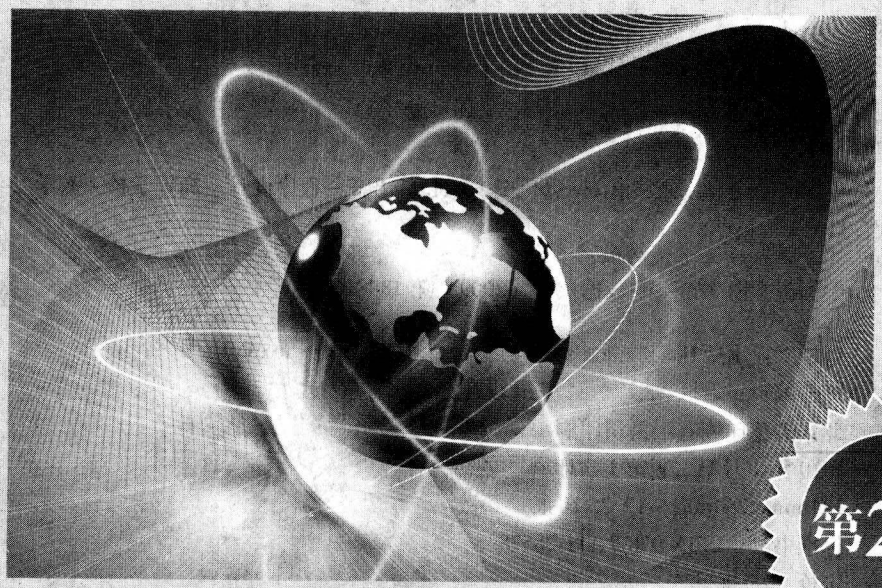
第2版

机械工业出版社
China Machine Press

高等院校计算机课程设计指导丛书

C++程序设计课程设计

刘燕君 刘振安 张一叶 编著



第2版

机械工业出版社
China Machine Press

课程设计可以充分弥补课堂教学和实验中知识深度和广度有限的问题，更好地帮助学生系统地掌握该门课程的主要内容。

本书独立于具体的C++语言教科书，重点放在C++语言面向对象的基本特征上，结合实际应用，通过详细的实例，循序渐进地启发学生完成设计。书中给出的实例完整并通过测试，有的设计还给出测试样例。另外，本书最后还结合课程设计和实际应用需要进行总结以拓宽知识面。

本书不仅是一本很好的课程设计教材，对广大工程技术人员也很有参考价值。

图书在版编目（CIP）数据

C++程序设计课程设计　第2版/刘燕君，刘振安，张一叶编著. —北京：机械工业出版社，2009.11
（高等院校计算机课程设计指导丛书）

ISBN 978-7-111-28578-6

Ⅰ.C…　Ⅱ.①刘…②刘…③张…　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（2009）第188816号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：迟振春　　版式设计：刘永青
北京瑞德印刷有限公司印刷
2010年1月第2版第1次印刷
184mm×260mm · 12.75印张
标准书号：ISBN 978-7-111-28578-6
定价：25.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991；88361066
购书热线：（010）68326294；88379649；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

高等院校
计算机课程设计指导丛书

专家指导委员会

（以姓氏拼音为序）

前　言

语言课程应注重边学边练，但由于课堂教学和实验的深度和广度有限，练习的深度也受到一定限制。为了弥补这一点，特设计了本课程设计。

本课程设计的主要特点如下：

（1）独立于具体的C++语言教科书，重点放在C++语言面向对象的基本特征上，以“不变”应“万变”，涵盖C++语言的重要基础知识。

（2）结合实际应用的要求，使课程设计既覆盖知识点，又接近工程实际需要。通过激发学习兴趣，调动学生主动学习的积极性，并引导他们根据实际编程要求，训练自己实际分析问题的能力及编程能力，养成良好的编程习惯。

