数组和指针

课后练习题

1. 数组基本练习

（1）提示用户从键盘输入 N（事先给定）个整数，并存入给定数组中。函数原型如下：

```
void input(int numArray[], int len);
```

其中，numArray 为用于保存用户输入的数组，len 为数组的长度。

参考程序：

```
void input(int numArray[], int len)
{
    int index;
    printf("请输入%d个整数，以空格隔开：\n", len);
    for (index = 0; index < len; index++)
        scanf("%d", &numArray[index]);
}
```

（2）在屏幕上打印给定的数组。函数原型如下：

```
void output(int numArray[], int len);
```

其中，numArray 为给定数组，len 为数组的长度。

参考程序：

```
void output(int numArray[], int len)
{
    int index;
    for (index = 0; index < len; index++)
        printf("%d ", numArray[index]);
    printf("\n");
}
```

（3）寻找给定数组中元素的最大值，并返回该最大值。函数原型如下：

```
int max(int numArray[], int len);
```

其中，numArray 为给定数组，len 为数组的长度。

参考程序：

```
int max(int numArray[], int len)
{
    int maxNum = numArray[0];
    int index;
    for (index = 1; index < len; index++)
        if (numArray[index] > maxNum)
            maxNum = numArray[index];
    return maxNum;
}
```

（4）求一个给定数组中元素的平均值，并返回该平均值。函数原型如下：

```
double average(int numArray[], int len);
```

其中，numArray 为给定数组，len 为数组的长度。

参考程序：

```
double average(int numArray[], int len)
{
    int sum = 0;
    int index;
    for (index = 0; index < len; index++)
        sum += numArray[index];
    return (double)sum/LEN;
}
```

（5）将给定数组反序，并返回该数组。函数原型如下：

```
int* reverse(int numArray[], int len);
```

其中，numArray 为给定数组，len 为数组的长度。

参考程序：

```
int* reverse(int numArray[], int len)
{
    int temp;
    int i1, i2;
    for (i1 = 0, i2 = len-1; i1 < i2; i1++, i2--) {
        temp = numArray[i1];
        numArray[i1] = numArray[i2];
        numArray[i2] = temp;
    }
    return numArray;
}
```

2. 指针基本练习

（1）写一个函数用于交换两个整数的值。函数原型如下：

```
void swap(int *a, int *b);
```

其中，a 和 b 为指向两个待交换的整数的指针。

参考程序：

```
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

（2）用指针遍历数组元素并求和。函数原型如下：

```
int sum(int *start, int *end);
```

其中，start 是指向数组首元素的指针，end 为指向数组末元素的下一个位置的指针。函数返回数组中所有元素之和。

参考程序：

```
int sum(int *start, int *end)
{
    int sum = 0;
    for ( ; start != end; start++)
        sum += *start;
    return sum;
}
```

（3）使用指针操作重做第 1 题（数组基本练习），参数列表均与（2）相同。

参考程序：

```
void input(int *start, int *end)
{
    printf("请输入%d个整数，以空格隔开：\n", end - start);
    for ( ; start != end; start++)
        scanf("%d", start);
}
```

```
void output(int *start, int *end)
{
    for ( ; start != end; start++)
        printf("%d ", *start);
    printf("\n");
}

int max(int *start, int *end)
{
    int maxNum = *start;
    for (start++ ; start != end; start++)
        if (*start > maxNum)
            maxNum = *start;
    return maxNum;
}

double average(int *start, int *end)
{
    return (double)sum(start, end)/(end - start);
}

int* reverse(int *start, int *end)
{
    int *array = start;
    for (end--; start < end; start++, end--)
        swap(start, end);
    return array;
}
```

3. 数学黑洞

任意一个 4 位自然数（各位数字均相同的除外，如 2222），将组成该数的各位数字重新排列，形成一个最大数和一个最小数，之后两数相减，其差值仍为一个自然数，重新进行上述计算，你会发现一个神秘的数。验证该结论：键盘输入一个 4 位数，输出每一步的差值，直到该神秘的数出现为止。

提示：

（1）所谓神秘数是指循环一定步数后，差值恒为某一常数；

（2）如果差值的位数小于 4，则自动补零，如差值为 738，则认为最大数为 8730，最小数为 0378。

参考程序：

