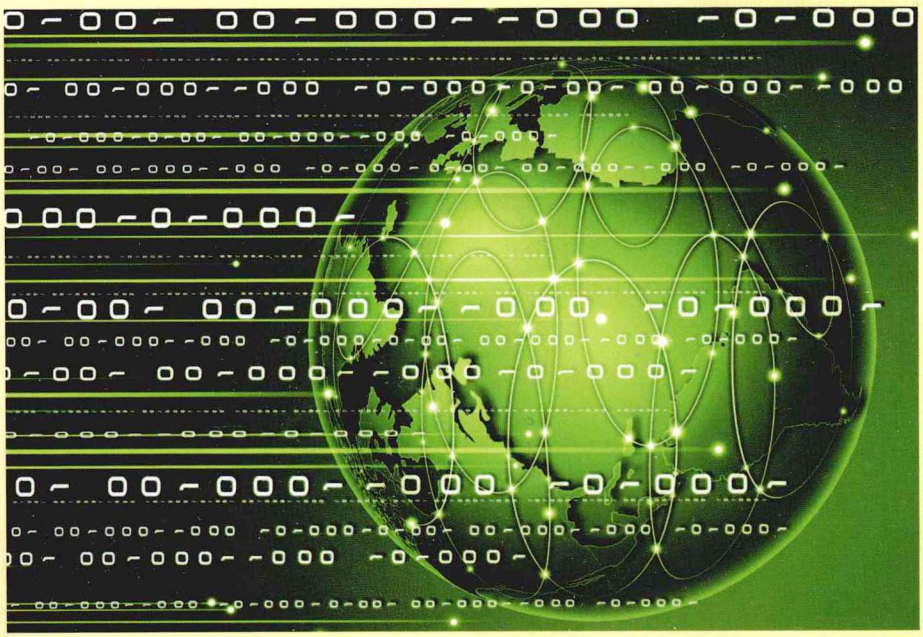
HZ BOOKS
华章教育

高等院校计算机课程设计指导丛书

数据结构课程设计

C++语言描述

刘燕君 苏仕华 刘振安 编著
中国科学技术大学



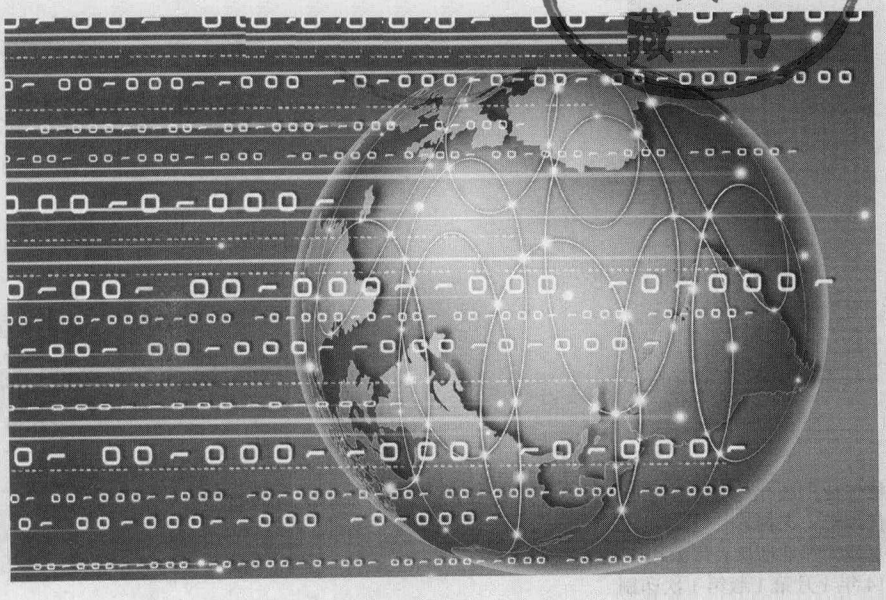
机械工业出版社
China Machine Press

高等院校计算机课程设计指导丛

数据结构课程设计

C++语言描述

刘燕君 苏仕华 刘振安 编著
中国科学技术大学



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

数据结构课程设计：C++ 语言描述 / 刘燕君，苏仕华，刘振安编著 . —北京：机械工业出版社，2014.1
（高等院校计算机课程设计指导丛书）

ISBN 978-7-111-44726-9

Ⅰ. 数… Ⅱ. ① 刘… ② 苏… ③ 刘… Ⅲ. ① 数据结构 – 课程设计 – 高等学校 – 教学参考资料
② C 语言 – 程序设计 – 课程设计 – 高等学校 – 教学参考资料 Ⅳ. ① TP311.12 ② TP312