（3）通过详细的实例，循序渐进地启发学生完成设计。课程设计将要求、算法和源程序分开，为学生创造独立思考的条件。学生在充分理解要求和算法的前提下，完全可以不按书中提供的参考程序，而设计自己的应用程序。

（4）对同一类型的设计题目，提供不同的解决方案，以拓宽学生的视野。

（5）课程设计分为基本部分与扩展部分，以满足不同学校和不同学生的要求。

（6）提供综合课程设计，以进一步锻炼学生使用面向对象方法思考问题的能力及动手能力。这些综合实验还可以供学生分工合作，以培养团队协作精神。

（7）最后一章结合课程设计题目和实际应用需要进行总结，进一步拓宽知识面。

另外，在实际编程中，为了提高编程质量，对空行、空格和注释均有要求。本书也尽可能地根据实际编程要求给出空行、空格和注释，有时因为标题和页码等实际原因，也会适当减少空行、空格和注释，但希望学生在书写代码时，还是严格按要求处理，以便建立良好的编程风格。

全书共分14章。第0章是课程设计简介；第1章是使用类改写程序；第2章是使用对象编程；第3章是使用对象指针作为函数参数；第4章是菜单设计与出圈游戏；第5章是使用包含和派生设计新的类；第6章是出圈游戏；第7章是使用模板；第8章是虚函数的多态性；第9章是使用循环链表设计出圈游戏；第10章是使用多种方法改写C程序；第11章是综合设计；第12章是研读程序；第13章是课程设计总结。

本书第1版曾被全国许多高等院校采用，有的学校还将其用作毕业设计的参考资料，第2版的编写也得到他们的支持和帮助，在此表示感谢，希望今后继续不吝赐教。

本书由刘燕君主笔，张一叶负责英文校对及资料翻译，最后由刘振安教授统稿。

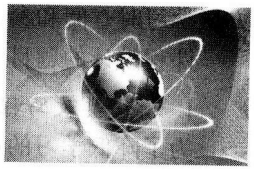
刘燕君

中国科学技术大学

2009年10月8日

目　录

第0章 课程设计简介



由于各校的情况不一，为了便于教师根据本校的特点和教学计划选择相应的课程设计内容，本章简要介绍本书的设计题目及其预期目标。

0.1 课程设计目标

一般来讲，课程设计比教学实验复杂一些，涉及的深度广些并更加接近实用，目的是通过课程设计的综合训练，培养学生实际分析问题、编程和动手的能力，最终目标是想通过这种形式，帮助学生系统掌握该门课程的主要内容，更好地完成教学任务。

自2004年出版《C++程序设计课程设计》以来，虽然多次重印和修订，但该书总体内容偏深，STL的内容过多。在听取了许多读者的意见之后，又编写了这本更接近大多数高等院校教学需要的课程设计。本课程设计偏重基础训练，通过课程设计的题目明确训练的内容，训练由易而难，逐步深入。有些题目（例如出圈游戏）始终贯穿其中，通过不同的方法对其求解，这样可以使读者体会面向对象编程的风格和特点。本课程设计强调主程序的编制方法，方便读者加深对通过类的对象启动程序的理解。

第12章给出一个研读程序，以便读者对面向对象编程有更深入的了解。虽然这个程序比较大，但注释比较清楚，还是比较容易理解的，教师在教学中可以根据情况提出考核标准。例如，可以用读书心得的形式考核，也可以作为团队合作完成的项目，以培养团队协作精神。

0.2 课程设计结构

结构化程序设计使用的是功能抽象，面向对象程序设计不仅能进行功能抽象，而且能进行数据抽象。“对象”实际上是功能抽象和数据抽象的统一。C++语言的“对象”是“类”的实例，程序设计的基础是设计类，所以类的有关概念都是重点，尤其要抓住抽象、封装、继承和多态性等要素。