```
#include <stdio.h>

int main(void)
{
    int num, prevNum = 0;
    int max, min;
    int numArray[4];
    int index;

    printf("请输入一个4位自然数（各位数字均相同的除外，如2222）：");
    scanf("%d",&num);

    while(num!=prevNum)//关键点1：设置判断循环结束的条件
    {
        prevNum = num;

        //关键点2：将读数入的4位数转化为数组
        for (index = 0; index < 4; index++) {
            numArray[index] = num%10;
            num /= 10;
        }

        //关键点3：排序
        for(int i=0;i<3;i++)
            for(int j=i+1;j<4;j++)
                if(numArray[j]>numArray[i])
                {
                    int temp = numArray[i];
                    numArray[i] = numArray[j];
                    numArray[j] = temp;
                }

        //关键点4：计算新的自然数，即用最大数减去最小数
        max = numArray[0]*1000+numArray[1]*100+numArray[2]*10+numArray[3];
        min = numArray[3]*1000+numArray[2]*100+numArray[1]*10+numArray[0];
        num = max-min;
        printf("%04d-%04d=%04d\n",max,min,num);//关键点5：打印的格式
    }

    printf("神秘数是%04d\n",num);

    return 0;
}
```

4. 猴子选大王

有 M 个猴子围成一圈，每个有一个编号，编号从 1 到 M。打算从中选出一个大王。经过协商，决定选大王的规则如下：从第一个开始，每隔 N 个，数到的猴子出圈，最后剩下来的就是大王。

要求：

（1）提示用户输入 M，N，编程计算哪一个编号的猴子成为大王。

（2）分别采用数组下标操作和指针操作实现上述问题的求解。

参考程序（采用数组下标操作，采用指针操作的程序大同小异）：

```
#include <stdio.h>
#define LEN 100

int monkeyKing(int monkeyArray[], int m, int n);

int main(void)
{
    int monkeyArray[LEN];
    int M, N;
    printf("请分别输入猴子总数M和报数间隔N（空格隔开）：");
    scanf("%d%d", &M, &N);
    printf("%d号猴子成为大王！\n", monkeyKing(monkeyArray, M, N));
    return 0;
}

int monkeyKing(int monkeyArray[], int m, int n)
{
    int i, j, k;
    for (i = 0; i < m; i++)
        monkeyArray[i] = 1;
    //模拟报数过程
    //i为数组下标变量，j为报数变量，k为出圈的猴子数
    i = 0; j = 0; k = 0;
    while (k != m-1) {
        j += monkeyArray[i];
        if (j == n) {
            monkeyArray[i] = 0;
            j = 0;
            k++;
        }
        if (++i == m)
            i = 0;
    }
```

```
    //寻找猴子大王，并返回其编号
    i = 0;
    while (monkeyArray[i] == 0)
        i++;
    return i + 1;
}
```

5. 数据的输入、排序、查找与插入

（1）写一个函数，提示用户输入一组整数（以空格隔开），将其按照从小到大排序并去掉重复数据，存入给定数组中。函数原型如下：

```
int inputAndSort(int numbers[], int len);
```

其中，numbers 为存储数据的数组，len 为该数组的长度。函数返回数组 numbers 中存储的整数的数量。

注：假设数组容量足够大。

（2）写一个函数，提示用户输入一个整数，并在给定的按从小到大顺序排好序的数组中查找该整数。若数组中存在该整数，则返回该整数在数组中的位置；否则，将该整数插入数组中（不改变数组的排序规律），返回插入后该整数在数组中的位置。不论属于何种情况，都需要输出必要的信息，告知用户查找和插入的情况。函数原型如下：