中国版本图书馆 CIP 数据核字（2013）第 269014 号

本书按照“数据结构”课程的大纲设计相应章节，而且给出知识的重点和难点、典型例题及实验解答。全书共分 11 章，给出了与数据结构内容相关的知识解析、算法分析以及课程设计、描述了相关数据结构的存储表示及其实际应用的操作算法，对用 C++ 模板方法描述的各种算法进行了详细的注释和性能分析，并对各应用的解题思路、方法进行了较详细的分析。

本书取材新颖、结构合理、概念清楚、语言简洁、通俗易懂、实用性强，重在培养学生对各种基本算法的应用技能，特别适合作为高等院校各类相关专业本科生、专科生学习数据结构的辅助教材和实践用书，也可以作为广大从事计算机软件与应用的工程技术人员及社会大众学习数据结构的参考用书。

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
北京诚信伟业印刷有限公司印刷
2014 年 1 月第 1 版第 1 次印刷
185mm × 260mm • 14 印张
标准书号：ISBN 978-7-111-44726-9
定 价：29.00 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 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

前　言

本书按照“数据结构”课程的教学大纲设计相应章节，而且给出知识的重点和难点、典型例题及实验解答。课程设计要比教学实验更复杂一些，涉及的深度也更广一些，而且更加实用，这样就可以通过课程设计的综合训练，培养学生分析问题、解决问题和编程等方面的实际动手能力，帮助学生系统掌握数据结构这门课程的主要内容，更好地完成教学任务。

本课程设计具有如下特点：

1）独立于具体的数据结构教科书，重点放在数据的存储以及在此存储结构上所实现的各种重要和典型的算法上，以较多的应用实例来涵盖数据结构这门课程要求掌握的各类重要基础知识。

2）结合实际应用的要求，使课程设计既覆盖教学所要求的知识点，又接近工程的实际需要。通过实践激发学生的学习兴趣，调动学生学习的主动性和积极性，并引导他们根据实际问题的需求，训练自己实际分析问题、解决问题以及编程的能力。

3）通过详细的实例分析、循序渐进的描述，启发学生顺利地完成设计。课程设计将设计要求、需求分析、算法设计、编程和实例测试运行分开，为学生创造分析问题、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，而设计出更有特色的应用程序。

4）有些课程设计提出了一些需要改进或需要完善的要求，供有兴趣的学生来扩展自己的设计思路，更进一步提高自己的能力和水平。

5）课程设计的内容基本上按课程教学的顺序设计，而且在各章中都增加了重点、难点解析和适当的例题，可让学生循序渐进地学习，尽量避免涉及后续章节的有关知识；而后续的课程设计尽量引用前面的课程设计内容，以便加深学生对知识的理解。

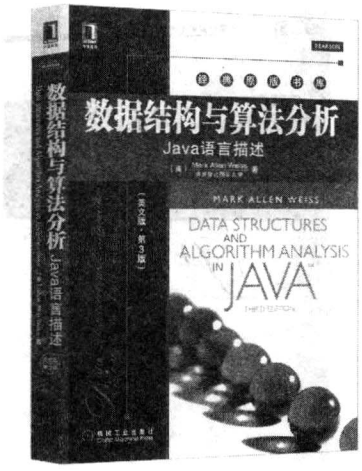
6）课程设计中提供了几个比较大的综合课程设计，以便进一步锻炼学生的动手能力。

本书的编写采取分工负责、集体讨论的方式，具体如下。刘燕君执笔第 3 ~ 5、7、9、11 章，苏仕华执笔第 6、8、10 章，刘振安执笔第 1、2 章并负责统稿。本书编写期间，刘燕君老师去亚洲大学做博士后研究工作，得到导师逢甲大学张真诚教授及亚洲大学资讯学院黄明祥院长的支持，才得以完成所承担的写作任务，在此表示衷心感谢。