类设计的重点是选择数据成员和成员函数。成员函数设计的难点是选择函数类型及其参数传递方式。数据类型及程序控制方式仍然是C++语言的基础；数组、指针、类和结构的使用技术是编程的核心技术。学生学习时，常常避开多文件编程和使用文件，但这些都是程序设计员必备的知识，因此本书特意加强了这方面的训练。

学生开始学习面向对象编程时，面对实际问题常常不知道如何选择一个类，而又习惯地回到面向过程的方法编程。为此，本书专门用一章介绍如何用过程编程思想解题，然后使用对象改写这些程序，演示如何设计一个类，通过这个类的对象用自己的成员函数完成一系列的动作，从而实现最终目的，希望通过该章的学习能较好地实现学生思想上的转变。

本书共有12个设计题目和一个课程设计总结。课程设计不再使用一个独立的例子涵盖许多知识点，而是按层次逐步深入。为了使学生理解它们之间如何相互配合，设计要求使用接近实际需要的方式编程。与上一版相比，程序大大简化。前6个设计都很简单，主要是引入面向对象的思维方法。全书多次使用不同方法实现出圈游戏，就是为了通过不同的实现方式，使学生加深对面向对象编程的理解。

本书还单列总结一章，将学习中需要掌握的知识以及测试和调试的基础知识汇总在一起，

方便学习时查阅和参考。

课程设计的题目与章的序号一致，简要说明如下：

1. 使用类改写程序（第1章）

为了降低学习的难度，专门挑选几个面向过程的趣味程序，演示如何设计一个类并产生一个对象，将它们转化为面向对象的程序。这里还从解一元二次方程出发，演示如何抽象一个代表一元二次方程的类，以及如何根据要解决的问题为这个类设计合适的数据成员和成员函数。

2. 使用对象编程（第2章）

本章的设计主要是强调使用类及其成员函数的知识，以便为以后的编程打下基础。

学习C++语言，首先要熟悉编程环境，以便能熟练地使用编程环境编制和调试程序，得到正确的可执行文件。本设计通过给出类的声明，演示按是否有头文件两种情况来实现程序，从而帮助读者熟悉编程规范，为以后的学习打下基础。

3. 使用对象指针作为函数参数（第3章）

动态内存分配是学习的难点之一，本章将通过为类对象进行动态内存分配演示使用指针的技巧，并熟悉跟踪程序和观察程序中变量和对象值的方法，提高程序调试技能。

本课程设计仍然使用一个头文件和一个源文件的方法，以便进一步巩固头文件的设计方法，熟悉编程环境及规范。本章的设计还涉及编制函数及参数传递方式的知识，这都是编程的基本功。

4. 菜单设计与出圈游戏（第4章）

本章通过设计游戏程序介绍菜单设计方法，并进一步介绍多文件设计和更正规的头文件设计方法。

本章使用两种方法设计游戏程序。第一种方法是使用动态数组，目的是进一步加强对动态内存管理的认识。第二种方法是使用向量数组，目的是了解它与普通数组的异同。为了熟悉函数重载的使用方法，这两个游戏程序使用相同的函数名。本章的设计实际上是两个题目，教师可以根据教学要求取舍。

5. 使用包含和派生设计新的类（第5章）

本章的设计任务是分别使用组合和派生的方法设计新类。本课程设计给出两个题目：一个是使用一个Point类产生Line类，另一个是使用一个Point类派生Line类。

通过这个例子，目的是使学生掌握在不同的实现方法中，如何设计相应的构造函数和复制构造函数，进一步理解如何调用它们及析构函数的执行顺序。

这两个题目可以根据教学情况分开做。本章之所以将这两个题目放在一起，是为了通过对比，进一步理解各自的特点。

6. 出圈游戏（第6章）

本章的设计任务是要求分别使用组合和派生的方法编写出圈游戏。第4章中的出圈游戏是定义一个类的外部函数Joseph进行求解，本章则要求使用对象求解，以便加深读者对面向对象解题的理解。