```
int searchAndInsert(int numbers[], int len, int* pcounter);
```

其中,numbers 为给定的按从小到大排好序的数组,len 为数组长度,pcounter 为指向数组计数器（统计已经存入数组的整数数量）的指针。

注：假设数组容量足够大。

（3）（选作）在第（1）和第（2）问中，我们假设数组容量足够大。然而实际中数组容量有限，当插入的数据过多时就会导致越界。这样的程序若出现在系统或者软件中很容易成为病毒攻击的对象。请改进上述两个函数，当出现越界操作是进行必要的处理，从而避免出现越界操作。

解答：

（1）此题思路较多，例如：1）先把用户输入的数全都存如数组，然后去除重复的数据，最后排序（3 个子函数）；2）存数的同时去除重复的数据，然后排序（两个子函数）；3）存数的同时去除重复的数据，并且始终保持从小到大存储。其中第 2 种方法不需要反复移动数组的元素，相对另外两种方法效率更高，故以下参

考程序采用第 2 种方法实现。同学们可以自己尝试采用第 1 和第 3 种方法实现，或者自己设计其他的求解思路。

参考程序：

```
int input(int numbers[], int len);
void sort(int numbers[], int counter);

int inputAndSort(int numbers[], int len)
{
    int counter = input(numbers, len);
    sort(numbers, counter);
    return counter;
}
int input(int numbers[], int len)
{
    int counter = 0;
    int temp, k;
    printf("请输入一组整数，空格隔开，回车结束：\n");
    do {
        scanf("%d", &temp);
        for (k = 0; k < counter; k++) //去除重复数据
            if (temp == numbers[k])
                break;
        if (k == counter)
            numbers[counter++] = temp;
    } while (getchar() != '\n');
    return counter;
}

void sort(int numbers[], int counter)
{
    int i, k, temp;
    for (i = 0; i < counter - 1; i++)
        for (k = i + 1; k < counter; k++)
            if (numbers[k] < numbers[i]) {
                temp = numbers[i];
                numbers[i] = numbers[k];
                numbers[k] = temp;
            }
}
```

（2）这一问包含两个关键环节，一是在数组中查找给定整数，二是将一个整数插入数组，可以用两个函数分别实现上述功能。查找的方法很多，最简单的方法就是按顺序将数组元素与待查找的整数比较。考虑到数组是排好序的，因此可以

不按顺序比较，而是直接将数组最中间的一个元素与待查找的整数比较，若找到则返回相应位置，否则也能确定该整数是在数组的前一半或者后一半中。这就是所谓的二分查找算法，可以用递归实现，也可以用循环实现。参考程序采用循环实现二分查找算法。

参考程序：

```
int binarySearch(int numbers[], int counter, int searchNum);
void insert(int numbers[], int* pcounter, int pos, int insertNum);

int searchAndInsert(int numbers[], int len, int* pcounter)
{
    int searchNum, pos;
    printf("请输入需要查找的整数：");
    scanf("%d", &searchNum);
    pos = binarySearch(numbers, *pcounter, searchNum);
    if (numbers[pos] == searchNum)
        printf("查找到整数 %d，存储在数组中下标为 %d 的位置！\n", searchNum, pos);
    else {
        insert(numbers, pcounter, pos, searchNum);
        printf("整数 %d 不在数组中，已插入数组中下标为%d的位置！\n", searchNum, pos);
    }
    return pos;
}

//在数组numbers中查找整数searchNum，若找到则返回searchNum在数组中所处的位置，
//否则返回searchNum应该插入的位置
int binarySearch(int numbers[], int counter, int searchNum)
{
    int mid, start = 0, end = counter - 1;
    while (start <= end) {
        mid = (start + end) / 2;
        if (numbers[mid] < searchNum)
            start = mid + 1;
        else if (numbers[mid] > searchNum)
            end = mid - 1;
        else
            return mid;
    }
    return start;
}
```

