第五次作业参考答案

钟文杰　　2025.10.30

作业 9

P156 T3

编写函数，求一个数的立方根，若立方根不是整数，则返回其整数部分。主函数从键盘输入一个数，调用此函数，输出此数的立方根的整数部分。

作业反馈：很多同学选择做这道题，但做对的同学并不多，问题主要出在两方面：忽略了输入的数可能是负数，或者对整数部分的定义不清楚。

Tips:

1. 类似于`for(int i=0;i*i*i<=n;i++)`的循环都会在 n 为负数时陷入死循环。

2. 整数部分 $[x]$ 的定义是不大于 x 的最大整数，所以要注意各种情况下函数的返回值。

参考代码 1:

```
//思路：找一个整数使得该整数的立方不大于给定的数
#include<stdio.h>
int main(){
    int m=0;
    float n=0;
    scanf("%f",&n);
    if(n>=0){
        for(;m*m*m<=n;m++);
        printf("%d\n",m-1);
    }
    else{
        for(;m*m*m>=n;m--);
        if((m+1)*(m+1)*(m+1)==n) printf("%d\n",m+1);
        else printf("%d\n",m);
    }
    return 0;
}
```

参考代码 2:

```
//思路：利用二分法大致求得原数的立方根，再输出整数部分
#include<stdio.h>
int main(){
    int m=0;
    float n=0,L=0,R=0,mid=0;
    scanf("%f",&n);
    if(n>=0) R=n;
    else L=n;
    while(R-L>0.01){//既然只是求整数部分，没有必要太精确
        mid=(L+R)/2;
        if(mid*mid*mid>=n) R=mid;
        else L=mid;
    }
    m=(int)R;//隐式转换，会强行舍弃小数点后的数字，无论正负
    if(m*m*m<=n) printf("%d\n",m);//无论取L还是R的整数部分
    //都有可能恰好与n的立方根之间隔着一个整数，因此还需要判断
    else printf("%d\n",m-1);
    return 0;
}
```

P156 T5

编写函数，其功能是在 float 型一维数组中查找最大值并返回到调用程序。

作业反馈：这道题基本上大家写的都没问题。

参考代码：

```
//思路：设置一个变量max,每次遇到比已储存的值大的数就改成那个数
//将数组从小到大排序后取最后一个数也可，但不如这个来的简单
#include<stdio.h>
float maxm(float* a,int n){
    float max=a[0];//必须先给max赋值为a中的某个值
    //否则可能会出现不存在于a中的初值反而成了最大值的情况
    for(int i=1;i<n;i++){
        if(a[i]>max) max=a[i];
    }
```

```
    return max;
}
int main(){
    float a[10]={1.1,2.3,-9.7,666,-53};
    printf("%f\n",maxm(a,5));
    return 0;
}
```

P156 T7

求两个整数的最大公约数和最小公倍数。要求编写两个函数，一个函数求最大公约数，另一个函数根据求出的最大公约数求最小公倍数。在主函数中由键盘输入两个整数后调用这两个函数，并输出结果。

作业反馈：这道题需要注意的是题目要求写两个函数分别求最大公约数和最小公倍数，并且根据最大公约数求最小公倍数，因此所有在主函数里解决问题/不借助最大公约数来求最小公倍数的做法都是不够好的。

Tips: 之所以这道题不需要考虑输入的整数是负数的情况，是因为讨论最大公约数和最小公倍数时确实不考虑正整数之外的情况。你可以将这两个概念推广到负数甚至分数情况，但是这并不是这门课该考虑的事情 hhh.