本章的程序属于标准的多文件模式，一个类分别具有自己的头文件和实现文件。

7. 使用模板（第7章）

本章的设计任务是要求编写一个点类模板Point和一个线段类模板Line演示构造函数、复制构造函数、析构函数及其调用顺序。要求分别用组合和派生的方法来实现新的类。

8. 虚函数的多态性（第8章）

前面几章主要是通过设计特定的函数（这些函数使用类作为参数）完成预定任务。本课程

设计的目的是通过产生一个类的对象，通过对象自己的函数成员完成设计任务。

本章的设计任务是设计职工信息表，并由此产生一个信息简表。这里使用数组，利用赋值兼容规则实现简表，并使用虚函数实现多态性，完成显示不同简表信息的任务。

本章还给出使用菜单和向量实现的设计方案，以便进一步理解向量的使用方法，这个要求可以作为选做项目，也可以根据教学需要加以取舍或增加新的要求。

9. 使用循环链表设计出圈游戏（第9章）

本章将使用循环链表再次设计出圈游戏程序，目的是让读者接触简单的链表概念并进一步熟悉面向对象的程序设计思想。另外，本章还涉及简单的读取文件操作，目的是理解字符流的概念。

10. 使用多种方法改写C程序（第10章）

学习C++语言首先要掌握使用类的思想，逐渐使自己从熟悉的面向过程的编程思维方法转到面向对象的思维方法，其中最关键的问题是建立类的概念。

本课程设计的目的是从不同方法入手，使用类来重新设计一个C程序，以便为进一步使用类打下基础。

11. 综合设计（第11章）

本设计的重点是学习使用链表和文件操作，同时进一步熟悉运算符重载。一般来说，编制实用程序都离不开文件存取和运算符重载，所以应该给予它们足够的重视。

本章使用链表实现学生成绩的统计、查询、删除以及文件的存取等操作。

12. 研读程序（第12章）

本课程设计是设计一个实用的小型学生成绩管理程序，要求不用链表而用向量来设计这个程序。它有查询和检索等功能，并且能够对指定文件操作，也可将多个文件组成一个文件。

本设计涉及的知识面较多，由于各校教学要求不一样，也可以根据实际教学情况增删内容。

13. 课程设计总结（第13章）

本章是将学习中需要掌握的知识以及测试和调试的基础知识汇总在一起，供学习时参考。随着学习的深入，经常查看本章的内容，将有助于完成课程设计。

0.3 评价标准

因为本书提供参考程序，所以一般情况下学生都能完成预定设计。如果学生只是按照程序去做，其分数只能在85分以下。为了证明学生已经掌握了设计所涵盖的知识点，应该向学生提一些问题，例如，程序如何实现及其原理等。考虑到各校情况不一，本书没有在各章的“评价标准”中规定必须提问题才算完成设计任务。

另外，程序的可读性均作为正确性的一个方面进行记分，不再单列考核标准。

一般遵循如下规律加以评价：

（1）严格控制90分，其标准是有创意。

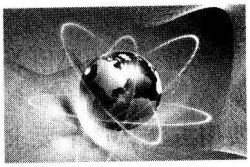
（2）85分以上，要求必须全部正确，并有一定改进或者能正确回答设计中的问题。

（3）有少许失误，可给75～83分。

（4）错误不多，可给60～73分。

（5）没有完成规定的要求，不予及格。

因为课程设计主要是为了锻炼学生，所以应该鼓励他们不要将此作为负担，而是提高钻研问题的兴趣，大胆放手去做。鉴于以上目的，应该以鼓励为主，避免不及格现象。另外，评分标准也可以只设“通过”、“没通过”和“优秀”3档，以激发学生的学习兴趣。



