Assignment_No.3

统计出现最多的数字

> 【问题描述】输入一组无序的整数，编程输出其中出现次数最多的整数及其出现次数。
> 【输入形式】读入整数的个数（大于等于1，小于等于100），然后在下一行输入这些整数。
> 【输出形式】输出出现次数最多的整数及其出现的次数，两者以一个空格分隔；若出现次数最多的整数有多个，则按照整数升序分行输出。
> 【样例输入】
> 10
> 0 -50 0 632 5813 -50 9 -50 0 632
> 【样例输出】
> -50 3
> 0 3
> 【样例说明】输入了10个整数，其中出现次数最多的是-50和0，都出现了3次。输出结束有换行。

解析

- 我们其实最简单的做法就是使用一个结构体，储存数字和它对应的出现次数
- 统计完毕后我们将结构体进行两次排序就行。
- 更建议大家阅读示例代码2，更加的简单易懂

示例代码1

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

typedef struct {
    int number;
    int count;
} Frequency;

int compare_count(const void* a, const void* b) {
    return ((Frequency*)b)->count - ((Frequency*)a)->count;
}

int compare_number(const void* a, const void* b) {
    return ((Frequency*)a)->number - ((Frequency*)b)->number;
```

```c
}

void findMostFrequent(int arr[], int n) {
    Frequency frequencies[n];
    int count = 0;

    for (int i = 0; i < n; i++) {
        int j;
        for (j = 0; j < count; j++) {
            if (frequencies[j].number == arr[i]) {
                frequencies[j].count++;
                break;
            }
        }
        if (j == count) {
            frequencies[count].number = arr[i];
            frequencies[count].count = 1;
            count++;
        }
    }

    qsort(frequencies, count, sizeof(Frequency), compare_count);
    int n_max = 1;
    for (int i = 1; i < count; i++){
        if (frequencies[i].count == frequencies[0].count)
            n_max++;
        else
            break;
    }

    qsort(frequencies, n_max, sizeof(Frequency), compare_number);

    printf("%d %d\n", frequencies[0].number, frequencies[0].count);

    for (int i = 1; i < count; i++) {
        if (frequencies[i].count == frequencies[0].count) {
            printf("%d %d\n", frequencies[i].number, frequencies[i].count);
        } else {
            break;
        }
    }
}

int main() {
    int n;
    scanf("%d", &n);

    int arr[n];
```

```
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    findMostFrequent(arr, n);

    return 0;
}
```

示例代码2

```
#include<stdio.h>
#define N 101

int main()
{
    int m,i,j,temp,count,max_count=0;
    static int arr[N];
    scanf("%d",&m);
    for(i=0;i<m;i++)
        scanf("%d",&arr[i]);
    for(i=1;i<m;i++){
        temp = arr[i];
        for(j=i;j>0&&arr[j-1]>temp;j--)
            arr[j] = arr[j-1];
        arr[j] = temp;
    }
    count = 1;  //计数
    for(i=0;i<m;i++){   //此处循环找出重复的最大个数放入max_count变量中
        if(arr[i]!=arr[i+1]){
            if(count>max_count)
                max_count = count;
            count = 1;
        }
        else
            count++;
    }
    count = 1;
    for(i=0;i<m;i++){
        if(arr[i]!=arr[i+1]){
            if(count == max_count)
                printf("%d %d\n",arr[i],count);
            count = 1;
        }
        else
            count++;
    }
    return 0;
```