参考代码：

```
//思路：用辗转相除法求最大公约数，直接用a*b/gcd(a,b)求得最小公倍数
#include<stdio.h>
int _gcd(int a,int b){//函数名前加下划线避免与下面设置的变量重名
    //给变量名前面加下划线也是一样的效果，但习惯上给函数名前面加
    int r=a%b;//若a<b则r=a,因此不用特意判断a,b谁大
    while(r!=0){//这里可以直接写!r,想想为什么?
        a=b;
        b=r;
        r=a%b;
    }
    return b;
}//gcd=Greatest Common Divisor
int _lcm(int a,int b){
    return a*b/gcd(a,b);
}//lcm=Least Common Multiple
int main(){
```

```
    int a=0,b=0,gcd=0,lcm=0;
    scanf("%d%d",&a,&b);
    gcd=_gcd(a,b);
    lcm=_lcm(a,b);
    printf("%d %d\n",gcd,lcm);
    return 0;
}
```

P156 T8

用结构体类型作为函数的类型，就可以同时返回多个值，请将例 4.2-4 的数组改成结构体类型，修改程序以使其能返回交换后的数据，思考与数组形式有什么不同。

作业反馈：这道题的程序可以写成像数组一样，但那样写成结构体就没什么意义了不是吗 ()

参考代码：

```
//思路：如题目描述
#include<stdio.h>
struct two{
    int a,b;
};
struct two swap(struct two pair){
    int t=pair.a;
    pair.a=pair.b;
    pair.b=t;
    return pair;
}
int main(){
    struct two pair={3,5};//不建议像课本一样起重名变量
    printf("pair=(%d,%d)\n",pair.a,pair.b);
    pair=swap(pair);
    printf("pair=(%d,%d)\n",pair.a,pair.b);
    return 0;
}
```

P156 T9

如何将例 4.2-8 中函数 max4 的函数体改成只有一条 return 语句，其后的表达式可以调用 max2 函数？

作业反馈：大家写的基本都对，方式不唯一。

参考代码：

```
//思路：仿照max2，运用三目运算符
#include<stdio.h>
int max2(int a,int b){
    return (a>b)?a:b;
    //在使用三目运算符时，把问号前面的条件表达式用括号括起来是好习惯
    //可以更轻松的检查逻辑有没有出错
}
int max4(int a,int b,int c,int d){
    return max2(max2(a,b),max2(c,d));
}
int main(){
    int a=0,b=0,c=0,d=0;
    scanf("%d%d%d%d",&a,&b,&c,&d);
    printf("%d\n",max4(a,b,c,d));
    return 0;
}
```

note: 虽然在求两数中较大者、求绝对值等多种场合三目运算符可以有效简化代码，但我实际上非常不建议大家在实际编程中使用三目运算符（尤其是大量使用），原因有二：

一、三目运算符起到的是分支语法中的 if（条件判断）的效果，利用三目运算符压缩程序在后期 debug 中很容易把自己绕进去。

二、三目运算符很容易写的特别长，也经常牵扯很多各变量。甚至在很多三目运算符惯用者的程序里，某几个多重三目运算符赋值语句就是程序的绝对核心，这对于其他不常用三目运算符的程序员阅读这个程序的体验是毁灭性的打击。

P157 T14

这道题不是作业题，只是我在写答案的时候看错题了，写都写了就放上来吧

设计与实现接收数组并对其中的数据进行降序排序的插入法排序函数。

参考代码：

```
//思路：利用地址形参实现函数直接对待排序数组做改动
#include<stdio.h>
void sort(int* a,int n){//没有必要返回任何值的函数就设置为void类型
    for(int i=1;i<n;i++){
        int t=a[i],j=0;
        for(;j<i;j++){
            if(a[j]<t) break;
        }//这个循环也可以写成for(;a[j]<t;j++); 想想为什么?
        for(int k=i-1;k>=j;--k){//写循环一定要注意终止条件
            a[k+1]=a[k];
        }
        a[j]=t;
    }
    return;
}
int main(){
    int a[10]={1,2,3,99,45,32,193};//图方便直接内置值
    sort(a,7);
    for(int i=0;i<6;++i) printf("%d ",a[i]);
    printf("%d\n",a[6]);//最后一个值后面不输出空格而是输出回车
    //适用于要求数值后不能有多余空格、回车的OJ
    return 0;
}
```

note: 有同学对程序的时间复杂度这一概念不大了解，这里作简单的介绍，有兴趣的同学可以进一步与助教交流或查询网络资料。

时间复杂度是算法执行时间关于问题规模的函数，它体现了该算法处理问题耗时随问题规模变大的变化趋势。计算（或者说估算，因为我们后面就会看到我们实际上并不关注也没法得到这个函数的具体表达式，我们只考虑最大的若干项）时间复杂度基于以下假设：