```
//将整数insertNum插入数组numbers中位置pos处，
//数组中pos位置以及以后的元素往后移动1个位置，计数器加1
void insert(int numbers[], int* pcounter, int pos, int insertNum)
{
    int index;
    for (index = *pcounter; index > pos; index--)
        numbers[index] = numbers[index-1];
    numbers[index] = insertNum;
    (*pcounter)++;
}
```

（3）（选作）这一问是一个开放性的问题，解决的方案当然很多。首先对本题的情况进行分析：本题中有两处可能导致数组越界，一处是第（1）问中存储用户输入的整数时，需要存储的整数数量超过数组的容量；另一处是第（2）问中数组已满时仍需要往输入插入数据的时候。此处提供以下几种解决方案供参考：

1）在可能越界的地方增加检查，在即将发生越界操作时予以阻止。例如，当用户输入的不重复整数个数超过数组容量时，舍弃多出来的数据，并提示用户数组已满，部分数据将被舍弃。这种方法以我们现有的知识即可实现，但是对用户来说不够友好。

2）在数组容量不足时，增大数组容量。这需要用到数组的动态创建和删除。

3）改用其他数据结构。

这一问为选作，不要求同学掌握，有兴趣的同学可以尝试使用第一种方法来完善自己的程序。

***6. 学分绩统计与排序**

事先给定学生的总数 N 和每个学生的姓名、课程的数量 M 以及每门课程对应的学分。提示用户从键盘输入 N 个学生的 M 门课程成绩，计算每个学生的学分绩，并按照学分绩从高到低的顺序输出每个学生的姓名、各门课程的成绩和学分绩。

要求与提示：

（1）定义以下数组用于存储相应的内容：

```
int credit[M];//存储 M 门课程的学分
char* names[N];//存储 N 个学生的姓名
double scores[N][M];//存储 N 个学生的各门课程成绩
double GPA[N];//存储 N 个学生的学分绩
```

（2）对学分绩进行排序时，需要对其他数组的顺序也需要进行相应的调整。（也可以专门定义一个数组来记录输出的顺序）

（3）根据模块化编程的思想和增量式开发方法，将上述问题分解成几个子问题，分别编写函数求解这些子问题，最终完成题目的要求。

分析：

本题有以下几个要点：

（1）**将需要解决的问题分解成几个子问题求解（模块化编程）**。对于本题，可以分为 4 个子问题：1）学生成绩的输入和存储；2）GPA 计算；3）按照学分绩从高到低排名；4）打印成绩。这 4 个子问题可以通过 4 个函数来实现，采用增量式开发方法，逐个编写并测试，最终在主函数中调用这些模块，完成题目要求。

（2）**数组作为函数参数**。本题涉及到整数数组、浮点数数组、字符串数组（指针数组）以及二维数组（数组的数组），这些数组在作为函数参数是如何使用是很重要的知识点。

（3）**排序**。此题在针对学分绩排序是，不仅仅要对存储学分绩的数组进行排序，还要对存储学生姓名的数组（字符串数组）和各门课成绩的数组（二维数组）进行排序。当然，也可以通过一个数组来记录排名，从而避免对字符串数组和二位数组进行排序。下面参考程序中采用的是第一种方法，同学可以自己尝试使用第二种方法。

（4）**学分绩计算方法**。这个大家应该会吧=^_^=

参考程序：

```
#include <stdio.h>

#define N 5
#define M 3

void inputScores(char* names[], double scores[][M], int n);
void calGPA(int credit[], double scores[][M], double GPA[], int n);
void sort(char* names[], double scores[][M], double GPA[], int n);
void printResult(char* names[], double scores[][M], double GPA[], int n);
```