由于我们才疏学浅，本书中的不妥之处在所难免，敬请读者不吝赐教，给予指正。

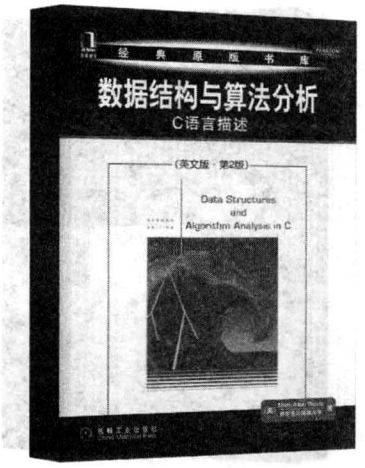
联系方式：zaliu@ustc.edu.cn

推荐阅读



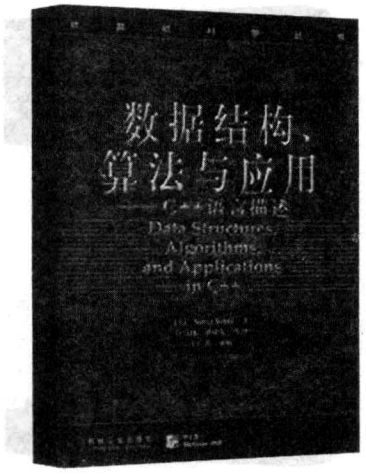
数据结构与算法分析：Java语言描述（英文版·第3版）

作者：Mark Allen Weiss ISBN：978-7-111-41236-6 定价：79.00元.



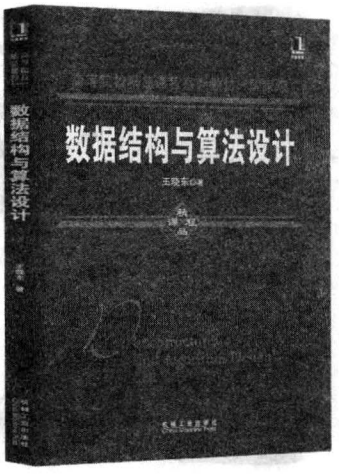
数据结构与算法分析：C语言描述（英文版·第2版）

作者：Mark Allen Weiss ISBN：978-7-111-31280-2 定价：45.00元



数据结构、算法与应用：C++语言描述

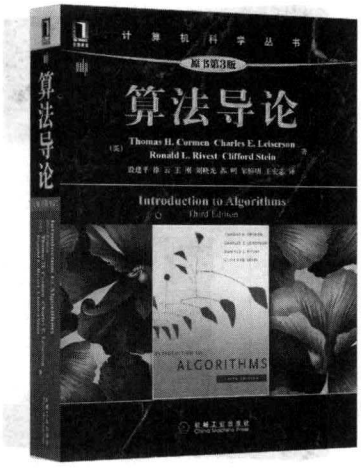
作者：Sartej Sahni ISBN：7-111-07645-1 定价：49.00元



