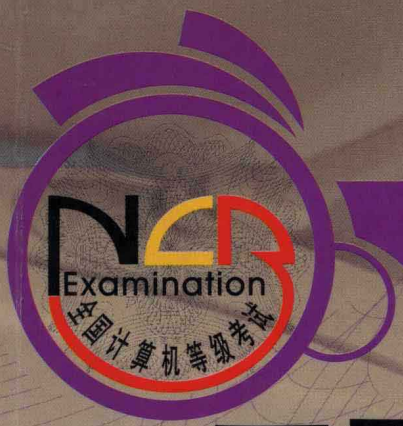

全国计算机等级考试

2007

历届上机真题详解

二级C语言程序设计

全国计算机等级考试命题研究组　编

南开大学出版社

2007

历届上机真题详解

全国计算机等级考试

历届上机真题详解

二级C语言程序设计

全国计算机等级考试命题研究组 编

南开大学出版社
天津

内容提要

本书提供了历届全国计算机等级考试二级C语言机试试题及其答案，并对其进行了详细准确的分析，指出考核的知识点、重点难点、解题思路、程序流程，另外针对许多题目提出了更为有效便捷的解法，使读者可以开阔思路，学到编程思想与技巧。

本书配套光盘中，有两部分内容：二级C语言上机考试的全真模拟环境；本书所有试题的题目源文件。读者可以利用配套光盘进行考前训练。

本书所有题目均进行了测试，保证能够在实际环境中正常运行。

本书针对参加全国计算机等级考试二C语言程序设计的考生，同时也可作为普通高校、大专院校、成人高等教育以及相关培训班的练习题和考试题使用。

图书在版编目(CIP)数据

全国计算机等级考试历届上机真题详解. 二级C语言程序技术 / 全国计算机等级考试命题研究组编. —4版.
天津：南开大学出版社，2006.11
ISBN 7-310-02282-3

Ⅰ.全… Ⅱ.全… Ⅲ.①电子计算机—水平考试—解题②C语言—程序设计—水平考试—解题
Ⅳ.TP3-44

中国版本图书馆CIP数据核字(2006)第110937号

南开大学出版社出版发行
出版人：肖占鹏
地址 天津市南开区卫津路94号 邮政编码 300071
营销部电话 (022)23508339 23500755
营销部传真 (022)23508542 邮购部电话 (022)23502200
*
河北省迁安万隆印刷有限责任公司印刷
全国各地新华书店经销
*
2006年11月第4版 2006年11月第4次印刷
787×1092毫米 16开本 29.75印张 750千字
定价：42.00元

如遇图书印装质量问题，请与本社营销部联系调换，电话 (022)23507125

前　言

全国计算机等级考试（National Computer Rank Examination，NCRE）是由教育部考试中心主办，用于考查应试人员的计算机应用知识与能力的考试。本考试的证书已经成为许多单位招聘员工的一个必要条件，具有相当的“含金量”。

为了帮助考生更顺利地通过计算机等级考试，我们做了大量市场调查，根据考生的备考体会，以及培训教师的授课经验，推出了《历届上机考试真题详解——二级 C 语言程序设计》。本书主要有 4 个组成部分。

一、二级 C 语言上机考试答题策略与技巧

在这部分中，针对 C 语言上机考试的改错题和编程题，分别提出了如何准备和应对的方法与技巧。

二、二级 C 语言上机考试题库

对于备战等级考试而言，做真题，是进行考前冲刺的最佳方式。这是因为它的针对性相当强，考生可以通过实际练习试题，来检验自己是否真正掌握了相关知识点，了解考试重点，并且根据需要再对知识结构的薄弱环节进行强化。

三、真题答案和详解

每道题都配有清晰准确的答案以及详尽的分析，精解考点，分析题眼，详解重点难点，剖析程序流程，给出应试技巧。另外，许多题目都有一个“提高补遗”部分，说明了如何更巧妙地解题，可以开阔读者的编程思路，这可以说是本书的画龙点睛之笔。