0. 程序是可以在有限时间内结束的。（序号 0 表示这是一切的前提，后面的 123... 都是并列关系，但这条跟他们不并列）

1. 基础运算（如加减乘除，关系运算）消耗的时间是常数量级，即我们忽略参与运算的数字变大导致的基础运算用时延长。

2. 程序能够处理大规模的问题，因而常数相对于问题的规模 n 来说是可以忽略的。（比如我们认为 $2n$ 和 n 是一样大的）

3. 一般来说时间复杂度指的是算法在不同情况下的平均耗时，如果专门考虑最优（或最差）情况我们会专门称其为最优（最差）情况下的时间复杂度。

以上述的程序为例，赋 n 个初值（或者读入 n 个值）需要 n 次运算，因而时间复杂度为 $O(n)$（这个记号表示处理问题的时间是 n 量级）而后续的排序过程需要对 n 个值做双循环，第一个循环从 1 到 $n-1$, 第二个从 0 到 $i-1$，又从 $i-1$ 到某个 j，因而最多需要 $2*(1+2+...+n)=n*(n+1)$ 次操作。但我们前面假设，常数对于 n 来说是可以忽略的，因此这里的时间复杂度是 $O(n^2)$。后面输出的时间复杂度同样为 $O(n)$, 总计时间复杂度是 $O(n^2+2n)$. 但我们只考虑最大的项，从而总时间复杂度是 $O(n^2)$.

我们常说的优化算法一般都是在减小算法的时间复杂度，程序本身的长度和运行时占用的内存都不那么重要。

P157 T15

将 4.3.1 节的程序改成从文件中读取 10 个整数，然后用选择法排序，最后输出中位数。

作业反馈：这道题有一些人选，但真的做了文件读写的同学好像就一两个，大家可以自己了解一下文件读写，后面的课程应该也会涉及。

字符串	说明
r	以只读方式打开文件，该文件必须存在。
r+	以读/写方式打开文件，该文件必须存在。
rb+	以读/写方式打开一个二进制文件，只允许读/写数据。
rt+	以读/写方式打开一个文本文件，允许读和写。
w	打开只写文件，若文件存在则文件长度清为零，即该文件内容会消失；若文件不存在则创建该文件。
w+	打开可读/写文件，若文件存在则文件长度清为零，即该文件内容会消失；若文件不存在则创建该文件。
a	以附加的方式打开只写文件。若文件不存在，则会创建该文件；如果文件存在，则写入的数据会被加到文件尾后，即文件原先的内容会被保留（EOF 符保留）。
a+	以附加方式打开可读/写的文件。若文件不存在，则会创建该文件，如果文件存在，则写入的数据会被加到文件尾后，即文件原先的内容会被保留（EOF符不保留）。
wb	以只写方式打开或新建一个二进制文件，只允许写数据。
wb+	以读/写方式打开或新建一个二进制文件，允许读和写。
wt+	以读/写方式打开或新建一个文本文件，允许读和写。
at+	以读/写方式打开一个文本文件，允许读或在文本末追加数据。
ab+	以读/写方式打开一个二进制文件，允许读或在文件末追加数据。

