王道考研系列

计算机考研
——机试指南

- 浙江大学CS考研复试机试“满分帝”负责编写
- 精选一流名校复试上机真题
- 九度OJ(ac.jobdu.com)在线测评
- 名校CS考研复试机试必备

◆ 王道论坛 组编

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

王道考研系列

计算机考研——机试指南

王道论坛　组编

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

目前已有越来越多的高校采用上机考试的形式来考查学生的动手编程能力，对于以应试为主的大学教学模式，上机往往是学生的薄弱环节。本书由浅入深、从简到难讲解了机试的相关考点，并精选名校的复试上机真题作为例题和习题，以给大家提供最可靠的练习指导。书中的所有机试试题在九度 OJ（ac.jobdu.com）均有收录，建议同学们在阅读本书时，结合上机练习，自己动手才是王道！

本书不仅可以作为研究生入学考试的复试复习用书，也可以作为计算机及相关专业的学生练习上机能力的指导用书。

图书在版编目（CIP）数据

计算机考研：机试指南 / 王道论坛组编. —北京：电子工业出版社，2014.1
（王道考研系列）
ISBN 978-7-121-22177-4

I. ①计… II. ①王… III. ①电子计算机－研究生－入学考试－自学参考资料 IV. ①TP3

中国版本图书馆 CIP 数据核字（2013）第 302635 号

策划编辑：谭海平
责任编辑：郝黎明
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1 092　1/16　印张：12.75　字数：326.4 千字
印　　次：2014 年 1 月第 1 次印刷
定　　价：36.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

无论结果如何，总算坚持到了最后。

初试考完了，是不是应该好好放松放松？是不是初试考得好，录取就肯定没有问题了？对不起，这个不是计算机专业研究生考试的规则。目前已有越来越多的高校采用上机考试的形式来考查学生的实际动手编程能力，并且机试在复试中所占的比例非常高，甚至很多高校规定复试成绩不及格者，一律不得录取。目前国内高校开展 ACM 教学的高校非常少，因此**提早开始准备和练习，对于一个完全没有接触过 ACM 的计算机考研人来说，是必须的！**

为方便各位道友练习机试，我们编写了本书，搭建了九度 OJ（ac.jobdu.com），本书中的题均可以在九度 OJ 上提交，并给出了相应的提交网址，提交时可以直接使用王道论坛的账号进行登录。如果在使用过程中遇到问题，欢迎到复试机试讨论专区发贴提出。九度 OJ 目前已收录了我们能够收集到的各高校上机复试真题，也欢迎大家向我们提供各高校上机真题，具体请通过邮件联系浩帆（E-mail：qihu@zju.edu.cn）。

考研其实没有什么诀窍，就是每天比别人早起一点，晚睡一点，比别人早准备一点，勤奋一点。我坚信一个写不出合格代码的计算机专业的学生，即便考上了研究生，无非也只是给未来失业判个缓期执行而已。

王道是道友们考研路上值得信任的好伙伴，五年多来他陪伴了近四十万的计算机考研人，不离不弃。王道尊重的不是考研这个行当，而是王道上这群执着的道友的梦想，看着你们圆梦，我们内心充满了成就感。

道友们，要忠实于自己心底的梦想，勇敢地坚持下去，而当下，请开始准备复试吧，熬过这两个月，一切就都好了。

目　录

第 1 章 从零开始

一、机试的意义

众所周知，机试是计算机考研复试中非常重要的一个环节。在越来越注重实践动手能力的今天，有越来越多的知名高校在计算机研究生入学考试当中采用了机试的考查形式，通过这种形式来考查学生的分析问题并利用计算机程序解决问题的能力。通过机试，可以考查一个考生从实际问题当中抽象得出数学模型的能力，利用所学的计算机专业知识对该模型进行分析求解的能力，以及利用计算机编程语言，结合数据结构和算法真正解决该实际问题的能力。