四、机试全真模拟环境配套光盘

本书配套光盘中有本考试的全真模拟上机环境，并配有大量全真模拟试题，可用于考前实战训练。另外，光盘中还附带了本书所有题目的源文件。

本书针对参加全国计算机等级考试二级 C 语言程序设计的考生，同时也可以作为普通高校、大专院校、成人高等教育以及相关培训班的练习题和考试题使用。

为了保证本书及时面世和内容准确，很多朋友做出了贡献，陈河南、李齐鹏、贺民、贺军、许伟、侯佳宜、戴文雅、戴军、何雄、周虎、刘大为、夏绪虎、任世华、汤效平、裘蕾、赵晓睿、邵世磊、薛飞、冯苗、王淼、刘占坤、李东锋、张新苗、陈占军、李季、邹晓东、于樊鹏、梁彩隆、赵世伟、郑炎、黄志雄、邱代燕、宫晓琳、田仙仙、王莹、李志云、陈安南、李晓春、王春桥、王雷、韦笑、龚亚萍等老师在编写文档、调试程序、排版、查错、预读、光盘制作与测试等工作中加班加点，付出了很多辛苦，在此一并表示感谢！

本书中的所有题目均进行了测试，保证能够在实际环境中正常运行。在学习的过程中，您如有意见或建议，请与我们联系：xiaoxiang-007@sohu.com。

全国计算机等级考试命题研究组

二级 C 语言上机考试答题策略与技巧

改错题策略与技巧

做好改错题，必须首先明确一条规则：“一定要让机器看得懂”。因为现在的上机考试都是采用机器评分，比较呆板，所以没有让动的地方，坚决不要“轻举妄动”，我们需要特别注意的是，如何寻找错误。

对于改错题，一般有 2~3 处错误不等，均在以“/********found*******/”的标志下方。有的题目明确指出错误紧跟在“***found***”下面第一行，而有的没有明确指出来。考生应重点注意“***found***”下面的第一条语句（可能这条语句分为几行）。

需要注意，有时候几条语句出现错误，实际上是一条错误引起的连锁反应，这时要抓住要害，由前向后修改，当按 Ctrl+F9 编译出现多处错误时，千万不要紧张或错误地以为考题有误，往往修改一个错误后，其他错误就自动消失。同时需要注意，程序修改时，应尽量按照源程序的思路。总的来说，改动的地方很小，切忌打破原程序结构，特别注意不要增删语句（题目中明确指出要增删语句时除外）。在修改程序错误的时候，用好调试工具，就可以轻而易举地发现很多语法错误及部分逻辑错误。