图 1: **fopen 参数图**

参考代码：

```
/*
思路：这一题要求从文件中读入数据，因此需要用到文件读取函数fscanf，
文件读写相关内容可以参考课本3.4节，即P74,75

下面的程序里就顺便实现了将结果用fprintf函数写入文件里的功能，
对文件读写有兴趣的同学可以与助教进一步交流或者查询网络资料
*/
#include<stdio.h>
#include<stdlib.h>//写文件读写建议调用这个库，利用exit函数中断程序
int input_data(int* a,int m){//这个m表示最多读入多少个数
    int i=0;
    FILE *fp=fopen("P157_15_in.txt","r");
    //文件与程序在同一文件夹时可以只写文件名，否则要写完整路径
    //打开文件，设置读写模式为“只读”，并用fp存储该文件的“地址”
    if(fp==NULL){//若未能成功打开/创建文件则退出程序
        puts("Error!");
        exit(-1);//直接退出整个程序，不管目前在不在主函数里
    }//养成好习惯，进行文件读写前必须要做这一步检查!
    for(;(i<10)&&(i<m);++i){
        //读入数量不能超过数组大小，且按要求最多读入10个数。
        //去掉中间的i<10可以让程序读入文件内所有的数据
        if(fscanf(fp,"%d",&a[i])==EOF) break;
        //从fp指向的文件中读入数据并且判断有没有读到EOF
        //EOF代表文件内容结束，停止读入
    }
    fclose(fp);//读入完成，必须要关闭文件。
    //如果后面还有用，到别的函数里关也行，程序结束前关掉就行
    return i;
}
void sort(int* a,int n){//按题目要求，使用选择排序
    int min=0,tmp=0,m=0;
    for(int i=0;i<n;i++){
        min=a[i];//重新给min赋初值，避免不在a中的初值反而成了最小值
        m=i;//m表示最小值具体在哪一个位置
        for(int j=i+1;j<n;j++){
```

```
            if(min>a[j]){
                min=a[j];
                m=j;
            }
        }
        tmp=a[i];
        a[i]=min;
        a[m]=tmp;
    }
    return;
}
double find_median(int* a,int n){
    double med=0;
    sort(a,n);
    if(n%2==0) med=(a[(n-1)/2]+a[(n+1)/2])/2.0;
    else med=a[(n+1)/2];
    return med;
}
void output_data(double m){
    FILE *fp=fopen("P157_15_out.txt","w");
    //以“只写”模式打开文件,若不存在则创建。覆盖文件内已有的内容
    //不想覆盖可以用"a"模式，可选模式有很多，见上方图1
    if(fp==NULL){
        puts("Error!");
        exit(-1);
    }
    fprintf(fp,"%lf",m);
    fclose(fp);
    return;
}
int main(){
    int a[20]={0},n=0;
    //预留5格给函数读入，再预留5格防止越界，所以数组大小开到20
    double med=0;
    n=input_data(a,15);
```

```
    med=find_median(a,n);
    output_data(med);
    return 0;
}
```

P157 T17

编写函数，其功能是输入的一个字符串按反序存放，在主函数中输入和输出字符串。

作业反馈：这道题基本上大家写的都没问题，但是部分同学没有单独写一个函数。

参考代码：

```
//思路：从头到中间逐个将字符与其对称位置的字符交换后输出
#include<stdio.h>
#include<string.h>//需要使用这个库里头的strlen函数
void reverse(int len,char* str){
    for(int i=0;2*i<len-1;i++){
        char tmp=0;
        tmp=str[i];
        str[i]=str[len-1-i];
        str[len-1-i]=tmp;
    }
    return;
}
int main(){
    int n=0;
    char str[105]={0};//定义数组时预留5格空位，保证不会出现越界问题
    scanf("%s",str);
    n=strlen(str);
    reverse(n,str);
    puts(str);
    return 0;
}
```