所以，大家在准备机试的过程中要特别注意以下几个方面：

1．如何将一个实际问题抽象成数学问题。例如，将高速公路网抽象成带权图，这就是一种简单的、直接的抽象。

2．如何将我们所学的计算机专业知识，运用到解决抽象出来的数学模型上去。这就要求我们在脑子里事先熟知一些常用的数据结构和算法，再结合模型求解的要求，很快地选择合适的编程思想来完成算法的设计。甚至可以利用一些经典算法特征，加入一些自己的优化，使得编写的程序更优雅、更高效（当然这是建立在充分理解经典算法的基础上）。

3．如何将我们为解决该数学模型所设计的算法编写成一个能被计算机真正执行的计算机程序。笔者认为，关于这个能力的定义有三个层次：①会编写（默写）一些经典算法的程序代码。②能够将自己的想法或设计的算法转换为程序代码。③能够使得自己编写的程序在大量的、多种多样的、极限的测试数据面前依旧正常完成功能（程序的健壮性）。我们在准备机试的训练过程中，就要依次经历这三个层次，从而最后能够在实际考试当中取得理想的成绩。

本教程从分析经典机试真题出发，它们均为近几年频繁被考查的数据结构和算法，利用 C/C++语言进行讲解，并加以一些相关知识的扩展，希望在读者准备计算机考研机试的过程中充当指引者的角色。同时，由于笔者自身能力的限制以及编写时间的不足，教程中难免存在一些疏漏和错误，恳请读者指正。

二、机试的形式

绝大部分机试所采用的形式，归结起来可以概括为：得到题目后，在计算机上完成作答，由计算机评判并实时告知结果的考试过程。

机试考试中的问题往往由五部分组成。首先是问题描述，描述该问题的题面，题面或直接告知考生所要解决的数学问题、或给出一个生活中的实际案例，以待考生自己从中抽象出所要解决的数学模型。第二是输入格式，约定计算机将要给出的输入数据是以怎样的顺序和格式向程序输入的，更重要的是它将给出输入数据中各个数据的数据范围，我们通过这些给出的数据范围确定数据的规模，为我们设计算法提供重要依据。第三是输出格式，明确考生将要编写的程序将以怎样的顺序和格式输出题面所要求的答案。第四第五部分即输入、输出数据举例（Sample）。好的 Sample 不仅能为考生提供一组简单的测试用例，同时也能明确题意，为题面描述不清或有歧义的地方做适当的补充。

另外也要特别注意，题目中给定的两个重要参数：①时间限制。②空间限制。这两个重要的参数限定了考生提交的程序在输出答案之前所能耗费的时间和空间。

我们来看一个典型的题目描述，从而了解机试题的问题形式。

例 1.1 计算 A+B （九度 OJ 题号：1000）

时间限制：1 秒　　内存限制：32 兆　　特殊判题：否

题目描述：

求整数 a,b 的和。

输入：

测试案例有多行，每行为 a,b 的值，a,b 为 int 范围。

输出：

输出多行，对应 a+b 的结果。

样例输入：

```
1 2
4 5
6 9
```

样例输出：

```
3
9
15
```

通过该例，我们基本了解了机试题的问题形式，以及问题各部分所起到的作用。这里补充解释一下所谓特殊判题（Special Judge）的含义，特殊判题常被应用在可能存在多个符合条件的答案的情况下，若评判系统采用了特殊判题，那么系统只要求输出任何一组解即可；若系统对该题并没有采用特殊判题，那么你必须严格按照题目中对输出的限定输出对应的答案（如输出字典序最小的解）。

得到题目后，考生在计算机上立即编写程序，确认无误后，将该程序源代码提交给评判系统。评判系统将考生提交的源代码编译后，将后台预先存储的输入测试数据输入考生程序，并将该程序输出的数据与预先存储在评判系统上的“答案”进行比对得出结果。