一般情况下，错误主要分为两大类：

（1）语法错误。对于这种错误，用编译器很容易解决。所以，改错题的第一步是先编译，解决这类语法错误。下面总结了二级 C 语言上机改错题中常见的语法错误：

① 丢失分号，或分号误写成逗号。

② 关键字拼写错误，如本来小写变成大写。

③ 语句格式错误，例如 for 语句中多写或者少写分号。

④ 表达式声明错误，例如：少了()。

⑤ 函数类型说明错误。与 main()函数中不一致。

⑥ 函数形参类型声明错误。例如，少*等。

⑦ 运算符书写错误，例如：/写成了\ 。

（2）逻辑错误，或者叫语义错误，这和实现程序功能紧密相关，一般不能用编译器发现。对于逻辑错误可以按这样的步骤进行查找：

（1）先读试题，看清题目的功能要求。

（2）通读程序，看懂程序中算法的实现方法。

（3）细看程序，发现常见错误点。

下面列出了一些常见逻辑错误以供参考：

① 变量初值错误。

② 循环次数不对。

③ 下标越界。

④ 运算类型不匹配。

编程题答题策略与技巧

学习程序设计的目的就是为了编程，因此编程题是上机考试的重点，但也是难点

一、上机题的注意事项

要做好编程题，平时就要多做一些典型的习题，掌握典型的算法。下面给出做编程题的思路及一些注意事项以供参考。

（1）认真阅读试题。需要收集的信息包括：

① 题目所给出的限制条件，如“不得使用 C 语言提供的字符串函数”等。

② 一些术语的解释，如“回文数是指其各位数字左右对称的整数”等。

③ 题目提供的算法，如下面第 10 题给出了求方程实根的迭代算法。

④ 程序的执行结果，程序的结果对变量定义、函数返回值、程序调试等很有用。

（2）清晰地理解 fun 函数。需要收集的信息包括：

① 函数的功能。

② 函数返回值类型。

③ 参数传递方式。

④ 运行结果。

（3）把握题目的考点，如字符串、字符的 ASCII 码、数组、指针、结构体、数学计算、排序算法等。

（4）实现 fun 函数。主要有以下注意事项：

① 根据 fun 函数的功能及考点，思考应采用的算法，此时要联想平时做过的同类型的习题。

② “扬长避短”。如果有几种算法或者有几种实现语句可选，则选用自己最擅长的。

③ 根据 fun 函数的功能、主函数调用情况及程序执行结果，定义合适的临时变量，并注意根据程序的需要即时、合理的给变量赋初值。

④ 保证良好的程序版式。这有利于阅读和改错。如，用有具体含义的变量名及合理地空格、空行、对齐等。

（5）调试运行程序。主要有以下注意事项：

① 先调试程序，如果直接运行，可能出现无限循环或死机现象。调试程序可以发现语法错误。调试时需要有意识地注意循环变量是否赋初值、循环界限、循环变量是否递增或递减，这样可以有效防止无限循环情况的发生。

② 运行程序。首先用题目给的输入数据，观察运行结果是否与题目所给的相同。如果不同，可能是函数返回类型或通过形参返回时出现错误，也可能是程序逻辑或算法不正确。然后用几组特殊的值判断结果是否正确。

③ 利用好 TC 的调试工具。如把 F7、F8 和 Ctrl+F4 组合运用。

二、上机题的分类

分析近年来的上机题，我们发现试题基本上变化不大，每年的试题重复率很高，上机试题分为 5 个大类，这五类分别为：字符串处理、数组处理、结构体及链表、数学问题、实际应用。下面将对上述 5 类问题作概括讲解。

（1）字符串处理

字符串处理一般使用指针或数组，for 或 while 循环语句。需要注意给修改后的字符串赋上结尾标志字符'\0'。可细分为以下几种：

① 字符 ASCII 码值应用

- 排序（21 题） 关键是采用合适的排序算法，一般使用易懂的选择排序法。
- 比较字符串大小
- 大小写转换（29 题、47 题）
- 删除指定 ASCII 码的字符（31 题、32 题、49 题、64 题、78 题） 算法：小写字符=大写字符+32

② 字符查找及删除指定字符（6 题、19 题、33 题、35 题、38 题、45 题、56 题、60 题、66 题、72 题、73 题、83 题、88 题、94 题、95 题） 这类题要用 if 语句，删除字符即把非删除字符拷贝到原串，如 if（str[i]!=c，）str[j]=str[i];。

③ 子字符串查找（43 题、44 题）

④ 字符统计（4 题）

⑤ 字符串逆置（17 题） 算法是：定义一个临时字符变量把字符串首尾对应位置的字符互换。

⑥ 回文数（23 题） 算法是：通过比较字符串首尾字符是否相等来判断是否为回文数。

⑦ 数字字符串转换成长整数（24 题） 算法是：把数字字符转换成数字（数字=数字字符-'0'）；把数字按位合并成一个长整数（即位上的数字乘以位权，并累加求和）。

⑧ 字符串长度的比较（25 题）

⑨ 子字符串移动（36 题、40 题） 算法是：把字符先拷贝到临时字符串中，然后再拷回原串。

⑩ 字符串连接（52 题、59 题）

（2）数组处理

数组与字符串是紧密相连的，如字符串处理也可转换成数组处理。数组处理多用数组的下标进行运算。数组处理又可以细分为以下几种：

① 数组元素排序

② 求数组元素的最大值、最小值和平均值（1 题、7 题、13 题、14 题、28 题、30 题、82 题、95 题） 求最大值或最小值首先要定义一个临时变量（如 temp），一般把第一个数组元素的值赋给 temp 作为比较初值，并在循环中改变 temp 的值，使 temp 是当前最小或最大的值。求平均值要先累加求和，注意必须先给保存和的变量赋初值 0。