P157 T18

编写函数，将两个字符串连接，在主函数中输入字符串并输出字符串连接结果

作业反馈：这道题基本上大家写的也没问题，但是也有部分同学没有单独写一个函数。

参考代码：

```
//思路：把str1的字符逐一赋进str中，再将str2的字符中贴到后面
#include<stdio.h>
#include<string.h>
void _strcat(char* str1,char* str2,char* str){
    //cat是指concatenate:v.连接
    //注意我们引用了string.h库，因此函数不能与这个库的strcat函数重名
    int l1=strlen(str1),l2=strlen(str2);
    for(int i=0;i<l1;i++){
        str[i]=str1[i];
    }
    for(int i=0;i<l2;i++){
        str[i+l1]=str2[i];
    }
    return;
}
int main(){
    char str1[105],str2[105],str[205];
    scanf("%s %s",str1,str2);
    _strcat(str1,str2,str);
    puts(str);
    return 0;
}
```

作业 10

P157 T19

已知凸五边形的 5 个顶点坐标，求五边形的面积。

作业反馈：算面积这一功能实现起来比较简单，就不多赘述了。判断五个顶

点按输入的顺序连线是否能组成一个凸五边形这一功能大家的思路基本都是统一的：判断边向量的外积是否保持方向统一，从而边向量是沿同一方向旋转的。(或者说内角没有优角)

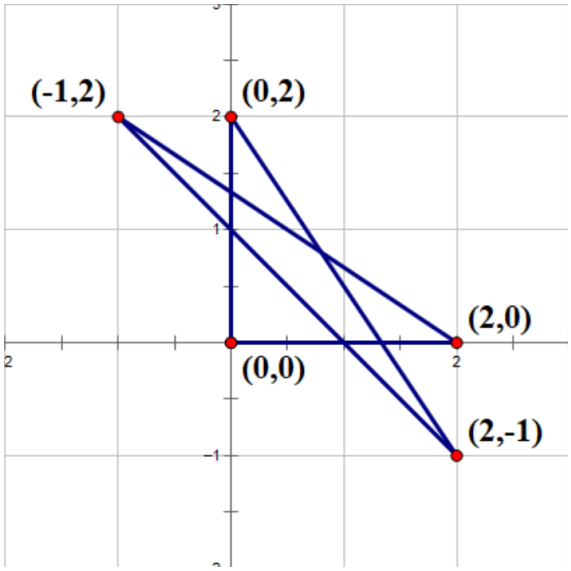

图 2: **五角星例子**

然而只做这一判断会将五角星也误认为是凸五边形 (如图 2 所示)，因此我们需要额外的判断，比如检查边向量是否只转了一圈，具体可参考下方代码：(代码块里好像显示不出 π, 就直接打 \pi 了，大家能看懂就好)

参考代码：

```
/*
思路：判定凸多边形的一个直接方法是：检查n个外角之和是否为2\pi.
其中外角定义为用\pi减去对应的内角度数
而这等价于检查多边形的边（对应的向量）是否沿同一个方向转过一圈
*/
#include<stdio.h>
struct point{
    float x,y;//x,y分别是点的横、纵坐标
};
int check(struct point* l){
    /*
    检查五个边向量是否沿同一方向转过一圈
    怎么检查向量转动方向呢？我们用向量的外积（俗称叉乘）来检查
    两个向量的外积为正说明向量逆时针转动(0,\pi),
```

```
    为负说明向量逆时针转动(0,\pi),
    为0说明转动角度为0或\pi.
    */
    int book=0;//在下面这个循环里，book与向量外积保持同号
    for(int i=0;i<5;i++){
        int vcp=l[i].x*l[i+1].y-l[i].y*l[i+1].x;
        //vcp=Vector Cross Product,向量外积
        if(vcp==0){
            printf("vcp[%d]=0\n",i);
            return 0;//第i条边与第i+1条边共线，这不是凸五边形
        }
        if(book==0) book=(vcp>0)?1:-1;
        else if(vcp*book<0){
            printf("angle %d rotates a different way\n",i+1);
            return 0;
        //从第i条边转到第i+1条边的转动方向与之前不同，这不是凸五边形
        }
    }//没有在这个循环中返回主函数，就说明边向量是沿同一个方向转动的

    book=0;
    //在下面这个循环里，book表示边向量转了多少圈
    //主函数里解释过，边向量一定转了整数圈
    for(int i=0;i<5;i++){
        if((l[i].y<0)&&(l[i+1].y)>=0) book++;
        //把边向量从y轴下方转到y轴上方作为转了一圈的标记
    }
    printf("book=%d\n",book);
    if(book==1) return 1;//恰好转了一圈，这是一个凸五边形
    else return 0;//不止转了一圈，这不是凸五边形（是五角星）
}
int main(){
    struct point p[7]={0},l[7]={0};
    //p是储存五个顶点坐标的数组,l是储存五条边对应的向量的数组
    for(int i=0;i<5;i++){
        scanf("%f%f",&p[i].x,&p[i].y);
```

```
    }
    p[5]=p[0];//为了方便地写后面的循环，补充定义第6个点与第1个点重合
    for(int i=0;i<5;i++){
        l[i].x=p[i+1].x-p[i].x;
        l[i].y=p[i+1].y-p[i].y;
        printf("%f,%f\n",l[i].x,l[i].y);
    }
    l[5]=l[0];//让第6条边与第1条边重合
    //因此边向量从第1条边转到第6条边一定转了整数圈
    if(check(l)==1) puts("This is a pentagon");
    else puts("This is not a pentagon");
    return 0;
}
```