评判系统评判考生程序后，实时地将评判结果返回到考生界面，考生可以根据该结果了解自己的程序是否被评判系统判为正确，从而根据不同的结果继续完成考试。

三、评判结果

本节将对评判系统评判考生提交程序后返回的结果做详细的说明，并且针对不同的返回结果，对可能出现错误的地方做出初步的界定。

Accepted（答案正确）：你的程序对所有的测试数据都输出了正确的答案，你已经得到了该题的所有分数，恭喜。

Wrong Answer（答案错误）：评判系统测试到你的程序对若干组（或者全部）测试数据没有输出正确的结果。

出现该种错误后，一般有两种解决方向：如果对设计的算法正确性有较大的把握，那么你可以重点考虑代码健壮性，即是否存在某些特殊数据使程序出现错误，比如边界数据，比如程序中变量出现溢出。另一种方向，即怀疑算法本身的正确性，那么你就需要重新考虑你的算法设计了。

Presentation Error（格式错误）：评判系统认为你的程序输出“好像”是正确的，只是没有严格按照题目当中输出所要求的输出格式来输出你的答案，例如你忽略了题目要求在每组输出后再输出一个空行。

出现这种错误，往往预示着你离完全正确已经不远了，出现错误似乎只是因为多输出了一些空格、换行之类的多余字符而已。但这不是绝对的，假如在排版题（后文会有介绍）中出现格式错误，那么有可能你离正确的答案仍然有一定的距离。

Time Limit Exceeded（超出时间限制）：你的程序在输出所有需要输出的答案之前已经超过了题目中所规定的时间。

若这种结果出现在你的评判结果里，依然有两种方向可供参考：①假如你确定算法时间复杂度能够符合题目的要求，那么依旧可以检查是否程序可能在某种情况下出现死循环，是否有边界数据可能会让你的代码不按照预想的工作，从而使程序不能正常的结束。②你设计的算法时间复杂度是否已经高于题目对复杂度的要求，如果是这样，那么你需要重新设计更加高效的算法或者对你现行的算法进行一定的优化。

Runtime Error（运行时错误）：你的程序在计算答案的过程中由于出现了某种致命的原因异常终止。

你可以考虑以下几个要点来排除该错误：①程序是否访问了不该访问的内存地址，比如访问数组下标越界。②程序是否出现了除以整数 0，从而使程序异常。③程序是否调用了评判系统禁止调用的函数。④程序是否会出现因为递归过深或其他原因造成的栈溢出。

Compile Error （编译错误）：你提交的程序并没有通过评判系统的编译，可根据更详细的编译信息修改你的程序。

Memory Limit Exceeded （使用内存超出限制）：你提交的程序在运行输出所有的答案之前所调用的内存已经超过了题目中所限定的内存限制。

造成这种错误的原因主要有两个方面：①你的程序申请过多的内存来完成所要求的工作，即算法空间复杂度过高。②因为程序本身的某种错误使得程序不断地申请内存，例如因为某种原因出现了死循环，使得队列中不断地被放入元素。当然也千万别忽略自己的低级错误，比如在声明数组大小时多打了一个 0。

Output Limit Exceeded（输出超出限制）：你的程序输出了过多的东西，甚至超出了评判系统为了自我保护而设定的被评判程序输出大小的最高上限。

一般来说该种错误并不常见，一旦出现了也很好找原因。要么就是你在提交时忘记关闭你在调试时输出的调试信息（我经常输出 DP 时的数组来动态地观察状态的转移）；要么就是程序的输出部分出现了死循环，使得程序不断地输出而超出系统的限制。

以上几种结果就是评判系统可能会返回的几个最基本的结果。若返回 Accepted，则你可以获得该题的所有分数。若返回其他错误，则根据不同的考试规则，你的得分将会有一定的差异。若你参加的考试采用按测试点给分规则，你依然能够获得你通过的测试点（即该程序返回正确结果的那部分测试数据）所对应的分数；但是，若你参加考试采用所有数据通过才能得分的评分规则，那么很可惜，到目前为止你在这道题上的得分依旧是 0 分。