数据结构与算法设计

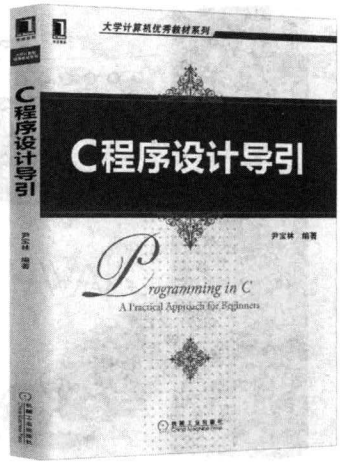
作者：王晓东 ISBN：978-7-111-37924-9 定价：29.00元

推荐阅读



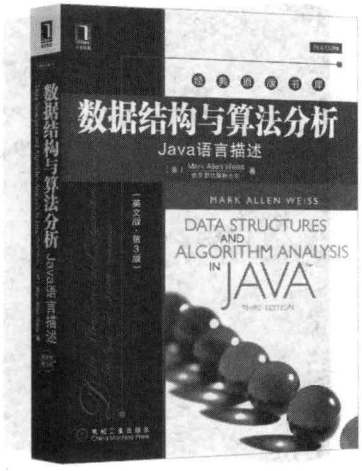
算法导论（原书第3版）

作者：Thomas H.Cormen 等 ISBN：978-7-111-40701-0 定价：128.00元



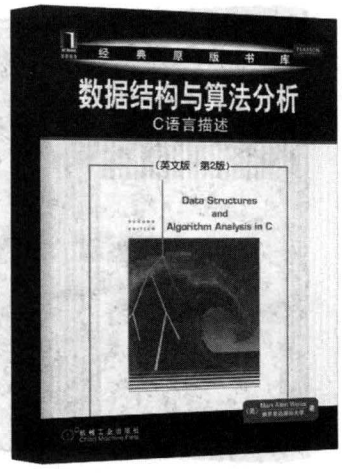
C程序设计导引

作者：尹宝林 ISBN：978-7-111-41891-7 定价：35.00元



数据结构与算法分析
——Java语言描述（英文版·第3版）

作者：Mark Allen Weiss ISBN：978-7-111-41236-6 定价：79.00元

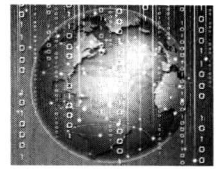


数据结构与算法分析
——C语言描述(英文版·第2版)

作者：Mark Allen Weiss ISBN：978-7-111-31280-2 定价：45.00元

目　录

第 1 章 数据结构概论



数据结构是计算机软件和计算机应用专业的核心课程。本章重点是掌握数据结构的基本概念、常用术语、算法描述和分析的基础知识，并熟悉本书使用的编程语言及编程环境，以便为数据结构课程的学习打下基础。

1.1 本章重点

本章重点是掌握数据结构研究的内容及基本术语，特别要注意如下几个方面的问题：

1）理解数据、数据元素、数据对象、结构和结点的定义，了解没有对数据结构进行定义的原因并掌握数据结构研究的内容。

2）理解描述数据结构所使用的直接前驱、直接后继、开始结点和终端结点的含义，以及数据的逻辑结构的分类（线性结构和非线性结构）。

3）理解数据的存储结构使用的顺序存储方法、链接存储方法、索引存储方法和散列存储方法等 4 种基本存储方法的含义。

4）理解数据类型（data type）是和数据结构密切相关的一个概念，以及引入抽象数据类型的实际意义。

5）掌握时间复杂度的计算方法，理解频度的含义及其计算方法。

1.2 本章难点

本章的难点是正确理解算法的有穷性和可行性的含义、算法与程序的区别以及各种用于描述算法的方法及其利弊，掌握空间复杂度的计算方法。

1.3 求解鸡兔同笼问题实验解答

1.3.1 实验要求

本实验的目的是熟悉编程环境，具体要求如下：

1）熟悉 Microsoft Visual C++ 6.0 编程环境和文件建立方法。

2）在 Microsoft Visual C++ 6.0 环境中，用 C++ 的类来编写并运行鸡兔同笼程序。

3）建立工程 shiyan1，类定义在 shiyan1.h 中，主程序定义在 shiyan1.cpp 中。

1.3.2 参考答案

为了节省篇幅和学习方便，一般的教科书均将类说明和实现及主程序放在一个文件中，甚至大量采用在声明时使用内联函数实现成员函数的方式，这其实是一种不好的习惯。

一般要求是将类的声明放在头文件中，非常简单的成员函数在声明中定义（默认内联函数形式），实现放在 .cpp 文件中，并在 .cpp 文件中将头文件包含进去。如果程序中还有其他