```
}
```

有序的交集输出

【问题描述】读入两组整数（每组不超过20个整数，并且同一组中的整数各不相同），编程求两组整数的交集，即在两组整数中都出现的整数，并按从大到小的顺序输出。若交集为空，则什么都不输出。
【输入形式】先输入第一组整数的个数，然后在下一行输入第一组整数；然后再以同样的方式输入第二组整数。
【输出形式】按从大到小顺序输出两组整数的交集（每个整数占6位，即按%6d格式输出每个整数）。
【样例输入】
`8`
`5 -105 0 4 32 -87 9 -60`
`7`
`5 2 87 10 -105 0 32`
【样例输出】`32     5     0  -105`
【样例说明】
第一组整数有8个，第二组整数有7个，在这两组整数中都出现的整数有4个，按从大到小顺序排序后输出的结果为：32 5 0 -105

解析

- 这道题在上一次的作业中出现过，在这里只是需要做一个排序而已

示例代码1

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct {
    int number;
    int count;
} Frequency;

int compare_count(const void* a, const void* b) {
    return ((Frequency*)b)->count - ((Frequency*)a)->count;
}

int compare_number(const void* a, const void* b) {
    return ((Frequency*)a)->number - ((Frequency*)b)->number;
}
```

```c

void findMostFrequent(int arr[], int n) {
    Frequency frequencies[n];
    int count = 0;
    for (int i = 0; i < n; i++) {
        int j;
        for (j = 0; j < count; j++) {
            if (frequencies[j].number == arr[i]) {
                frequencies[j].count++;
                break;
            }
        }
        if (j == count) {
            frequencies[count].number = arr[i];
            frequencies[count].count = 1;
            count++;
        }
    }
    qsort(frequencies, count, sizeof(Frequency), compare_count);
    int n_max = 1;
    for (int i = 1; i < count; i++){
        if (frequencies[i].count == frequencies[0].count)
            n_max++;
        else
            break;
    }
    qsort(frequencies, n_max, sizeof(Frequency), compare_number);
    printf("%d %d\n", frequencies[0].number, frequencies[0].count);

    for (int i = 1; i < count; i++) {
        if (frequencies[i].count == frequencies[0].count) {
            printf("%d %d\n", frequencies[i].number, frequencies[i].count);
        }
        else
            break;
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    findMostFrequent(arr, n);
    return 0;
}
```

示例代码2

```c
#include<stdio.h>
#define N 100

int main()
{

    int m,n,i,j,temp,k=0;
    static int arr1[N],arr2[N],arr3[N];
    scanf("%d",&m);
    for(i=0;i<m;i++)
        scanf("%d",&arr1[i]);
    scanf("%d",&n);
    for(i=0;i<n;i++)
        scanf("%d",&arr2[i]);
    for(i=0;i<m;i++)
        for(j=0;j<n;j++)
            if(arr1[i] == arr2[j]){
                arr3[k++] = arr1[i];
                break;
            }
    for(i=1;i<k;i++){
        temp = arr3[i];
        for(j=i;j>0&&arr3[j-1]<temp;j--)
            arr3[j] = arr3[j-1];
        arr3[j] = temp;
    }
    for(i=0;i<k;i++)
        printf("%6d",arr3[i]);
    return 0;
}
```

字符串分隔

【问题描述】输入两个字符串str和cut。cut由若干个字符构成，其中每个字符均可作为一个分隔字符对str进行分隔。注意：str和cut中均可以包含空格。如果cut中含有空格，则空格也作为str的分隔字符。cut中字符不能用减号（系统问题）。
【输入形式】分两行输入两个字符串str和cut。
【输出形式】分行输出str被分隔后的各字符串。
【样例输入】（其中“□”代表一个空格）
`jfi,dpf.,jfpe&df&jfpf/□□jfoef$djfo□,pe`
`,.□/&$`
【样例输出】

> jfi
> dpf
> jfpe
> df
> jfpf
> jfoef
> djfo
> pe
> 【样例说明】输入字符串 str ="jfi,dpf.,jfpe&df&jfpf/□□jfoef$djfo□,pe", cut =",.□/&$" ，cut中的每个字符（包括空格）均可作为str的分隔符。输出结束无换行符。