note: 有一位同学的做法是：对每一个点 P_n，求出以所有顶点的坐标平均点 O 为起点，到这个点的射线 OP_n 的倾角 θ_n, 然后将点按 θ_n 从小到大排序，再检查边向量是否沿同一个方向转动。由于倾角 $\theta \in [0, 2\pi)$, 他这么做就自然保证了边向量最多只转一圈。这个做法非常好，实际上它稍加改动就能非常轻松地实现接下来提到的两个拓展功能，而且是这一问题的最佳解决方案。

Extension problems:

(1). 判断给定的 5 个顶点是否能按某种顺序连边来得到一个凸五边形。

(2). 在 (1) 的要求上，如何找到一个可行的顺序。（运用数学知识我们知道实际上这个顺序忽略旋转忽略翻转是唯一的）

对于这两个要求, 我们很容易想到一个办法：把 5 个顶点所有可能的顺序全部试一遍。然而即使将旋转、翻转后相同的顺序去除，也要枚举 12 种顺序。如果有 n 个顶点希望找到凸 n 边形，那足足需要枚举 $(n-1)!/2$ 种顺序。更别提验证一个顺序是否可行还需要至少去算 n 次外积，这显然是很低效的。上述那位同学的做法则仅仅需要 $O(n \log n)$ 的时间复杂度。(排序可以用 $n \log n$ 量级的操作次数解决，而接下来只要验证这一个顺序是否可行就可以解决问题)

证明没有算法能用更小的时间复杂度解决上述的两个问题是比较困难的，但是可以凭直觉理解一下：解决上述问题需要验证一些可能的 n 个点的排列顺序是否能构成凸 n 边形，上述解法仅使用了排序算法就将需要验证的顺序个数降低至 1 个。这要求更优的算法要么用比 $O(n \log n)$ 更快的速度找到这个唯一可能的顺序，要么用 $O(n)$ 的速度找到不超过 $\log n$ 个顺序，验证完它们之后就能解决上述问题，而这是做不到的。

我们还有如下更进一步的问题：

如果不允许验证某个具体的顶点顺序是否组成凸 n 边形，如何解决问题 *(1)?*

解决这个问题很难绕开“凸包”及相关概念，当然也已经远超本课程内容，这里不再继续展开。欢迎有兴趣的同学查阅相关资料以及与我交流。(/doge)

P157 T20

编写函数 1，使得给定的一个 4×4 的二维整型数组转置；扩展函数为函数 2，将给定的一个 $N\times N$ 的二维整型数组转置；再编写函数 3，从键盘输入一个整数 N, 再从键盘输入 $N\times N$ 个整数，为 $N\times N$ 的二维整型数组赋值，最后按行列逻辑输出这个数组。

