第三次上机题解

关于读入

这次的题目中出现了大量没有限定组数的数据，那么，对于一组未指定组数的数据，我们怎么输入呢？当然是用循环了，也就是`while(scanf())`。既然有了循环，肯定要有结束条件。这时候我们就需要了解一下scanf的返回值。

scanf的返回值分为三种情况

1. 正整数，表示正确输入参数的个数。

```
例如执行 scanf("%d %d", &a, &b);
如果用户输入"3 4"，可以正确输入，返回2（正确输入了两个变量）；
如果用户输入"3,4"，可以正确输入a，无法输入b，返回1（正确输入了一个变量）。
```

2. 0，表示用户的输入不匹配，无法正确输入任何值。

```
如上例，用户如果输入",3 4"，返回0。
```

3. EOF，这是在stdio.h里面定义的常量（通常值为-1），表示输入流已经结束。在

```
Windows下，用户按下CTRL+Z（会看到一个^Z字符）再按下回车（可能需要重复2次），就表示输入结束；
Linux/Unix下使用CTRL+D表示输入结束
```

我们oj上是将文件重定向到标准输入输出的，所以我们应该读到`EOF`的时候跳出，所以多组数据读入的时候就是`while(scanf()!=-1){}`

但是往往为了简便，我们会写成`while(~scanf()){}`，这实际上是一个trick，~-1 = 0，表示的含义和上述的相同。

A 高低位交换

本题考察位运算。容易发现n >> 16可获得正整数n的“低位”，再与n << 16做或运算，就可将“高位”和“低位”交换。

```c
#include <stdio.h>
int main()
{
    unsigned int n;
    scanf("%u", &n);
    printf("%u\n", (n >> 16) | (n << 16));
    return 0;
}
```

当然我们也可以将原数转换成二进制，进行交换后再转换成十进制输出，稍微麻烦一些。

```c
#include <stdio.h>
int main()
```

```
{
    unsigned n;
    scanf("%u", &n);
    int i = 31;
    int code[32];
    while (i >= 0) {
        if(n % 2 == 0) {
            code[i] = 0;
        } else {
            code[i] = 1;
        }
        n /= 2;
        i -= 1;
    }
    n = 0;
    for (i = 16; i <= 31; i += 1) {
        n = n * 2 + code[i];
    }
    for (i = 0; i <= 15; i += 1) {
        n = n * 2 + code[i];
    }
    printf("%u\n",n);
    return 0;
}
```

B 求三角形的面积

已知三角形三个点的坐标，由向量法或者海伦公式可求三角形面积。向量法和海伦公式提示里都有。

海伦公式：

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

int main()
{
    double p1_x, p1_y;
    double p2_x, p2_y;
    double p3_x, p3_y;

    double e1, e2, e3,p;

    scanf("%lf%lf", &p1_x, &p1_y);
    scanf("%lf%lf", &p2_x, &p2_y);
    scanf("%lf%lf", &p3_x, &p3_y);

    e1 = sqrt((p1_x - p2_x)*(p1_x - p2_x) + (p1_y - p2_y)*(p1_y - p2_y));
    e2 = sqrt((p2_x - p3_x)*(p2_x - p3_x) + (p2_y - p3_y)*(p2_y - p3_y));
    e3 = sqrt((p1_x - p3_x)*(p1_x - p3_x) + (p1_y - p3_y)*(p1_y - p3_y));

    p = (e1 + e2 + e3) / 2.;

```

```
    printf("%.4f", sqrt(p*(p - e1)*(p - e2)*(p - e3)));
}
```

向量法：

```
#include <stdio.h>
#include <math.h>
int main()
{
    double x1, x2, x3, y1, y2, y3;
    scanf("%lf%lf%lf%lf%lf%lf", &x1, &y1, &x2, &y2, &x3, &y3);
    double x12 = x2 - x1, y12 = y2 - y1, x13 = x3 - x1, y13 = y3 - y1;
    printf("%.4f\n", fabs(x12 * y13 - x13 * y12) / 2);
    return 0;
}
```

C 好懂的题

题意是给定两个时刻h1:m1, h2:m2，求这个两个时刻的中点时刻。由于题目保证了m1 % 2 = m2 % 2，故直接将两个时刻换算为分钟后求和并除以2，再换算成时和分。