解析

- 使用库函数 strtok 就好了，很简单

示例代码1

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

void splitString(char str[], char cut[]) {
    char *token = strtok(str, cut);
    while (token != NULL) {
        printf("%s\n", token);
        token = strtok(NULL, cut);
    }
}

int main() {
    char str[1000];
    char cut[100];

    fgets(str, sizeof(str), stdin);
    fgets(cut, sizeof(cut), stdin);

    str[strcspn(str, "\n")] = '\0';
    cut[strcspn(cut, "\n")] = '\0';

    splitString(str, cut);

    return 0;
}
```

示例代码2

```c
#include<stdio.h>
#include<string.h>
int main()
{
    int i,j,n,m,to_return=0;
    char str[1025],cut[100];
    gets(str);
    gets(cut);
    n = strlen(str);
    m = strlen(cut);
    for(i=0;i<n;i++){
        for(j=0;j<m;j++){
            if(str[i]==cut[j]){
                if(to_return == 0)
                    to_return = 1;
                break;
            }
        }
        if(j>=m)
            if(to_return == 1){
                printf("\n%c",str[i]);
                to_return = 0;
            }
            else
                printf("%c",str[i]);
    }
    return 0;
}
```

计算星期

【问题描述】任意输入一个日期，求这一天是星期几。
【输入形式】从键盘输入一行字符串“Y-M-D”，表示一个有效的公历日期。其中Y为年（范围为1980—3000年），M为月，D为天，都不带有前缀0。提示：输入语句格式为 `scanf("%d-%d-%d",&year,&month,&day);`
【输出形式】输出只有一行，是代表星期的字符串。对于星期一至星期日，分别输出Monday、Tuesday、Wednesday、Thursday、Friday、Saturday、Sunday。输出结束不换行。判断闰年的算法是：年份能被4整除并且不能被100整除，或者能被400整除。
【样例输入】 `2004-1-6`
【样例输出】 `Tuesday`

解析

- 其实就是计算天数然后对7去余。
- 选定一个基准点去计算就好了。

示例代码

```c
#include <stdio.h>

int isLeapYear(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}

int dayOfWeek(int year, int month, int day) {
    int daysInMonth[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    int totalDays = 0;

    for (int i = 1980; i < year; i++)
        totalDays += isLeapYear(i) ? 366 : 365;

    for (int i = 1; i < month; i++) {
        totalDays += daysInMonth[i];
        if (i == 2 && isLeapYear(year)) {
            totalDays++;
        }
    }

    totalDays += day;

    return (totalDays+1) % 7;
}

int main() {
    int year, month, day;
    scanf("%d-%d-%d", &year, &month, &day);

    int dayIndex = dayOfWeek(year, month, day);
    char* daynames[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
    printf("%s\n", daynames[dayIndex]);
    return 0;
}
```

赌王密码的计算

【问题描述】赌王喜欢“A”，密码由6行6列扑克牌中每行“A”的位置数字组合而成。扑克牌点数由1～9,J,Q,K,A组成，每行的扑克牌中最多只能出现一次“A”；也可能没有“A”，则密码中对应的位置数字是0。
【输入形式】6行6列字符
【输出形式】6个位置数字组成的密码，输出后不换行。
【样例输入】
789AJK
QKA358
123456
456789
AJQK78
56789A
【样例输出】
`430016`
【样例说明】第1行中“A”出现的位置是4，第2行中“A”出现的位置是3，第3行和第4行中没有出现“A”，则对应的位置数字是0，第5行中“A”出现的位置是1，第6行中“A”出现的位置是6，所以组成的密码是430016。

解析

- 我们一次读一行就好，然后去找这一行中是否存在A或者A在第几个，把找到的数据扔到密码数组里面就行了。

示例代码

```c
#include <stdio.h>

int main() {
    int count;
    char a,password[7]={"000000"};

    for (int i = 0; i < 6; i++){
        for (count = 1; count <= 6; count++){
            scanf("%c", &a);
            if(a == 'A')
                password[i] = count + '0';
        }
        getchar();
    }
    printf("%s\n", password);

}
```

营业额占比

【问题描述】从键盘输入学校附近某烧烤店某年每月的营业额，然后计算每月的营业额在年营业额中所占的百分比（四舍五入为整数，且不会超过全年的70%），并以样例输出所示的水平直方图形式打印出来。
【输入形式】输入12个月的营业额（浮点数），中间用一个空格分隔。
【输出形式】水平直方图形式输出。
【样例输入】 `10 20.7 20.3 40 60.6 80 130 120 110 65 35 15`
【样例输出】