作业反馈：实际上绝大部分选做了这一题的同学都没有按要求把输入功能写进函数 3 里，但是问题不大，我改作业的时候也没有因此认为大家做错了。(毕竟一般而言，写程序要么单独写个函数读入数据，要么就在主函数读入数据，很少有把读入和实际功能放到一起的)

参考代码：

```
//思路：如题目描述
#include<stdio.h>
void transpose4(int* m){
    int tmp=0;
    for(int i=0;i<4;i++){
        for(int j=i+1;j<4;j++){
            tmp=*(m+i*4+j);
            *(m+i*4+j)=*(m+j*4+i);
            *(m+j*4+i)=tmp;
        }
    }
    return;
}
void transposeN(int *m,int n){
    int tmp=0;
    for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++){
            tmp=*(m+i*n+j);
            *(m+i*n+j)=*(m+j*n+i);
            *(m+j*n+i)=tmp;
        }
```

```
    }
    return;
}
void work(int *m,int n){//n表示二维数组第二维的大小
    int k=0;
    scanf("%d",&k);
    for(int i=0;i<k;i++){
        for(int j=0;j<k;j++){
            scanf("%d",m+i*n+j);
            //m本身就是地址，不用再加取地址符了
        }
    }
    for(int i=0;i<k;i++){
        for(int j=0;j<k-1;j++){
            printf("%d ",*(m+i*n+j));
            //调用二维数组中m[i][j]的方式，后面学到二维数组会详细讲
            //这里也可以写m[i*n+j],效果是一样的
        }
        printf("%d\n",m[i*n+k-1]);
    }
    return;
}
int main(){
    int a[15][15]={{0,0,0,0},{0,0,0},{0,0}};
    //多维数组赋初值类似于数学中表示以集合为元素的集合
    work(*a,10);//编译器会把a理解成指向一维数组的指针
    //因此*a指向的就是第一行对应的一维数组的首地址
    return 0;
}
//注：运行这个程序，你会发现矩阵的数并没有对齐，想想怎么将其对齐？
```

P157 T21

编写程序，找出 100 1000 之间的所有可逆素数。要求：判断素数、颠倒顺序的功能用两个自定义函数实现

作业反馈：这道题也有部分同学没有按要求编写两个函数，请注意题目要求。

还有一个问题：有的同学主函数里循环变量可以等于 1000，却仍然使用了 100× 个位 +10× 十位 + 百位的逆转方法，这是有问题的。

Tips: 我知道有的同学记忆中可逆素数的定义要求该数不能是回文数，但这道题没有说不能，所以就按允许素数是回文素数来写答案，大家考场上遇到类似这种情况也要注意题目的定义跟自己熟识的定义是否一致。

参考代码：

```
//思路：从100到1000逐一判断每个数是否满足要求，输出其中满足要求的数
#include<stdio.h>
int is_prime(int n){
    int book=1;//book=1表示n是素数，book=0则n不是素数
    for(int i=2;i*i<=n;i++){
        //我们熟知任何合数n一定会有不大于n的算数平方根的因子
        if(n%i==0){
            book=0;
            break;//找到n的一个非平凡因子就够了
        }
    }
    return book;//book有登记，记录的意思
    //也有人喜欢用mark,token等词来命名有标记意义的变量
}
int reverse(int n){
    int rev=0;
    while(n!=0){
        rev*=10;
        rev+=n%10;
        n/=10;
    }
    return rev;
}
int main(){
    int count=0;
    for(int i=100;i<1001;i++){
        if(is_prime(i)&&is_prime(reverse(i))){
            if(count==5){//6个一行，方便阅读
                printf("%d\n",i);
```

```
                count=0;
            }
            else{
                printf("%d ",i);
                count++;
            }
        }
    }
    return 0;
}
```