《计算机程序设计》作业 №-05及第4次上机

作业四内容要点： 数组 循环 排序

【姓名_____学号_____】

【要求】

(一)在计算机上编程程序，加上必要的注释。
(二)上机实验，经助教检查通过后，复制源码并记录实验结果，完成报告。
(三)实验报告：记录调试及改错过程；知识点或方法技巧的收获心得.

1 、 成绩统计。

一个班有30个学生，每个学生有三门课。输入全部成绩，并计算每个人三门课的**平均成绩**，统计**平均85分及以上的人数**，用冒泡排序法将平均成绩按照从高到低的顺序**排序**、并**输出**排序后的平均成绩。

将下面的数据信息另存为一个文本文件 score.txt，使用**输入重定向**从该文件输入数据。

1	46	95	77
2	66	88	15
3	74	87	80
4	36	73	71
5	76	25	69
6	76	82	68
7	13	91	38
8	96	80	90
9	83	80	42
10	30	71	83
11	54	95	74
12	97	77	100
13	81	93	67
14	0	83	79
15	34	92	59
16	83	85	96
17	59	83	50
18	73	80	24
19	75	65	68
20	57	77	52
21	97	88	85
22	55	88	74
23	56	83	33
24	93	80	85
25	38	60	80
26	77	86	71
27	85	86	83
28	35	82	56
29	44	90	74
30	96	82	77

(一)【源码】

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

```c
int main(){
    int num,i,t=0;//num为学号,t为85分及以上的人数
    float avgscore[30],scr1,scr2,scr3,c;//avgscore为每人的平均分；scr1~3为3科各自的成绩，c是冒泡排序法中用来调换顺序的临时变量
    FILE *fp;
    fp = freopen("/Users/page/Documents/4-app/Homework/All_Homework_c/120501/120501/in.txt","r",stdin);//将"输入"重定向到文件"in.txt"
    /*
     <stdio.h>中的
     freopen("D:\\in.txt","r",stdin)的作用,
     就是把标准输入流stdin重定向到D:\\in.txt文件中,
     这样在用scanf(C语法)或是用cin(C++语法)输入时便不会从标准输入流读取数据,
     而是从in.txt文件中获取输入。（"输入"是指从外部文件中获取输入）
     只要把输入数据事先粘贴到in.txt中即可。

     类似的,
     freopen("D:\\out.txt","w",stdout)的作用,
     就是把stdout重定向到D:\\out.txt文件中(若有原文件，则抹除后输入；若无，则新建一个),
     这样输出结果就可以通过打开out.txt文件查看。
     */

for(i=0;(fscanf(fp,"%d%f%f%f",&num,&scr1,&scr2,&scr3)!=EOF)&&i<=29;i++){//四个数据为一组是一个人的数据
        /*fscanf(文件，提取对象，指针)意为从文件fp中依次提取一组符合要求的数据并存放到变量中*/
        avgscore[i]=(scr1+scr2+scr3)/3.0;//计算平均数并赋值给average[i]
    }//得到全班的平均分
    //冒泡排序法
    for(int j = 29; j > 0 ; j--) {
        for(int i = 30; i > 29 - j; i--) {
            if(avgscore[i] > avgscore[i - 1]) {
                c = avgscore[i];
                avgscore[i] = avgscore[i - 1];
                avgscore[i - 1] = c;
            }
            else{
                continue;
            }
        }
    }
    fclose(fp);//关闭定向输入
    for(int i = 0; i < 30; i++) {//输出排序后的平均值
        printf("第%d名：平均%f分\n",i+1, avgscore[i]) ;
        if(avgscore[i]>=85){
            t++;
```

```
        }
    }
    printf("\n平均分85分及以上共有%d人。\n",t);//输出85分及以上总人数
    return 0;
}
```

(二)【运行结果】

```
第1名：平均91.333336分
第2名：平均90.000000分
第3名：平均88.666664分
第4名：平均88.000000分
第5名：平均86.000000分
第6名：平均85.000000分
第7名：平均84.666664分
第8名：平均80.333336分
第9名：平均80.333336分
第10名：平均78.000000分
第11名：平均75.333336分
第12名：平均74.333336分
第13名：平均72.666664分
第14名：平均72.333336分
第15名：平均69.333336分
第16名：平均69.333336分
第17名：平均68.333336分
第18名：平均64.000000分
第19名：平均62.000000分
第20名：平均61.666668分
第21名：平均61.333332分
第22名：平均60.000000分
第23名：平均59.333332分
第24名：平均59.000000分
第25名：平均57.666668分
第26名：平均57.333332分
第27名：平均56.666668分
第28名：平均56.333332分
第29名：平均54.000000分
第30名：平均47.333332分

平均分85分及以上共有6人。
Program ended with exit code: 0
```

(三)【实验报告】

应将冒泡排序法改成从大到小排

使用freopen（“r”）实现输入重定向

使用fscanf实现获取输入

2 、合并两个数组。

现有数组 int a[20], b[10]; 对数组初始化: a[0]..a[9]中按照从小到大顺序存放了10个整数：{10，20，30，40，50，60，70，80，90，100}，b中有10个无序的整数 {1，43，72，23，9，87，54，3，55，0}。