③ 移动数组元素（39 题）

④ 把指定数组元素移动到字符串或数组中（41 题、55 题、75 题）

⑤ 元素分段存放（61 题）

（3）数学问题

数学问题较多地应用到数组和常用数学算法，注意对于用于累加或累乘的变量要先赋初值。可细分为以下几种：

① 公式求值（9 题、69 题、86 题） 一般应分析公式的特点，把它拆分成几个独立的单元，分别求值，然后组合。

② 多项式求值（8 题、26 题、68 题、70 题、76 题、90 题、92 题、93 题、97 题、

100 题） 首先分析公式的组成特点，一般后项可以由前项累加或累乘求得，然后再利用循环累加多项式当前所有项的和。

③ 素数问题，一般要求把查找到的素数保存在形参数组中返回，把素数的个数保存在形参指针所指向的内存或用 return 语句返回。

- 求 n 到 m 之间的所有素数（2 题、20 题、51 题、99 题） 首先要注意寻求素数的范围是否包括 n 和 m，查找界限不能错；其次清楚素数的定义（素数：只能被 1 和自身整除的整数）；再次需注意判断素数的算法：查找 i 的除数 j 的范围按定义来说应为[1,i]，但 1 不是素数，2 才是最小的素数。从 2 开始查找除数 j 易于素数的判断，因为一个数若是素数，则不能被[2,i-1]之间的数整除，结果必有 j==i（也可以写成 j>=i）。数学证明：如果 i 不能被[2,sqrt(i)]之间的数整除，则在（sqrt（i），i-1]也不存在能整除 i 的数，所以寻找素数除数的范围可以缩小为[2,sqrt(i)]，因为 sqrt(i)<=i/2<i-1，所以也可为[2,i/2]，当然考试时不必考虑效率问题，可直接把范围写成[2,i-1]或[2,i)，这样不易出错。循环遍历 n 到 m 之间的所有整数，依次判断 i 是否为素数。
- 求紧临 m 的前（或后）的 k 个素数（5 题） 这类题关键是求素数的算法及当已保存素数的个数等于 k 时退出循环。可以在第一个 for 循环语句的条件中加一个条件表达式（如 k>=0）或者在第一个 for 循环循环体中用 if 语句和 break 语句组合退出循环。

④ 求方程的解（10 题） 一般使用迭代算法，while 或 do...while 循环语句，注意循环中止条件。

⑤ 矩阵问题（11 题、12 题、14 题、18 题、42 题、53 题、80 题） 矩阵问题其实就是对二维数组(aa[M][N],bb[M]M])处理，一般是矩阵的转置、加减乘除运算、半三角元素运算及求周边元素的和或平均值。一般题目都给出了算法。关键点：对下标的操作及下标界限，对数组 bb[M][M]应注意 M 的奇偶性，对周边元素求值应注意不要多加四角元素。

⑥ 数的按位分离及合并（15、48、65、71、77、81、84、87、91） 算法是：整除（/）缩减数的位数，取模（%）求位上的数字。

⑦ 四舍五入（16 题） 算法是：若预保留的位数的个数为 n，则先把实数乘以 10 的 n+1 次方；然后对此数加 5 再与 10 整除，其个位将被舍弃；把所得的整数强制转换成实数型再除以 n 就得到最后的结果。