函数，应将其原型在相应的头文件中声明，而将主程序单独编写在一个文件中（这个文件也可以包括与主程序密切相关的函数，但这些函数应该比较简单），这就是多文件编程规范。

在本书的配套教材[⊖]中，使用类模板的方式描述算法，并在头文件中定义算法，所以读者必须尽快熟悉这种方法。

这个实验的目的就是以最简单的程序设计为例，简要说明一个源文件和一个头文件的构成模式，为以后的学习打下基础。

1. 建立工程文件

建立工程 shiyan1 和源程序文件 shiyan1.cpp 均很简单，不再赘述。这里仅简要说明头文件的建立方法。

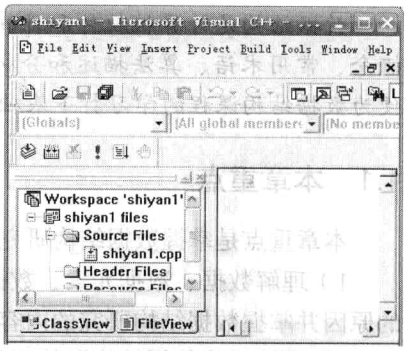


图 1-1 FileView 示意图

1）头文件的建立方法与 .cpp 文件类似。选中图 1-1 中所示的 FileView，进入空项目。单击它，展开树形结构。选中 shiyan1 files 结点，展开向导建立的内容。这时 Header Files 里是空的，没有 C++ 程序的头文件名称。

2）选中图 1-1 中所示的 Header Files 标记，再从 File 菜单中选 New 命令，弹出 New 对话框，如图 1-2 所示。

3）选择图 1-2 中所示的 Files 列表框中的 C/C++ Header File 项，在右边的 File 框中输入 shiyan1（因为默认后缀为 .h，所以不必输入 shiyan1.h）。

4）单击 OK 按钮，得到图 1-3 所示的头文件，在右边的编辑框中编辑头文件 shiyan1.h 即可。

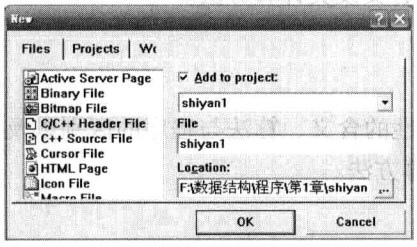


图 1-2 使用 Files 选项卡添加头文件示意图

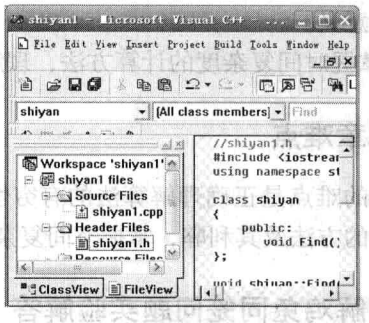


图 1-3 添加和编辑头文件示意图

2. 头文件

```
//shiyan1.h
#include <iostream>
using namespace std;

class shiyan
{
    public:
       void Find();
};
void shiyan::Find()
{
     int sum=0;
```

⊖ 配套教材是指《数据结构：C++ 语言描述》（ISBN 978-7-111-44926-3）一书，该书已由机械工业出版社同步出版。需要说明的是，本课程设计完全可独立使用。——编辑注

```
    for( int i=1;i<24; i++)
    {
            sum++;
            for( int j=35-i;j<13;j++)
            {
               sum++;
               if((i+j==35)&&(2*i+4*j==94))
                      cout<<" 鸡有 "<<i<<" 只，兔有 "<<j<<" 只 "<<endl;
            }
    }
   cout<<" 一共循环 "<<sum<<" 次。"<<endl;
}
```

3. 主程序文件和运行结果

```
// 鸡兔同笼
//shiyan1.cpp
#include "shiyan1.h"

void main()
{
    shiyan obj;
    obj.Find();
}
```