```
#include <stdio.h>
int main()
{
    int h1, m1, h2, m2;
    scanf("%d:%d", &h1, &m1);
    scanf("%d:%d", &h2, &m2);
    int t = (h1 * 60 + m1 + h2 * 60 + m2) / 2;
    printf("%02d:%02d\n", t / 60, t % 60);
    return 0;
}
```

D 猴子吃桃

设第n天有a[n]个桃子，晚上吃掉一半又多一个，那么第n + 1天有a[n] / 2 - 1个桃子。 所以a[n + 1] = a[n] / 2 - 1，即a[n] = 2(a[n + 1] + 1)。第m天有1个桃子，即a[m] = 1，可递推求得a[1]，即最初的桃子数。

```
#include <stdio.h>
int main()
{
    int n;
    scanf("%d", &n);
    while (n--) {
        int m;
        scanf("%d", &m);
        int ans = 1;
        int i;
        for (i = 2; i <= m; i += 1) {
            ans = 2 * (ans + 1);
        }
```

```
        printf("%d\n", ans);
    }
    return 0;
}
```

E lyz学姐的体重

题意

找一个数x，使得

$x \mod 3 = a$

$x \mod 5 = b$

$x \mod 7 = c$

思路

使用 循环语句，遍历所有数，不断判断当前值是否满足条件即可。

```
#include <stdio.h>

int main()
{
    int a,b,c;
    while(~scanf("%d%d%d", &a, &b, &c)){
        int i = 1;
        while(1)
        {
            if(i % 3 == a && i % 5 == b && i % 7 == c)
            {
                printf("%d\n", i);
                break;
            }
            i++;
        }
    }
    return 0;
}
```

思路2（拓展）

读完题后，会觉得这个东西可以直接算出来。

事实上，的确有一个定理，可以解决这个问题：

中国剩余定理

(这个定理涉及了一些数论的知识，有能力的同学可以尝试理解一下)

```c
#include <stdio.h>

int main() {
    int a, b, c;
    while (~scanf("%d%d%d", &a, &b, &c)) {
        int res = (70 * a + 21 * b + 15 * c) % 105;
        printf("%d\n", res ? res : 105);
    }
    return 0;
}
```

F 难懂的题

题意

求一个1-n的排列内，有多少个区间的最大值为n。

思路

由于因为n只有一个，所以就是求有多少个区间包含n。

同时，因为区间是连续的，所以满足条件的区间的必定 **一个端点在n左边（或n上），另一个端点在n右边（或n上）。**

假设我们知道了n左边有l个，右边有r个，则答案为$(l+1)*(r+1)$，显然，答案只与n的位置有关。

因此，我们只需要求出n是第几个数就可以了。

```c
#include <stdio.h>

int main(){
    int n,i;
    int a;
    int plc = 0;
    scanf("%d",&n);
    i = 1;
    while (i <= n) {    // for (i=1; i<=n; i++)
        scanf("%d", &a);
        if (a == n) {
            plc = i;    // 记录下n的位置
        }
        i++;
    }
    printf("%d\n", plc * (n - plc +1) );
    return 0;
}
```

G Ausar的草稿框

题意

n个筐，多次操作

每次操作可以：

- 某个筐内的数量加上某个数
- 把所有筐里的东西的加起来放到某个筐中。

思路

读完题目，我们可以简单的将题目转换成“比较编程的描述”

n个变量，多次操作

- 某个变量加上某个数
- 把所有变量的值加起来，和赋值给个某个变量，同时其他变量赋值为0

显然，我们需要使用**第i个变量**，这就是数组的使用场景。

这样，就是一个很简单的“模拟”。

```c
#include <stdio.h>
int main()
{
    int n, m, i;
    int a[105] = {0};
    scanf("%d%d", &n, &m);
    while (m--) {
        int o, x, k;
        scanf("%d%d%d", &o, &x, &k);
        if (o == 1) {
            a[x] += k;
        } else {
            for (i = 1; i < n; i += 1) {
                a[n] += a[i];
                a[i] = 0;
            }
        }
    }
    for (i = 1; i <= n; i += 1) {
        printf("%d ", a[i]);
    }
    return 0;
}
```

H 摸鱼村村长

思路

当你读完这道题目时，可能可以想到一个较为朴素的做法： 记录下每个人得了多少票，判断得票数最多的人是否票数超过$\frac{cnt}{2}$,如果超过，则该人为村长。

那么，该如何记录每个人得到了多少票呢?

我们可以开一个数组，假设为`a[N]`，令`a[i]`表示第i个人的得票数，这样如果第`i`个人得票，则令`a[i]++`。

这样，如果我们要得到第i个人的得票数，`a[i]`就是我们想要的。

当然，由于这道题的输入的数字范围可能过大，因此这种方法不适用于此题。(显然，N需要大于序号最大的那个人的序号)

但是，这种朴素方法在程序中用于**记录每人得票数**时用到的思想十分重要，建议大家认真理解这种方法。