假如评判结果显示你提交的程序是错误的，你可以在修改程序后再次提交该题，直到获得满意的分数或者放弃作答该题。

四、复杂度的估计

本节将详细讨论题目中所给定的时间限定和空间限定对我们程序设计的指导作用。

如例 1.1 所示，该题给予程序 1 秒的运行时限，这也是最常见的时间限制（或最常见的时间限制数量级）。对于该时限，通常，我们所设计的算法复杂度不能超过百万级别，即不能超过一千万。即若算法的时间复杂度是 $O(n^2)$，则该 n（往往在题目中会给出数据范围）不应大于 3000，否则将会达到我们所说的千万数量级复杂度，从而程序运行时间超出题目中给出的用时限定。举例来说，我们不能在 1 秒时限的题目当中对 10000 个整数进行冒泡排序，而必须使用快速排序等时间复杂度为 $O(n\log n)$的排序算法，否则程序很可能将会得到运行时间超出限制的评判结果。因此你可以对你的程序在最坏情况下的复杂度进行一个估算，假如确定其在百万数量级之内，那么你的程序一般是不会超出时间限制的。对于其他时间限制的情况，可以参考 1 秒时限对时间复杂度的要求，做出一定的估计，从而保证自己的程序运行所需的时间不会超过题目中对运行时间的限制。

我们同样可以知道，例 1.1 中限定的内存空间为 32MB，即你的程序在评测系统中运行时，不得使用超过 32MB 大小的内存。空间限定则比较好处理，你可以简单地计算你

所申请的内存空间的大小（例如，我们可以轻易地计算 int mat[300][300]所占用的内存大小）。只要该大小没有超过或过分接近空间限定（运行时需要一些额外的空间消耗），那么你的程序应当是符合空间限制条件的。现如今的机试题，一般不会对空间做过多的限制，大多数情况只对时间做出明确的要求，所以考题在空间上将会尽量满足考生程序的需要。正因为如此，我们在很多情况下都应该有“空间换时间”的思想。

五、OJ 的使用

我们该去何处开始机试训练和模拟考试呢？可能大部分人已经听说过在线评判系统（Online Judge），例如 HDOJ（acm.hdu.edu.cn）、POJ(poj.org)。但是这些 OJ 都是为 ACM/ICPC 训练而设计的，虽然形式与机试大同小异但是难度却有较大差别，普通的考生在其中训练不仅打击了信心，也在一定程度上浪费了时间。因此，推荐专门为大家计算机考研机试所设计的九度 OJ（ac.jobdu.com），它收录了近三百道各大高校近几年来的机试真题，真正为我们准备机试提供了极大的便利。

我们在它的首页上可以看到九度 OJ 所包含的各种功能。

首先，需要单击“注册”按钮，为自己注册一个账号，有了该账号你就能在九度 OJ 上进行日常练习，同样也有资格参加九度 OJ 举行的各类比赛和模拟考试。

现假设我们已经有了一个账号，并且想要开始在线练习。那么，可以在页面左边的导航栏找到在线练习，可以选择题目汇总来查看九度 OJ 收录的所有在线练习题，或者

单击考研机试题集来查看九度为各位考生收录的近几年各大高校机试真题。例如我们选择题目汇总中的题号为 1000 的问题，例 1.1 中的例子便出现在页面当中。当用户完成作答以后，可以在页面左侧单击“提交答案”按钮，将源代码提交给九度在线评测系统，提交完成后，系统会自动跳转到评测状态页面，通过查看你刚才提交的源代码的评测状态，你就可以知道你是否通过了该题，从而达到锻炼自我的目的。在本文中，凡是涉及九度 OJ 上收录的例题，本书会明确该题的题号。