⑧ 整除及奇偶判断问题（27 题、58 题、63 题、98 题） 要注意整除的条件。

（4）结构体和链表

此类型题目所要求的 fun 函数功能比较简单，与数组、指针操作密切相关。关键是对成员的引用，可细分为以下几种：

① 求最大值、最小值和平均值（22 题、34 题、37 题、50 题、54 题、85 题）

② 排序（46 题）

③ 元素分段存放（57 题）

④ 查找元素（67 题、74 题、79 题、89 题）

（5）实际应用

这类题目有统计单词数（62 题）。

第一部分　改错题

★★

第 1 题

下列给定程序的功能是：读入一个整数 k (2≤k≤10000)，打印它的所有质因子（即所有为素数的因子）。例如，若输入整数 2310，则应输出：2、3、5、7、11。

请改正程序中的错误，使程序能得出正确的结果。

注意：不要改动 main 函数，不得增行或删行，也不得更改程序的结构！

试题程序：

```
#include     "conio.h"
#include     "stdio.h"
/********found********/
IsPrime ( int n )
{
    int  i,  m;
   m=1;
/********found********/
   for(i=2;i<n;i++)
        if(!(n%i))
        {
            m=0;
            break;
        }
        return(m);
}

main()
{
    int j, k;
    clrscr();
    printf("\nplease enter an integer number between 2 and 10000:");
    scanf("%d",&k);
    printf("\n\nThe prime factor(s) of %d is(are):",k);
    for(j=2;j<k;j++)
        if((!(k%j))&&(IsPrime(j)))
            printf(" %4d,",j);
        printf("\n");
}
```

【答案】

第 1 处：IsPrime (int n); 应改为 IsPrime (int n)

第 2 处：if !(n%i) 应改为 if (!(n%i))

【解析】

考查知识点：函数声明，表达式运用。

解题思路：首先分析清楚源程序的设计思路，然后在错误标志“/******found*****/”的下方查找错误。对于程序改错题，应先编译查找其中的语法错误，通过编译器的提示很容易找到错误的地方及原因，然后再寻找逻辑错误，修改了语法错误后再次编译，直到修改完所有的语法错误。查找逻辑错误时，需要运行程序根据结果来考查。对于本题，错误标志分别出现在函数定义和条件语句处，此时应该回忆一下它们的注意事项，逐一审查。

答案详解：

对于本题，第一次编译提示函数定义不合法；修改之后再次编译会提示 if 语句缺少“(”，根据编译提示，能很快修改完程序。具体分析如下。

（1）函数在被调用之前必须先定义，否则会导致编译错误。本题的函数 IsPrime 定义不正确，函数定义的形式如下：

```
函数类型  函数名(数据类型 形式参数; 数据类型 形式参数...)  /*此处不能用";"*/
     {
         函数体;
     }
```

（2）if 语句的声明如下：

if(条件) { 语句或复合语句}

循环语句或条件语句的条件必须放在()中，因此本题需要把条件放在()中。

【提高补遗】

如果在函数定义之前用到了被调函数，要使调用函数不发生编译错误，可以在调用函数前进行函数声明（函数声明又叫函数原型，函数可以在一个文件中被声明多次），声明格式如下：

```
函数类型  函数名( );
```

★★★

第 2 题

下列给定程序中，函数 fun 的功能是：逐个比较 a、b 两个字符串对应位置中的字符，把 ASCII 值大或相等的字符依次存放在到 c 数组中，形成一个新的字符串。例如，若 a 中的字符串为 aBCDeFgH，b 中的字符串为：ABcd，则 c 中的字符串应为：aBcdeFgH。

请改正程序中的错误，使它能得出正确的结果。

注意：不要改动 main 函数，不得增行或删行，也不得更改程序的结构。

试题程序：

```
#include <stdio.h>
#include <string.h>
void fun(char *p,char *q,char *c)
{
    /********found********/
    int k=1;
    /********found********/
    while(*p != *q)
```

```
    {
        if(*p<*q)
            c[k]=*q;
        else
            c[k]=*p;
        if(*p)
            p++;
        if(*q)
            q++;
        k++;
    }
}
main()
{
    char a[10]="aBCDeFgH",b[10]="ABcd",c[80]={'\0'};
    fun(a,b,c);
    printf("The string a:");
    puts(a);
    printf("The string b:");
    puts(b);
    printf("The result:");
    puts(c);
}
```