```c
#include<stdio.h>

#define N 10000
int votes[N];

int main()
{
    int x, y, win = 0, cnt = 0;
    while(scanf("%d", &x) != EOF)
    {
        cnt++;
        if(x < N)
        {
            if(++votes[x] > win) //用于记录最大的票数
            {
                win = votes[x];
                y = x;
            }
        }
        else
        {
            printf("Invalid Vote!\n");
        }
    }

    if(win >= cnt / 2)
    {
        printf("We have a winner: %d\n", y);
    }
    else
    {
        printf("No body win!\n");
    }
    return 0;
}
```

```
36
```

那么，这道题出题时，出题人想考察的做法是什么呢：

本题解法的核心思想就是删除，每次删除一对不同的数字，直到最后都剩下相同的，那么最后剩下的就肯定是出现次数大于1/2的。

在具体实现中呢，我们使用两个变量，分别记录当前出现最多的数字`a`和这个数字出现的次数`b`。

在遍历所有数字的过程中，如果遇到和`a`相同的数字，那么次数`b`加1，否则`b`减1（表示抵消了一对不同得数字）；如果`b`等于0，那么`a`已经不再是出现次数最多的数字了(不同的一对数字抵消)，然后指定为下一个数字，其对应次数为1。

代码

```
#include<stdio.h>

int main()
{
    int a, b, c;
    a = b = 0;
    while (~scanf("%d", &c))
    {
        if (b == 0)
        {
            a = c;
            b++;
        }
        else if (a == c)
        {
            b++;
        }
        else
        {
            b--;
        }
    }
    printf("%d", a);
    return 0;
}
```

思路二

将所有读入投票信息使用数组储存下来，然后我们对于每个a[i]，扫描一遍数组，统计数组里有多少个元素与a[i]相等，统计完毕后，如果与a[i]相等的元素个数>= n / 2，那么a[i]就是我们所要求的。 但本题没有给出数字个数的范围，用这种方法通过只是侥幸，因为数据比较小，并且题目没有刻意限制空间大小。当数字个数很多的时候，或题目对空间的要求很苛刻，数组就不能存下这么多数，本方法就失效了。

```
```

```c
#include <stdio.h>
int a[2000005];
int n;

int main()
{
    int x;
    while (scanf("%d", &x) != EOF) {
        a[n] = x;
        n += 1;
    }
    int i, j;
    for (i = 0; i < n; i += 1) {
        int cnt = 0;
        for (j = 0; j < n; j += 1) {
            if (a[i] == a[j]) {
                cnt += 1;
            }
        }
        if (cnt >= n / 2) {
            printf("%d\n", a[i]);
            break;
        }
    }
    return 0;
}
```

思路三

这个方法用到了部分超纲知识，有需要得同学可以阅读

那么，我们是否有办法得到每个人究竟得了多少票呢?

如果有同学了解过排序算法，可以知道有*复杂度*为$O(n \log_2 n)$的排序算法，通过这个，我们就可以的得到所有人的了多少票了。

将所有读入投票信息使用数组储存下来，然后进行排序，使得相同序号的票相邻，这样，我们扫描一遍这个数组，当发现相邻两个数不一样时，记录下来位置。同志，这说明目前位置到上一次记录的地方这一段是一样的，这样就可以得到每一段的长度了（即每个人被投票了多少次）。

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

int a[2000000];

int comp_int(const void *fi, const void *se)
{
    return *(int *)fi-*(int *)se;
}

int main()
```

```
{
    int cnt = 1;
    while (scanf("%d", &a[cnt]) != EOF)
    {
        cnt++;
    }
    cnt--;
    qsort(a+1, cnt, sizeof(int), comp_int); //调用了c库自带的快速排序函数
    int la = 1;
    int ans;
    for (int i = 1; i <= cnt; i++)
    {
        if(i == cnt || a[i] != a[i+1])
        {
            if(i - la + 1 > cnt / 2)
            {
                ans = a[i];
                break;
            }
        }
    }
    printf("%d", ans);
    return 0;
}
```