```
1( 1%) #
2( 3%) ###
3( 3%) ###
4( 6%) ######
5( 9%) #########
6(11%) ###########
7(18%) ##################
8(17%) #################
9(16%) ################
10( 9%) #########
11( 5%) #####
12( 2%) ##
```

【样例说明】第一部分为月份，占2列；第二部分为百分比，占5列；第三部分从第9列开始，为用#号表示的比例，1个#号表示1%。输出结束后换行。

解析

- 就是简单的计算和以及占比就ok

示例代码

```c
#include <stdio.h>

int main() {
    double monthlyTurnover[12];
    double annualTurnover = 0.0;

    // 输入12个月的营业额
    for (int i = 0; i < 12; i++) {
        scanf("%lf", &monthlyTurnover[i]);
        annualTurnover += monthlyTurnover[i];
    }

```

```
    // 计算百分比并打印水平直方图
    for (int month = 1; month <= 12; month++) {
        int percent = (int)((monthlyTurnover[month - 1] / annualTurnover) * 100.0 + 0.5); // 四舍五入
        printf("%2d(%2d%%) ", month, percent);

        for (int i = 0; i < percent; i++)
            printf("#");
        printf("\n");
    }

    return 0;
}
```

读数字（傻瓜版）

【问题描述】
从键盘输入不超过10行10列的整型二维数组中的元素，求出各奇数行（下标为0，2，4，6…的行，即第1，3，5，7…行）之和，并把和的每位数字转成相应的拼音（数字0～9的拼音分别为：ling，yi，er，san，si，wu，liu，qi，ba，jiu）输出，输出格式参照样例输出的格式。
【样例输入1】

```
5 6
56  78  36  4   50  80
19  44  95  72  -8  60
85  67  -3  32  12  35
29  21  47  88  28  -9
7   66  53  40  20  15
```

【样例输出1】

```
304:  san ling   si
228:   er   er   ba
201:   er ling   yi
```

示例代码

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

void intToPinyin(int sum) {
    char* pinyin[] = {"ling", "yi", "er", "san", "si", "wu", "liu", "qi", "ba", "jiu"};
```

```
    char* output[11] = {"ling","ling","ling","ling","ling","ling","ling","ling","ling","ling","ling"};
    int z = 0, n = sum;

    while (n > 0) {
        int digit = n % 10;
        output[z++] = pinyin[digit];
        n /= 10;
    }

    printf("%d:",sum);
    z = (z == 0 ? 1 : z );
    for (int i = z - 1; i >= 0; i--)
        printf("%5s", output[i]);
    printf("\n");
}

int main() {
    int m, n, i;
    scanf("%d%d", &m, &n);

    int array[11][11];
    int rowSums[11] = {0};
    int sum = 0;

    for (i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &array[i][j]);
            if (i % 2 == 0)
                rowSums[i] += array[i][j];
        }
    }

    for (int j = 0; j <= i; j +=2)
        intToPinyin(rowSums[j]);
    return 0;
}
```

统计字符的个数

【问题描述】输入一行含空格在内的字符，分别统计其中每个小写字母的个数，并按字母顺序输出个数不为零的小写字母及其对应的个数，每对占1行；若无小写字母则输出“None”。
【样例输入1】 `6a1b2c3 D4abcdxyz`
【样例输出1】
a:2

```
b:2
c:2
d:1
x:1
y:1
z:1
```

【样例说明1】输入字符串中，小写字母a，b，c各出现2次，d，x，y，z各出现1次，其他小写字母没出现就不输出。

【样例输入2】 `ABC123ABC DEF SHU.`

【样例输出2】 `None`

【样例说明2】输入的字符串中无小写字母。输出结束要换行。

解析

- 灵活使用标准库函数 `ctype.h` 中的函数就行

示例代码

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

int main() {
    char input[1000];
    int lowercaseCount[26] = {0};  // 用于统计每个小写字母出现的次数