```c
int main(void)
{
    int credit[M] = {2, 3, 4};  //存储M门课程的学分
    char* names[N] = {          //存储N个学生的姓名
        "小A",
        "小B",
        "小C",
        "小D",
        "小E",
    };
    double scores[N][M];        //存储N个学生的各门课程成绩
    double GPA[N];              //存储N个学生的学分绩

    inputScores(names, scores, N);          //输入各门课成绩
    calGPA(credit, scores, GPA, N);         //计算学分绩
    sort(names, scores, GPA, N);            //根据学分绩排序
    printResult(names, scores, GPA, N);     //打印结果

    return 0;
}

//输入各门课成绩
void inputScores(char* names[], double scores[][M], int n)
{
    int i, j;
    for (i = 0; i < n; i++) {
        printf("请依次输入%s %d门课的成绩（空格隔开）：", names[i], M);
        for (j = 0; j < M; j++)
            scanf("%lf", &scores[i][j]);
    }
}

//计算学分绩
void calGPA(int credit[], double scores[][M], double GPA[], int n)
{
    int i, j, totalCredit = 0;
    for (i = 0; i < M; i++)
        totalCredit += credit[i];

    for (i = 0; i < n; i++) {
        GPA[i] = 0.0;
        for (j = 0; j < M; j++)
            GPA[i] += credit[j] * scores[i][j];
        GPA[i] /= totalCredit;
    }
}
```

```
void swap1(double *a, double *b);
void swap2(char **a, char **b);
void swap3(double a[], double b[], int m);
//根据学分绩排序
void sort(char* names[], double scores[][M], double GPA[], int n)
{
    int i, j;
    for (i = 0; i < n-1; i++)
        for (j = i + 1; j < n; j++)
            if (GPA[j] > GPA[i]) {
                swap1(&GPA[i], &GPA[j]);
                swap2(&names[i], &names[j]);
                swap3(scores[i], scores[j], M);
            }
}
void swap1(double *a, double *b)
{
    double temp = *a;
    *a = *b;
    *b = temp;
}

//也可以通过字符串拷贝函数实现，
//此时函数原型为void swap2(char *a, char *b)
void swap2(char **a, char **b)
{
    char *temp = *a;
    *a = *b;
    *b = temp;
}

void swap3(double a[], double b[], int m)
{
    int k;
    for (k = 0; k < m; k++)
        swap1(&a[k], &b[k]);
}
```

```
//打印结果
void printResult(char* names[], double scores[][M], double GPA[], int n)
{
    int i, j;
    //打印标题行
    printf("排名  姓名  ");
    for (j = 0; j < M; j++) {
        printf("课程%d  ", j+1);
    }
    printf("GPA\n");
    //打印成绩
    for (i = 0; i < n; i++) {
        printf(" %02d   %s  ", i, names[i]);
        for (j = 0; j < M; j++)
            printf("%.2lf  ", scores[i][j]);
        printf("%.2lf\n", GPA[i]);
    }
}
```

运行结果示例：

```
请依次输入小A 3门课的成绩（空格隔开）：67 88.5 78
请依次输入小B 3门课的成绩（空格隔开）：99 98 97
请依次输入小C 3门课的成绩（空格隔开）：34 60 59
请依次输入小D 3门课的成绩（空格隔开）：85.5 88 90
请依次输入小E 3门课的成绩（空格隔开）：67 78 89
排名   姓名   课程1   课程2   课程3    GPA
 01    小B    99.00   98.00   97.00   97.78
 02    小D    85.50   88.00   90.00   88.33
 03    小E    67.00   78.00   89.00   80.44
 04    小A    67.00   88.50   78.00   79.06
 05    小C    34.00   60.00   59.00   53.78
```

题外话：独立完成此题对大家提高程序设计能力还是很有好处的。此外，通过增加学号、班级、课程名称等信息，并将重要信息的输入和输出改用文件读写实现，这个程序还是很有实用价值的！=^_^=