同样，九度 OJ 也经常为广大考生提供在线模拟考试训练。我们可以单击“比赛及考试”按钮，来查看过往的比赛以及近期的比赛安排，从而有选择地参加在线比赛，考察自己的训练质量，为日后进一步训练做出导向。

总结

本节主要介绍机试的意义、形式，以及机试题的形式和考察的方法，并在最后给出了我们可以完成机试训练的网站——九度 Online Judge。

第 2 章

经典入门

本章通过对频繁出现在考研机试真题中的一些经典问题进行讲解，并对解题方法做详细的说明，旨在使读者快速了解机试的考查形式和答题方法，以便进一步学习。

一、排序

排序考点在历年机试考点中分布广泛。排序既是我们必须要掌握的基本算法，同时也是学习其他大部分算法的前提和保证。

首先我们来学习对基本类型的排序。所谓对基本类型排序，即我们对诸如整数、浮点数等计算机编程语言内置的基本类型进行排序的过程。

例 2.1 排序 （九度教程第 1 题）

时间限制：1 秒　　内存限制：32 兆　　特殊判题：否

题目描述：

对输入的 n 个数进行排序并输出。

输入：

输入的第一行包括一个整数 n(1<=n<=100)。接下来的一行包括 n 个整数。

输出：

可能有多组测试数据，对于每组数据，将排序后的 n 个整数输出，每个数后面都有一个空格。每组测试数据的结果占一行。

样例输入：

```
4
1 4 3 2
```

样例输出：

```
1 2 3 4
```

来源：

2006 年华中科技大学计算机保研机试真题

提交网址：http://ac.jobdu.com/problem.php?pid=1202

这便是一例曾经出现在机试中关于排序考点的真题。面对这样的考题，读者可能很

快就会联想到《数据结构》教科书上各种形形色色、特点不同的排序算法。例如，我们可以自然想到选择冒泡排序来完成此题。但是在确定使用冒泡排序之前，我们不应忽略对其复杂度的考量，而应特别注意在选择将要采用的算法时估计其复杂度。以此处为例，假如我们采用冒泡排序来完成此题，应该注意到冒泡排序的时间复杂度为 $O(n^2)$。而 n 的取值范围也在题面中明确地给出($1 \leqslant n \leqslant 100$)，这样我们可以估算出 n^2 的数量级仅在万级别，其时间复杂度并没有超过在前一章中所讲的百万数量级复杂度，所以使用冒泡排序在该例限定的一秒运行时间里是完全可以接受的；同时冒泡排序的空间复杂度为 $O(n)$，即大致需要 100 * 32bit（数组长度 * int 所占内存）的内存，这样，所需的内存也不会超过该例限定的内存大小（32MB）。只有经过这样的复杂度分析，我们才能真正确定冒泡排序符合我们的要求。于是，我们可以开始着手编写该例的解题代码。

代码 2.1

```
#include <stdio.h>
int main(){
    int n;
    int buf[100]; //定义我们将要使用的变量n,并用buf[100]来保存将要排序的数字
    while(scanf("%d",&n)!=EOF){ //输入n,并实现多组数据的输入
        for (int i=0;i<n;i++){
            scanf("%d",&buf[i]);
        } //输入待排序数字
        for(int i=0;i<n;i++){
            for(int j=0;j<n-1-i;j++){
                if(buf[j]>buf[j+1]){
                    int tmp=buf[j];
                    buf[j]=buf[j+1];
                    buf[j+1]=tmp;
                }
            }
        } //冒泡排序主体
        for (int i=0;i<n;i++){
            printf("%d ",buf[i]);
        } //输出完成排序后的数字,注意,题面输出要求在每个数字后都添加一个空格
        printf("\n"); //输出换行
    }
    return 0;
}
```

初见这段代码，读者可能对主循环体的循环条件大为不解。`while (scanf ("%d", &n) != EOF)`，它在完成输入 n 的同时又完成了什么看起来令人迷惑的条件判断？要回答这个问题，我们首先要明确两点：第一，scanf 函数是有返回值的（尽管大多时候都会被人忽略），它将返回被输入函数成功赋值的变量个数。在此例中，若成功完成输入并对 n 赋值，scanf 函数的返回值即为 1。我们即是通过该返回值来完成循环条件的判断的。