    // 读取输入字符，直到输入结束
    gets(input);

    for (int i = 0; input[i] != '\0'; i++) {
        char c = input[i];
        if (islower(c))
            // 如果是小写字母，增加对应字母的计数
            lowercaseCount[c - 'a']++;
    }

    int found = 0;  // 用于标记是否找到小写字母

    for (int i = 0; i < 26; i++) {
        if (lowercaseCount[i] > 0) {
            printf("%c:%d\n", 'a' + i, lowercaseCount[i]);
            found = 1;
        }
    }

    if (!found)
        printf("None\n");
```

```
29
30      return 0;
31  }
```

校园歌手大赛，不评委大赛

> 【问题描述】校园歌手大奖赛中，有5个评委为参赛的选手打分，分数取值1～10，且各不相同。选手最后得分为：去掉一个最高分和一个最低分后其余3个分数的平均值。同时对评委评分进行裁判，即在5个评委中找出最公平（即评分最接近平均分）的评委。
> （1）输入评委编号（int型一维数组）及相应的打分（int型一维数组）；
> （2）求解并输出平均分（double型变量，保留两位小数）；
> （3）求解并输出最公平的评委（假设只评出一位最公平的评委，如果几位评委分数一致，输出最先输入的评委编号）编号。

示例代码

```c
#include <stdio.h>

int main() {
    int judges[5];  // 评委编号
    int scores[5];  // 评分
    double average = 0.0;  // 平均分
    int mostFairJudge = -1;  // 最公平评委的编号
    double closestDiff = 10.0;  // 初始化为最大可能差距

    for (int i = 0; i < 5; i++) {
        scanf("%d %d", &judges[i], &scores[i]);
        average += scores[i];

        if (scores[i] > scores[mostFairJudge])
            mostFairJudge = i;
    }

    // 找到最高分和最低分的索引
    int maxIndex = 0;
    int minIndex = 0;
    for (int i = 1; i < 5; i++) {
        if (scores[i] > scores[maxIndex])
            maxIndex = i;
        if (scores[i] < scores[minIndex])
            minIndex = i;
    }

    // 计算平均分
```

```
    average = (average - scores[maxIndex] - scores[minIndex]) / 3.0;

    // 查找最公平评委
    for (int i = 0; i < 5; i++) {
        double diff = scores[i] - average;
        if (diff < 0)
            diff = -diff;  // 取绝对值
        if (diff < closestDiff) {
            closestDiff = diff;
            mostFairJudge = i;
        }
    }

    // 输出平均分和最公平评委
    printf("%.2lf\n", average);
    printf("%d\n", judges[mostFairJudge]);

    return 0;
}
```

加密字符

【问题描述】
在情报传递过程中，为了防止情报被截获，往往需要用一定的方式对情报进行加密。简单的加密算法虽然不足以完全避免情报被破译，但仍然能防止情报被轻易识别。我们给出一种加密算法，对给定的一个明文字符串（括号中是一个“原文 -> 密文”的例子）：
（1）明文字符串中所有的字母都按字母表顺序被循环左移了三个位置（`deac -> abxz`），其他非字母的字符不变；
（2）逆序存储（`abxz ->zxba`）。
编写程序，输入明文字符串（含空格），输出加密后的密文字符串。输出结束要换行。
【输入形式】 输入一行，包含一个字符串，其长度小于80个字符。
【输出形式】输出加密字符串。
【样例输入】 `Hello! Ace 30`
【样例输出】 `03 bzX !liibE`

示例代码

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

int main() {
```

```c
    char input[80];
    fgets(input, sizeof(input), stdin);

    for (int i = 0; input[i] != '\0'; i++) {
        if (isalpha(input[i])) {
            // 如果是字母，按字母表顺序左移三位
            char shift = (islower(input[i])) ? 'a' : 'A';
            input[i] = (input[i] - shift - 3 + 26) % 26 +
shift;
        }
    }

    int len = strlen(input);
    for (int i = 0; i < len / 2; i++) {
        // 逆序存储
        char temp = input[i];
        input[i] = input[len - 1 - i];
        input[len - 1 - i] = temp;
    }

    printf("%s\n", input);

    return 0;
}
```