第 1 章
使用类改写程序

学习 C++ 语言，首先要掌握使用类的思想，逐渐使自己从熟悉的面向过程的编程思维方法转到面向对象的思维方法，其中最关键的问题是建立类的概念。

本课程设计的目的是练习使用类来编写程序。本章将从几个面向过程的程序入手，练习使用类来重新设计它们，以便为建立类的概念打下基础。

1.1 真伪问题

已知 4 位同学中的一位数学考了 100 分。当小李询问这 4 位是谁考了 100 分时，4 个人的回答如下：

A 说：不是我。

B 说：是 C。

C 说：是 D。

D 说：他胡说。

已知三个人说的是真话，一个人说的是假话。现在要根据这些信息，找出考 100 分的人。

这是一个逻辑推理题目，要求先使用面向过程的方法编程，然后再用面向对象的方法编程，以便从中体会使用类的思想。

1.1.1 使用面向过程的方法编程

为了解这道题，首先要将 4 个人所说的 4 句话写成关系表达式。定义一种字符型变量 thisman，表示要寻找考了 100 分的人。在程序中写入如下语句：

```
char thisman = '';   //定义字符型变量并将其初始化为空
```

接着让“==”在这里的含义为“是”，让“!=”在这里的含义为“不是”。利用关系表达式可以将 4 个人所说的话表示成表 1-1 所示的关系。注意 D 说的“他胡说”是针对 C 的，所以他的意思是“不是我”，其表达式就是“thisman != 'D'”。

表 1-1 使用关系式表达说话人所说的话

说话人	说话人所说的话	写成关系表达式
A	“不是我”	thisman != 'A'
B	“是 C”	thisman == 'C'
C	“是 D”	thisman == 'D'
D	“他胡说”	thisman != 'D'

在具体赋值时，C++ 中的字符在存储单元中是以 ASCII 码的形式存放的。因此，用赋值语句 thisman = 'A' 和 thisman = 65 是等效的，至于使用哪种方法，取决于程序的具体编制方法。下面给出一种参考程序。

```
#include <iostream>          //预编译命令
using namespace std;
```

```
void main()
{
  int k=0, sum=0, flag=0;
  char thisman='';                          //初始化为空字符
  for(k=1; k<=4; k++)                       //k 既是循环控制变量，也表示第 k 个人
  {
    thisman=64+k;                           //每次换一个人试验
    sum=(thisman!='A')+(thisman=='C')+(thisman=='D')+(thisman!='D');
    if(sum==3)
    {                                       //如果 4 句话有 3 句话为真，则输出该人
      cout<<"考 100 分者为"<<thisman<<endl;
      flag=1;                               //有解标志置 1
    }
  }
  if(flag!=1)                               //无解
    cout<<"无解!"<<endl;
}
```

注意 如果无解，应该给出相应信息。

1.1.2 使用面向对象的方法编程

对上面这类简单的程序，一般不需要使用类就可以简单地求解。这说明对许多简单的问题，可以沿用面向过程的设计方法。其实，正因为简单，初学者往往难以从对象入手解题。本章就是针对这类简单问题，考虑如何使用对象来求解，从而加深对类的理解。

1. 使用类求解的程序

这个程序很简单，可以简单地将主函数中的求解程序改为类的一个成员函数来实现，即修改后的主函数可以具有如下形式：

```
void main()
{
  类名  类的对象;                  //产生一个对象
  类的对象.求解的成员函数;         //求解并输出结果
}
```

假设设计一个 Find 类和一个求解的成员函数 answer，用类产生一个对象 It，则主函数的具体形式如下：

```
void main()
{
  Find It;
  It.answer();
}
```

由此可以写出类的定义如下：

```
class Find{
  public:
    void answer();
};
```

直接将 main 的定义改为成员函数 answer 的定义即可。

```
//answer 函数的定义
void Find::answer()
{
  int k=0, sum=0, flag=0;
```

```
  char thisman = '';                    //初始化为空字符
  for(k = 1; k <= 4; k ++ )             //k 既是循环控制变量，也表示第 k 个人
  {
    thisman = 64 + k;                   //每次换一个人试验
    sum = (thisman! = 'A') + (thisman == 'C') + (thisman == 'D') + (thisman! = 'D');
    if(sum == 3)
    {                                   //如果 4 句话有 3 句话为真，则输出该人
      cout << "考 100 分者为" << thisman << endl;
      flag = 1;                         //有解标志置 1
    }
  }
  if(flag! = 1)                         //无解
    cout << "无解!" << endl;
}
```