第二，也要注意到，该例题面中明确了输入数据会有多组，我们将要对每组输入都输出相应答案。并且，事先我们并不知道将会有多少组数据被输入到程序中。于是，我们可以使用该循环测试条件来完成对输入是否结束的判断。过程如下：如仍有测试数据未被测试完，那么该组测试的开头一行将如题面所说，为一个整数（n），其将会被 scanf 语句赋值给变量 n，那么 scanf 返回值为 1，不等于 EOF(−1)，循环条件成立，程序进入循环体，执行程序；如果输入已经到达结尾（输入文件到达末尾或在命令台中输入 Ctrl+Z），scanf 函数无法再为变量 n 赋值，于是 scanf 函数返回 EOF（end of file）。至此，循环条件不成立，程序跳过循环体，执行 return 0，使程序正常的结束。该循环判断条件既保证了可以对多组测试数据进行计算，同时又使程序在输入结束后能够正常退出。反之，如果我们不使用循环，程序在处理完一组数据后就会退出，后续测试数据无法被正常处理。但若使用死循环而不加以任何退出条件，那么程序在处理完所有的测试数据后仍不能正常退出，虽然程序已经输出了所有测试数据的答案，但评判系统还是会认为程序在限制时间内无法运行完毕，于是返回超时的评判结果。正因为如此，初学者很容易因为这个原因，出现莫名其妙的程序超时。所以，我们必须使用该循环判断条件来完成以上两个功能。

值得一提的是，若输入为字符串而程序采用 gets()的方法读入，则相同功能的循环判断语句为 while(gets(字符串变量))。

解决了关于循环测试条件的疑惑，对于代码其他部分，相关注释已在代码中给出，读者应该非常容易理解它的工作流程。

另外需要指出的是，假如你使用 VC6.0 编译这段程序，很可能会出现编译错误。这是因为 VC6.0 对 for 循环测试条件中定义的指示变量 i 作用域的不同规定造成的。C++ 标准指出，在 for 循环循环条件中定义的变量，其作用域仅限于 for 循环循环体内部（就像在 for 循环循环体内定义的局部变量）。于是我们在多个 for 循环中都重新定义指示变量 i 来完成相应的功能。而 VC6.0 中则不同，在 for 循环退出以后，指示变量 i 依然可见（就像在 for 循环循环体外定义的局部变量）。所以我们在后续 for 循环循环条件中定义的新指示变量 i 会与该旧的指示变量 i 产生冲突，而无法被成功定义，于是编译器于此给出了编译错误。因此，我个人并不推荐使用 VC6.0 编译器。假如你在练习和考试中，不得不使用该编译器，那么请在 for 循环循环条件第一次定义指示变量 i 后，不必重新定义，而是直接使用我们已经定义的并且依旧可见的变量来完成新的 for 循环功能（莫忘初始化）。

在明确以上注意点后，我们重新回到排序问题本身上来。该例程使用冒泡排序来完成题目要求。但假如修改题目中 n 的取值范围，使其最大能达到 10000，我们又应该如何解决该问题呢。读者可能很快就能反应过来，冒泡排序因其较高的时间复杂度（O（10000 * 10000）已经超过了百万数量级）而不能再被采用，于是我们不得不使用诸如快速排序、归并排序等具有更优复杂度的排序算法。那么，新问题出现了，恐怕大部分读者都对此感到恐惧：是否能正确的写出快速排序、归并排序等的程序代码？其实，我