程序运行结果如下：

```
鸡有 23 只，兔有 12 只
一共循环 24 次。
```

1.4 百钱买百鸡问题课程设计

问题：假设每只母鸡值 3 元，每只公鸡值 2 元，两只小鸡值 1 元。现用 100 元钱买 100 只鸡，能同时买到母鸡、公鸡、小鸡各多少只？

1.4.1 设计要求

1）通过使用 C++ 编程求解，熟悉 Microsoft Visual C++ 6.0 编程环境。

2）通过编程熟悉头文件和命名空间的使用方法。

3）不使用类，通过两种算法求解以说明算法之间的运行效率相差甚大。

4）使用类的方式编写最少循环次数的程序。

1.4.2 解答

1. 解题思路

设母鸡、公鸡、小鸡分别为 i、j、k 只，则可以列出如下两个方程：

$$\begin{cases} i+j+k=100 \\ 3i+2j+0.5k=100 \end{cases}$$

这里有 3 个未知数，所以是一个不定方程。要求同时买到母鸡、公鸡、小鸡，就是给出一个限制条件：任何一个都不能为 0。

这需要使用三重循环，通过枚举找出所有符合条件的解答。

2. 第 1 种解题方法

由于 1 元钱能买两只小鸡，所以买的小鸡只数需要从 2 开始，每次增加 2。因为已经考

虑让 i 和 j 从 1 开始枚举，所以不需要判别“$i*j*k!=0$”的附加条件。

```
//参考程序
#include <iostream>
using namespace std;
void main( )
{
     int m=0,n=0,sum=0;
     int i,j,k;
     for(i=1;i<100;i++)
     {
        ++sum;
       for(j=1; j<100;j++)
       {
           ++sum;
          for(k=2;k<100;k=k+2)
         {
                  ++sum;
                  m=i+j+k;
                  n=3*i+2*j+k/2;
                  if((m==100)&&(n==100))
                      cout<<"母鸡: "<<i<<" 公鸡: "<<j<<" 小鸡: "<<k<<endl;
         }
       }
     }
     cout<<"一共循环 "<<sum<<" 次。"<<endl;
}
```

程序运行结果如下：

```
母鸡: 2 公鸡: 30 小鸡: 68
母鸡: 5 公鸡: 25 小鸡: 70
母鸡: 8 公鸡: 20 小鸡: 72
母鸡: 11 公鸡: 15 小鸡: 74
母鸡: 14 公鸡: 10 小鸡: 76
母鸡: 17 公鸡: 5 小鸡: 78
一共循环 490149 次。
```

其实，第 3 层就循环了 480249 次。

3. 第 2 种解题方法

考虑到母鸡为 3 元一只，假设 100 元都买母鸡，最多只能买 33 只。因为要求每个品种都要有，而小鸡只能为偶数，因此母鸡最多能买 30 只，即第一循环变量 i 可从 1 到 30。

因为公鸡为 2 元一只，最多只能买 50 只。又因为至少需要买 1 只母鸡和 2 只小鸡，所以公鸡不会超出 50–3=47 只。因为在循环时已经确定枚举的母鸡数 i，一只母鸡相当于 1.5 只公鸡，所以第二层循环时，公鸡 j 只要从 1 到 47–1.5i 即可。

因为 $i+j+k$=100，所以直接求得 k=100–i–j，不再需要第 3 层循环。也就是说，

```
    k=100-i-j;
    if(3*i+2*j+0.5*k==100)
        cout<<" 母鸡: "<<i<<" 公鸡: "<<j<<" 小鸡: "<<k<<endl;
//改进的算法
#include <iostream>
using namespace std;
void main( )
{
      int k=0,sum=0;
      int i,j;
      for(i=1;i<=30;i++)
```

```
        {
              ++sum;
              for( j=1; j<=47-1.5*i;j++)
              {
                    ++sum;
                    k=100-i-j;
                    if(3*i+2*j+0.5*k==100)
                         cout<<" 母鸡: "<<i<<" 公鸡: "<<j<<" 小鸡: "<<k<<endl;
              }
        }
        cout<<" 一共循环 "<<sum<<" 次。"<<endl;
}
```

程序运行结果如下：

```
母鸡: 2 公鸡: 30 小鸡: 68
母鸡: 5 公鸡: 25 小鸡: 70
母鸡: 8 公鸡: 20 小鸡: 72
母鸡: 11 公鸡: 15 小鸡: 74
母鸡: 14 公鸡: 10 小鸡: 76
母鸡: 17 公鸡: 5 小鸡: 78
一共循环 735 次。
```

其中第二层循环 705 次。

4. 使用类的方式编写程序

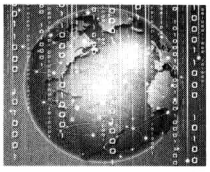
设计一个鸡类，用这个类产生一个对象，对象调用成员函数求解。完整的程序如下：

```
class chicken{
        int i,j,k;
    public:
        chicken(int i=0, int j=0, int k=0){};
        void Find();              // 用来求解的成员函数
};

void chicken::Find()
{
     int sum=0;
     for(i=1;i<=30;i++)
     {
           ++sum;
           for( j=1; j<=47-1.5*i;j++)
           {
               ++sum;
               k=100-i-j;
               if(3*i+2*j+0.5*k==100)
                       cout<<" 母鸡: "<<i<<" 公鸡: "<<j<<" 小鸡: "<<k<<endl;
           }
    }
    cout<<" 一共循环 "<<sum<<" 次。"<<endl;
}
void main()
{
    chicken ck;
    ck.Find();
}
```