编程序把b中的数据**合并**到a中，并保持**a数组仍然有序**。

(一)【源码】

```
//
//  main.c
//  120502
//
//  Created by 李佩哲 on 2021/10/27.
//

#include <stdio.h>

int main() {
    int a[20] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
    int b[] = {1, 43, 72, 23, 9, 87, 54, 3, 55, 0};
    for(int i = 0; i < 10; i++) {
        a[10 + i] = b[i];
    }
    int c;
    for(int j = 0; j < 19; j++) {
        for(int i = 0; i < 19 - j; i++) {
            if(a[i] > a[i + 1]) {
                c = a[i];
                a[i] = a[i + 1];
                a[i + 1] = c;
            }
            else
                continue;
        }
    }
    for(int i = 0; i < 20; i++) {
        printf("%d", a[i]) ;
```

```
        if(i < 19)
            printf(",");
        else
            printf("\n");
    }
    return 0;
}
```

(二)【运行结果】

```
0,1,3,9,10,20,23,30,40,43,50,54,55,60,70,72,80,87,90,100
Program ended with exit code: 0
```

(三)【实验报告】

先合并两个数组，再进行冒泡排序

3 、 约瑟夫环

约瑟夫(Josephus)问题是由古罗马的史学家约瑟夫提出的,他参加并记录了公元 66-70 年犹太人反抗罗马的起义。约瑟夫作为一个将领带兵驻守裘达伯特城,在城市沦陷之后,他和 40多名将士在附近的一个洞穴中避难，将士们群情激奋地表示宁死不投降，于是约瑟夫提出自杀规则：所有n个人围坐一圈，从第一个人开始从1到m报数，报到m的人在战友协助下有尊严地结束其生命（对就是杀了他），下一个人重新报数，依次执行，直到最后一人则英勇而悲壮地自杀。故事的结尾显然约瑟夫同志留到了最后，他决定去当一个历史学家记录战士们的英勇事迹☺ 。

现在假设你就是约瑟夫，你也想当历史学家 ☺ 。需要解决当任意给定 n和 m 后，求出最后留下来的人的编号。

编程序，设置数组 **soldiers[100], 输入n (设n<=100)和m,** 输出最后一个剩下的编号。

例如：

输入n=5 ， m= 2

前四个被杀死人的顺序是： 2, 4, 3, 1, 最后留下来的是5

输出 5

(一)【源码】

```
//
//  main.c
```

```c
//  120503
//
//  Created by 李佩哲 on 2021/10/27.


#include <stdio.h>

int main() {
    int n, m;
    printf("n,m=");
    scanf("%d%d", &n, &m );
    int soldiers[100];
    for(int i = 0; i < n; i++) {
        soldiers[i] = 1;//设置总人数
    }
    for (int i=n; i<100; i++) {
        soldiers[i]=0;
    }
    int t = 0;
    for(int i = (m - 1)%n; t < n - 1 ; t++) {//激动人心的连环杀人环节
        soldiers[i] = 0;//杀人
        printf("kill i=%d\n",i);
        if(i > n - 1) {//判断超范围了没，超了就从头再来
            i = i % n;//回到开头
        }
        int r=0;
        while (r < m) {//找下一个人的编号
            i=(i+1)%n;
            LOOP:switch (soldiers[i]) {
                case 0:
                    i=(i+1)%n;
                    goto LOOP;
                    break;

                case 1:
                    r++;
                    break;

                default:
                    break;
            }
        }
    }

    for(int i = 0; i < n; i++) {//输出活着的人
        if(soldiers[i] != 0) {
            printf("%d survives.\n", i + 1);
        }
    }
    return 0;
}
```

(二)【运行结果】

```
n,m=10 15
kill i=4
kill i=0
kill i=8
kill i=9
kill i=3
kill i=2
kill i=7
kill i=6
kill i=1
6 survives.
Program ended with exit code: 0
```

(三)【实验报告】

Switch…case…

以及goto语句可以完成对死亡人的跳过

4 、 矩阵相乘。

设有两个矩阵为：

$$A=\begin{bmatrix}1&2&3&4&5\\6&7&8&9&10\\11&12&13&14&15\\16&17&18&19&20\\21&22&23&24&25\end{bmatrix}\qquad B=\begin{bmatrix}3&0&16\\17&-6&9\\0&23&-4\\9&7&0\\4&13&11\end{bmatrix}$$

求乘积矩阵 $C=AB$。

具体要求如下：

(1) 矩阵 A 与 B 的元素在程序中直接用数组初始化进行赋值。

(2) 以矩阵形式输出乘积矩阵 C。

(一)【源码】

```
//
//  main.c
//  120504
//
//  Created by 李佩哲 on 2021/10/28.
//
```

```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
    int a[5][5]={
        {1,2,3,4,5},
        {6,7,8,9,10},
        {11,12,13,14,15},
        {16,17,18,19,20},
        {21,22,23,24,25}
    };
    int b[5][3]={
        {3,0,16},
        {17,-6,9},
        {0,23,-4},
        {9,7,0},
        {4,13,11}
    };
    unsigned long long c[5][3]={0};
    for (int i=0;i<5;i++){
        for(int j=0;j<3;j++){
            for(int k=0;k<5;k++){
                c[i][j]+=a[i][k]*b[k][j];
            }
        }
    }
    for (int i=0; i<5; i++) {
        for (int j=0; j<3; j++) {
            printf("%llu,",c[i][j]);
        }
        putchar('\n');
    }
    return 0;
}
```

(二)【运行结果】

```
93,150,77,
258,335,237,
423,520,397,
588,705,557,
753,890,717,
Program ended with exit code: 0
```

(三)【实验报告】

按行遍历A

然后按列遍历B

然后按元素遍历A，B

存到C里面