们并不用为此感到担心，C++已经为我们编写了快速排序库函数，只需调用该函数，便能轻易地完成快速排序。

代码 2.2

```
#include <stdio.h>
#include <algorithm>
using namespace std;
int main(){
    int n;
    int buf[10000];
    while(scanf("%d",&n)!=EOF){
        for(int i=0;i<n;i++){
            scanf("%d",&buf[i]);
        }
        sort(buf,buf+n); //使用该重载形式,表明将要使用自己定义的排列规则
        for(int i=0;i<n;i++){
            printf("%d ",buf[i]);
        }
        printf("\n");
    }
    return 0;
}
```

因为我们将要在程序中使用了 sort 库函数，在头文件方面包含了 algorithm 头文件，并使用 using namespace std 语句声明了我们将会使用标准命名空间（sort 被定义在其中）。此例中，sort 函数的两个参数代表待排序内存的起始地址和结束地址（sort 函数还有另外的重载形式，在后文中会有所提及），在此例中起始地址为 buf，结束地址为 buf + n。该函数调用完成后，buf 数组中的数字就已经通过快速排序升序排列。我们只需对其输出即可。

可能习惯于用 C 来编写程序或对 C++不很熟悉的读者会对某些 C++特性感到有所困难。那么，关于 sort，我给出两种建议：要么记住 sort 并不复杂的基本用法；要么使用 C 中快速排序的函数 qsort（可自行学习）。

那么假如我们将要对给定的数组进行降序排列该如何操作呢？有一个简单的方法，只需简单将升序排列后的数组倒序，便能得到降序排列的数组。但是这里，还要介绍另一种使用 sort 函数的方法来完成降序排列。

代码 2.3

```
#include <stdio.h>
#include <algorithm>
using namespace std;
bool cmp(int x,int y){ //定义排序规则
    return x>y;
}
int main(){
    int n;
    int buf[100];
```

```
    while(scanf("%d",&n)!=EOF){
        for(int i=0;i<n;i++){
            scanf("%d",&buf[i]);
        }
        sort(buf,buf+n,cmp); //使用该重载形式,我们表明将要使用自己定义的排
                               列规则
        for(int i=0;i<n;i++){
            printf("%d ",buf[i]);
        }
        printf("\n");
    }
    return 0;
}
```

如该代码所示，我们新定义了一个 cmp 函数，来实现对于新的排序规则的定义。关于 cmp 函数的定义规则我们只需简单的记得，当 cmp 的返回值为 true 时，即表示 cmp 函数的第一个参数将会排在第二个参数之前（即在我们定义的规则中，cmp 表示第一个参数是否比第二个参数大，若是，则排在前面）。为了实现降序排列，我们只要判断两个参数的大小，当第一个参数比第二个参数大时返回 true。然后，只需要调用 sort 函数的另一种重载方式：sort（排序起始地址，排序结束地址，比较函数），如代码 2.3 所示，我们便完成了对我们给定的内存区间的降序排列。

对于利用 sort 函数为其他基本类型（如 double，char 等）排序的方法也大同小异，读者可以自己动手进行尝试。

使用内置函数 sort 的方法，简单易懂，且不易出错，对各种类型的排序写法也大同小异。所以，强烈推荐大家使用 sort 来完成排序。

例 2.2　成绩排序　（九度教程第 2 题）

时间限制：1 秒　　内存限制：32 兆　　特殊判题：否

题目描述：

有 N 个学生的数据，将学生数据按成绩高低排序，如果成绩相同则按姓名字符的字母序排序，如果姓名的字母序也相同则按照学生的年龄排序，并输出 N 个学生排序后的信息。

输入：

测试数据有多组，每组输入第一行有一个整数 N（N<=1000），接下来的 N 行包括 N 个学生的数据。每个学生的数据包括姓名（长度不超过 100 的字符串）、年龄（整型数）、成绩（小于等于 100 的正数）。

输出：

将学生信息按成绩进行排序，成绩相同的则按姓名的字母序进行排序。然后输出学生信息，按照如下格式：姓名 年龄 成绩

样例输入：

```
3
abc 20 99
bcd 19 97
bed 20 97
```