1.5 评分标准

本课程设计作为选做项目，主要是为了熟悉环境，只要调试通过即可获得 80 分。如果改写类并调试通过，可以根据情况给予 80 ~ 85 分的成绩。



第 2 章
类和类模板基础

本章重点是熟悉模板以及动态分配内存的使用方法，并掌握多文件编程和基本调试方法，为后面的课程设计打下基础。

2.1 重点和难点

本章重点是设计类模板以及申请并使用动态分配内存的方法，难点是友元函数。此外，将增加部分函数模板和类模板的知识。

2.1.1 模板函数专门化和模板重载

定义函数模板时，还可以使用关键字 typename 代替 class，下面给出一个例子。

【例 2.1】使用显式规则和关键字 typename 编制函数模板。

```
#include <iostream>
using namespace std;
template <typename T>          //使用 typename 替代 class
T max(T m1, T m2)              //求最大值
{ return(m1>m2)?m1:m2;}
template <typename T>          //必须重写
T min(T m1, T m2)              //求最小值
{ return(m1<m2)?m1:m2;}
void main( )
{
    cout<<max("ABC","ABD")<<","<<min("ABC","ABD")<<","
        <<min('W','T')<<","<<min(2.0,5.)<<endl;
    cout<<min<double>(8.5,6)<<"," <<min(8.5,(double)6)<<","<<max((int)8.5,6)<<endl;
    cout<<min<int>(2.3,5.8)<<","<<max<int>('a','y')<<","<<max<char>(95,121)<<endl;
}
```

程序输出结果如下：

```
ABD,ABC,T,2
6,6,8
2,121,y
```

对于那些不标准的书写方式，就不能从函数的参数推断出模板参数。在例 2.1 中，如果使用语句 min(8.5,6)，就需要用 min(double,int) 的形式，与现在定义的模板参数不符，无法通过编译。定义的模板参数中的两个参数的类型必须一致，面对一个整数和一个实数，编译系统无法建立正确的模板函数，即无法实例化。这时也可以对参数表中的参数进行强制转换，语句 min(8.5,(double)6) 就是通过“(double)6”使它们的参数一致。更一般的是使用显式方式 min<double>(8.5,6) 解决这一问题，以保证能正确推断出模板参数。而显式调用方式 max<char>(95,121) 则是输出字符。由此可见，显式规则可以用于特殊场合。

1. 模板函数专门化

虽然按照默认约定，定义一个模板，用户可以使用能想到的任何模板参数（或者模板参

数组合），但有些用户却宁肯选择另外的实现方法。例如，例 2.1 定义的模板函数 max 虽然可以处理字符串，但用户希望换一种处理方法。用户的方案是：如果模板参数不是指针，就使用这个模板；如果是指针，就使用如下的处理方法。

```
char *max(char *a, char *b){    return (strcmp(a,b)>=0?a:b);}
```

由于普通函数优先于模板函数，在执行如下语句

```
char *c1="ABC", *c2="ABD";
cout<<max("ABC","ABD")<<","<<max(c1,c2)<<endl;
```

时，第一个模板函数使用字符串参数，它用普通参数调用函数模板；第二个模板函数使用字符指针参数，则使用指针参数调用函数模板。这两个函数模板使用不同的定义，这样就可以形成完整的模板系，便于管理，并保证在无调用时不会生成任何无用代码。这可以通过提供多个不同定义方式的模板函数来处理，并由编译器根据使用时所提供的模板参数，在这些定义中做出选择。这些对模板可以互相替换的定义称为用户定义的专门化，或简称为用户专门化（有的教材称为定制）。对于模板函数而言，则称为模板函数专门化。

前缀“template <>”说明这是一个专门化，在描述时不用模板参数。可以写成

```
template <>char *max<char*>(char *a, char *b){return (strcmp(a,b)>=0?a:b);}
```

在函数名之后的“char*”说明这个专门化应该在模板参数是 char* 的情况下使用。由于模板参数可以从函数的实际参数列表中推断，所以不需要显式地描述它，即可以简化为

```
template <>char *max<>(char *a, char *b){return (strcmp(a,b)>=0?a:b);}
```