2. 完整的源程序

为了简单起见，将类的定义写在前面，最后再写 main 函数的定义。

```
#include < iostream >                   //预编译命令
using namespace std;
//定义类
class Find{

  public:
    void answer();
};
//定义类的成员函数
void Find::answer()
{
  int k = 0, sum = 0, flag = 0;
  char thisman = '';                    //初始化为空字符
  for(k = 1; k <= 4; k ++ )             //k 既是循环控制变量，也表示第 k 个人
  {
    thisman = 64 + k;                   //每次换一个人试验
    sum = (thisman! = 'A') + (thisman == 'C') + (thisman == 'D') + (thisman! = 'D');
    if(sum == 3)
    {                                   //如果 4 句话有 3 句话为真，则输出该人
      cout << "考 100 分者为" << thisman << endl;
      flag = 1;                         //有解标志置 1
    }
  }
  if(flag! = 1)                         //无解
     cout << "无解!" << endl;
}
//主函数
void main()
{
  Find It;
  It. answer();
}
```

程序运行输出结果如下：

```
考 100 分者为 C
```

3. 使用构造函数初始化的程序

可以将上面的程序改为使用构造函数初始化，然后使用成员函数求解。尽管这道题目过于简单，似乎有点小题大做，但为了了解类的使用方法，我们还是有意将它复杂化。

使用面向对象的方法一般需要考虑如下问题：

（1）设计一个类。

（2）为这个类选择属性。

（3）设计类的构造函数，以便初始化类的属性。

（4）设计相应的成员函数。

（5）设计用来求解的成员函数。

按照如上思路，假设完全按照前面使用的整数变量来设计类（因为字符变量是计算用的临时变量，所以不作为类的属性），设计一个具有 3 个数据成员的类 Find，求解的成员函数为 answer，则主程序具有如下形式：

```
void main()
{
  Find It(0, 0, 0);
  It.answer();
}
```

由此可以写出它的参考程序。注意原来的语句

```
int k=0, sum=0, flag=0;
```

因为已经在构造函数中实现，所以 answer 函数可以直接使用这些数据成员。

```
//方法二
#include <iostream>                    //预编译命令
using namespace std;
//定义类
class Find{
  private:
    int k, sum, flag;
  public:
    Find(int x, int y, int z)          //定义成员函数
    {
        k=x;
        sum=y;
        flag=z;
    }
    void answer();                     //声明求解的成员函数
};
//定义用来求解的成员函数
void Find::answer()
{
    char thisman=' ';                  //初始化为空字符
    for(k=1; k<=4; k++)                //k 既是循环控制变量，也表示第 k 个人
    {
      thisman=64+k;                    //每次换一个人试验
      sum=(thisman!='A')+(thisman=='C')+(thisman=='D')+(thisman!='D');
      if(sum==3)
      {                                //如果 4 句话有 3 句话为真，则输出该人
        cout<<"考 100 分者为"<<thisman<<endl;
        flag=1;                        //有解标志置 1
      }
    }
    if(flag!=1)                        //无解
      cout<<"无解!"<<endl;
}
```

```
//主函数
void main()
{
  Find  It(0, 0, 0);
  It. answer();
}
```