【答案】

第 1 处：int k = 1；应改为 int k = 0;

第 2 处：while(*p != *q) 应改为 while(*p || *q)

【解析】

考查知识点：变量赋初值，循环语句及条件。

解题思路：有两个错误标志，说明有两个错误。先检查语法错误，编译程序后发现没有错误及警告，说明没有语法错误，只有逻辑错误；逻辑错误必须根据程序的功能及预期结果考查。本题的两个错误标志距离很近，第一个错误标志在变量定义处，几乎可以肯定要么是变量的类型错误，要么是变量的初值不正确。对于循环语句，着重考查条件是否存在逻辑错误或者表述是否正确。

答案详解：

（1）fun 函数定义变量 k 并赋初值为 1，k 作为字符串 c 的下标，但 C 语言数组的下标是从 0 开始的（QB 等语言数组下标从 1 开始），因此应改为：int k = 0;。这个错误会导致 c 数组没有输出，因为初始化语句 c[80]={'\0'}，将 c 数组第一个字符单元赋值为'\0'，这种情况下使用 puts()函数是不能输出的，因为该类函数在遇到'\0'时就结束。同样，使用 printf("%s",c)也不能得到预期的输出。

（2）这是一个逻辑错误，根据函数的功能可知，while(*p != *q)循环条件出现有问题，按照此循环条件进行判断时，当遇见一个等值情况就终止了循环。应改为：while(*p || *q);其实这样的答案还是不容易理解，要理解这点需要深入循环，不妨逐条语句分析一下：

```
{
    if(*p<*q) c[k]=*q; /*这句和下句是把 a 或 b 中大的一个值存入 c*/
```

```
    else c[k]=*p;
    /*如果两个数组不等长，就把长的一个数组的剩余部分继续拷贝到 c 数组中*/
    if(*p) p++;
    if(*q) q++;
    k++;
}
```

【提高补遗】

在实际编程中，也可以这样来写 fun 函数，请务必记住这两种思路。

```
void fun(char *p,char *q,char *c)
{
    int k=0;
    while(*q&&*p)                  /*按较短的串处理，也就是条件*q&&*p 表达的意思 */
    {
        if(*p<*q) c[k]=*q;         /*这个 if-else 语句可以用条件赋值语句来代替
                                   c[k]=(*p<*q)? *q: *p ; */
    else c[k]=*p;
        p++;                       /*自然也可以写成*p++; 下行亦同 */
        q++;
        k++;
    }
   /*下面两行把长的那个数组的剩余部分存入 c 数组中 */
    while(*p)c[k++]=*p++;
    while(*q)c[k++]=*q++;
}
```

★★

第 3 题

下列给定程序中，函数 fun 的功能是：依次取出字符串中所有数字字符，形成新的字符串，并取代原字符串。

请改正函数 fun 中的错误，使它能得出正确的结果。

注意：不要改动 main 函数，不得增行或删行，也不得更改程序的结构！

试题程序：

```
#include <stdio.h>
#include <conio.h>
void fun(char *s)
{
    int i,j;
    /********found********/
    for(i=0,j=0;s[i]!='\0';i++)
        if(s[j]>='0' && s[i]<='9')
            s[j]=s[i];
    /********found********/
        s[j]="\0";
}
main()
{
```

```
    char item[80];
    clrscr();
    printf("\nEnter a string :");
    gets(item);
    printf("\n\nThe string is : \%s\n",item);
    fun(item);
    printf("\n\nThe string of changing is : \%\n",item);
}
```

【答案】

第 1 处：s[j]=s[i]；应改为 s[j++]=s[i];

第 2 处：s[j]="\0"；应改为 s[j]= '\0';