这里给出了 template <> 前缀，第二个 <> 也属多余之举，可以简单地写成如下形式：

```
template <>char *max(char *a, char *b){return (strcmp(a,b)>=0?a:b);}
```

具体的使用方法见例 2.2。

2. 模板重载

C++ 模板的机制也是重载。模板提供了看起来很像多态性的语法，当提供细节时，模板就可以生成模板函数。因为选择调用哪一个函数是在编译时实现的，所以是静态联编。下面通过重载进一步扩大已定义函数模板 max 的适用范围。

【例 2.2】专门化和重载。

```
#include <iostream>
using namespace std;
template <typename T>          // 声明第 1 个函数模板
T max(T m1, T m2)              // 求两个数的最大值
{ return(m1>m2)?m1:m2;}
template <typename T>          // 声明函数模板时，需要重写 template
T max(T a, T b, T c)           // 用 3 个参数重载第 1 个函数模板
{ return max(max(a,b),c);}
template <class T>             // 声明函数模板时，需要重写 template
T max(T a[ ], int n)           // 变换参数类型重载函数模板，求数组中的最大值
{
    T maxnum=a[0];
    for(int i=0; i<n;i++)
        if (maxnum<a[i])maxnum=a[i];
    return maxnum;
}
template <>                    // 函数模板专门化
char *max(char *a, char *b)// 使用指针
```

```
{return (strcmp(a,b)>=0?a:b);}
int  max(int m1, double m2)                                    //普通函数
{ int m3=(int)m2; return(m1>m3)?m1:m3;}
void main( )
{
        char *c1="ABC",*c2="ABD";                              //1 定义字符指针
        cout<<max("ABC","ABD")<<" ";                           //2
        cout<<max("ABC","ABD","ABE")<<" ";                     //3
        cout<<max(c1,c2)<<" ";                                 //4
        cout<<max(2.0,5.,8.9)<<" ";                            //5
        cout<<max(2,6.7)<< endl;                               //6
        double d[]={8.2,2.2,3.2,5.2,7.2,-9.2,15.6,4.5,1.1,2.5}; //定义实数数组 d
        int a[]={-5,-4,-3,-2,-1,-11,-9,-8,-7,-6};              //定义整数数组 a
        char c[]="acdbfgweab";                                 //定义字符串数组 c
        cout<<"intMax="<<max(a,10)<<" doubleMax="<<max(d,10)
            <<" charMax="<<max(c,10)<<endl;
}
```

程序输出如下：

```
ABD ABE ABD 8.9 6
intMax=-1 doubleMax=15.6 charMax=w
```

注意语句 2 和语句 4 的区别：它们执行重载的过程一样，但在重载函数调用时，语句 2 使用定义的模板，语句 4 则使用专门化（指针参数）的函数模板。语句 5 也是调用定义的模板，而语句 6 则使用普通的函数，即它不调用模板。

2.1.2 类模板

类模板的专门化与函数模板同理，将留待第 3 章结合栈举例。

使用类模板可以简化设计，下面给出两个类模板的例子，从中可以看出使用类模板的好处。

【例 2.3】求 4 个数中最大值的类模板程序。

```
#include <iostream>
using namespace std;
template<class T>
class Max4{
    T a,b,c,d;
    T Max(T a, T b){return (a>b)?a:b;}
  public:
    Max4(T, T, T , T );
    T Max(void);
};

template<class T>                              //定义成员函数必须再次声明模板
Max4<T>::Max4(T x1, T x2, T x3, T x4):a(x1),b(x2),c(x3),d(x4){}

template<class T>                              //定义成员函数必须再次声明模板
T Max4<T>::Max(void)                           //定义时要将 Max4<T> 看做整体
{ return Max(Max(a,b),Max(c,d));}

void main()
{
    Max4<char>C('W','w','a','A');              //比较字符
    Max4<int>A(-25,-67,-66,-256);              //比较整数
    Max4<double>B(1.25,4.3,-8.6,3.5);          //比较双精度实数
    cout<<C.Max()<<" "<<A.Max()<<" "<<B.Max()<<endl;
}
```