同理，也可以将字符变量 thisman 在构造函数中初始化，读者可以自己实现这一功能。

1.2 新郎新娘问题

教堂中来了 A、B、C 三个新郎和 X、Y、Z 三个新娘，问新人中的三位，他们互相与谁结婚。下面是三个人的回答，但全是假话，据此判断谁与谁结婚。

A 说他与 X 结婚；

X 说她和 C 结婚；

C 说他与 Z 结婚。

将 A、B、C 三人用 1、2、3 表示，X = 1 表示 X 和 A 结婚，Y! = 1 表示 Y 不能与 A 结婚。他们说的都是假话，所以真实的情况是：

X! = 1　A 不与 X 结婚

X! = 3　X 不和 C 结婚

Z! = 3　C 不与 Z 结婚

不过别忘记另外隐含的一个重要条件：三个新娘互相不能结婚。

```
X! = Y&&X! = Z&& Y! = Z
```

由此可以得出判断的依据为：

```
if(X! = 1&&X! = 3&&Z! = 3&& X! = Y&&X! = Z&&Y! = Z)
```

以新娘进行循环比较，穷举各种可能的情况，代入上述表达式中进行推理运算，符合上述情况的就是正确结果。

求出 X、Y、Z，就可以知道与她们结婚的新郎是谁。因为是以字符 A 为基准进行换算，所以不要忘记减 1。例如，对于 X 而言，与她结婚的是'A' + X - 1。如果还不容易理解，可以直接把'A'，1 作为基准，问题就迎刃而解了。

```
#include < iostream >
using namespace std; >

void main()
{
int  X, Y, Z;                                    //声明 3 个新娘
for(X = 1;   X <= 3;   X ++ )
  for(Y = 1;   Y <= 3;   Y ++ )
    for(Z = 1;   Z <= 3;   Z ++ )
      if(X! = 1&&X! = 3&&Z! = 3&& X! = Y&&X! = Z&&Y! = Z)
      {
        char c = 'A' + X - 1;                    //计算与 X 结婚的新郎
        cout ≪ "X 与" ≪ c ≪ "结婚" ≪ endl;
        c = 'A' + Y - 1;                         //计算与 Y 结婚的新郎
        cout ≪ "Y 与" ≪ c ≪ "结婚" ≪ endl;
        c = 'A' + Z - 1;                         //计算与 Z 结婚的新郎
        cout ≪ "Z 与" ≪ c ≪ "结婚" ≪ endl;
      }
}
```

运行结果为：

```
X 与 B 结婚
Y 与 C 结婚
Z 与 A 结婚
```

下面给出最简单的面向对象程序，读者可以练习使用构造函数初始化参数。

```
//使用对象编程参考程序
#include < iostream >
using namespace std;

class bride{
public:
    bride(){}
    void Find();
};
void bride::Find()
{
int  X, Y, Z;                                    //声明 3 个新娘
for(X = 1;   X <= 3;   X ++ )
  for(Y = 1;   Y <= 3;   Y ++ )
    for(Z = 1;   Z <= 3;   Z ++ )
      if(X! = 1&&X! = 3&&Z! = 3&& X! = Y&&X! = Z&&Y! = Z)
      {
          char c = 'A' + X - 1;                  //计算与 X 结婚的新郎
          cout << "X 与 " << c << "结婚" << endl;
          c = 'A' + Y - 1;                       //计算与 Y 结婚的新郎
          cout << "Y 与 " << c << "结婚" << endl;
          c = 'A' + Z - 1;                       //计算与 Z 结婚的新郎
          cout << "Z 与 " << c << "结婚" << endl;
      }
}

void main()
{
  bride It;
  It. Find();
}
```

1.3 求解一元二次方程

编写一个求方程 $ax^2 + bx + c = 0$ 的根的程序，用三个函数分别求当 $b^2 - 4ac$ 大于零、等于零和小于零时的方程的根。要求从主函数输入 a、b、c 的值并输出结果。使用面向过程的解法一般是把变量设为 double 型，使用库函数 cmath 进行计算，参考程序如下：

```
#include < cmath >
#include < iostream >
using namespace std;

void GTzero(double, double);
void EQzero(double, double);
void  LTzero(double, double);
double x1, x2, d, p, q;
void GTzero(double a, double b)
{
  x1 = ( - b + sqrt(d))/(2 * a);
  x2 = ( - b - sqrt(d))/(2 * a);
}
```