【解析】

考查知识点：字符串，循环语句，数据提取整理。

解题思路：有两个错误标志，说明有两个错误。先检查语法错误，编译程序后发现在 s[j]="\0" 处警告，说明可能有错误；修改警告处再次运行时，输出的只是一个字符而不是一个字符串，说明字符串的输出提前结束了，请查看改错题 2 的答案解析（1）。

答案详解：

（1）字符用单引号' '表示，而用双引号" "表示字符串。字符串应以'\0' 结尾，s[j] 是 s 中最后一个元素，表示的是一个字符，应使用单引号' '表示，所以应改为： s[j]= '\0'; 。

（2）函数要把数字保存在字符串 s 中，但是，表示下标的变量 j 一直没有增加，因此 s[j]=s[i]；应改为 s[j++]=s[i]; 。

【提高补遗】

实际编程中，s[j]= '\0'是经常被遗忘的，如果在输出时使用了 puts()函数或 printf("%s",s); 当串中出现了'\0'或者再没有合适的位置给出结束符（'\0'）时，得到的结果就会出错。

从题中可见在原串中构造整理新串的一般做法：

设计两个数组下标指针，其中一个指示原串位置，另一个指向新串位置，原串下标指针在主循环中一直后移，而新串指针仅在存入新值时后移，如果是字符串，还不要忘记在处理完原串后将新串的下一个位置给出结束符（'\0'）。

输出字符串需要结束符（'\0'），也仅是使用了需要结束符来判断字符串已经结束的函数时，才是必须的，有的函数输出字符需要显示传入字符个数，而不是依靠结束符。这是字符串处理函数通常采用的两种判断字符串结束的策略。

★★★

第 4 题

下列给定程序中，fun 函数的功能是：分别统计字符串中大写字母和小写字母的个数。例如，给字符串 s 输入：AaaaBBb123CCccccd，则应输出结果：upper = 5，lower = 9。

请改正程序中的错误，使它能计算出正确的结果。

注意：不要改动 main 函数，不得增行或删行，也不得更改程序的结构！

试题程序：

```
#include <conio.h>
#include <stdio.h>
```

```
/********found********/
void fun(char *s,int a,int b)
{
    while(*s)
    {
        /********found********/
        if (*s>='A'&&*s<='Z')
            a++;
        /********found********/
        if(*s>='a'&&*s<='z')
            b++;
        s++;
    }
}
main()
{
    char s[100];
    int upper=0, lower=0;
    clrscr();
    printf("\nPlease a string : ");
    gets(s);
    fun(s,&upper,&lower);
    printf("\n upper=%d lower=%d\n", upper,lower);
}
```

【答案】

第 1 处：void fun(char *s,int a,int b) 应改为 void fun(char *s,int *a,int *b)

第 2 处：a++; 应改为 (*a)++;

第 3 处：b++; 应改为 (*b)++;

【解析】

考查知识点：函数定义，参数传递及指针运用。

解题思路：有三个错误标志，说明有三个错误，根据错误标志出现思考常见的错误考点。先检查语法错误，编译程序后有两个错误，说函数的参数类型与 main 函数中调用的 fun 函数类型不匹配，修改第一处错误后再次编译，没有任何语法错误。运行并不能得到题目的结果，说明第二、三处错误为逻辑错误，要根据函数的功能进行修改。

答案详解：

（1）main 函数中调用的 fun 函数参数是按指针传递的，因此 fun 函数定义参数表中形参应定义成指针，所以 int a 和 int b 应改为 int *a 和 int *b。

（2）a 和 b 分别为指向 upper 和 lower 的指针，执行 a++和 b++后，a、b 已经不再指向 upper 和 lower，而是指到 upper 和 lower 后面去了。因而 upper 和 lower 的值一直都是 0。程序需要 upper 和 lower 的值递增，因此必须使指针所指的变量发生变化，而不是指针自己变化。所以应改为：(*a)++; (*b)++;。

【提高补遗】

实参与形参之间数值的传递方式有按值传递以及地址传递，缺省为按值传递。参数的传递机制在按值传递时，函数操纵的是实参的本地拷贝，当形参发生改变时，实参值不会

被改动。如果用变量的地址值或指针变量做实参，相应的形参应用同类型的指针变量，这种采用地址值的传送，能达到改变实参内容的目的。如果要求函数有几个返回值，除了 return 语句，还可以采用这种按地址值的传递方式。

★★

第 5 题

假定整数数列中的数不重复，并存放在数组中。下列给定程序中，函数 fun 的功能是：删除数列中值为 x 的元素，同时将其他元素前移。n 中存放的是数列中元素的个数。

请改正程序中的错误，使它能得出正确的结果。

注意：不要改动 main 函数，不得增行或删行，也不得更改程序的结构。

试题程序：

```
#include <stdio.h>
#define N 20
fun(int *a,int n,int x)
{
    int p=0,i;
    a[n]=x;
    while(x!=a[p])
    {
        p=p+1;
    }
    if(p==n)
        return -1;
    else
    {
 /********found********/
        for(i=p;i<n;i++)
            a[i+1]=a[i];
        return n-1;
    }
}
main()
{
    int w[N]={-3,0,1,5,7,99,10,15,30,90},x,n,i;
    n=10;
    printf("The original data:\n");
    for(i=0;i<n;i++) printf("%5d",w[i]);
    printf("\nInput x (to delete):");
    scanf("%d",&x);
    printf("Delete :%d\n",x);
    n=fun(w,n,x);
    if(n==-1)
        printf("***Not be found!***\n\n");
    else
    {
        printf("The data after delete :\n");
```

```
        for(i=0;i<n;i++)
            printf("%5d",w[i]);
        printf("\n\n");
    }
}
```

【答案】

第 1 处：a[i+1]=a[i]; 应改为 a[i]=a[i+1];

【解析】

考查知识点：数组下标操作，查找算法，在原数组上翻新数组元素。

解题思路：有一个错误标志，说明有一个错误（一般就是错误标志下一个语句，可能占几行）。编译程序后没有发现语法错误。运行程序，删除一个数组中的值后看看删除结果，很容易就能发现问题。

答案详解：

fun 函数功能是删除一个给定的值，首先是从第一个元素开始顺序查找，直到最后一个元素，如果没有查找到给定的数字，返回-1；如果查找成功，把后面的数组元素依次向前移动一位，即：a[i]=a[i+1]，删除成功，元素的个数 n 减 1。所以 a[i+1]=a[i]; 应改为 a[i]=a[i+1];

【提高补遗】

本题用到了查找算法，从表列中查一个数的最简单方法是从第一个数开始顺序查找，将要找的数与表列中的数一一比较。直到找到为止，如果表列中无此数，则应找到最后一个数，然后判定“找不到”（如上题 fun 函数中，当 p==n 时表示查完最后一个元素）。这就是“顺序查找法”，其优点在于可以对原表列不做排序；而缺点在于查找的效率较低。

如果上题给出的数组中有重复的元素，fun函数又应该如何呢？

```
fun(int *a,int n,int x)
{
    int p,i,m=0;                        /*p 为数组下标，m 用于记录数组中跟查找元素相
                                          同的的元素的个数 */
    for(p=0;p<n-m;p++)                  /* 顺序查找数组中的每一个元素，n-m 表示数组
                                          元素的个数，如果找到了被查找值，数组长度减小，
                                          即（n-m） */
    {
        if((p<n-m)&&(a[p]==x))          /* 判断数组中的元素是否跟被查找数相同的元素*/
        {
            m++;                        /* 如果查找成功，元素的个数加 1 */
            for(i=p;i<n-m;i++)          /* 删除查找到的元素 */
            {
                a[i]=a[i+1];
            }
        }

    }
    return (m>0 ? n-m:-1);              /* 使用条件操作符判断删除是否成功，成功则返回
                                          数组的大小，失败则返回 -1 */
}
```

【课外作业】