



# C++编程



# 目录

- 程序头文件
- 输入输出
- 赋值
- 数据类型与范围
- 运算符与表达式
- 浮点数
- 分支
- 循环
- 迭代
- 多重循环
- 穷举
- 函数
- 递归
- 不定长输入
- 数组
- 标记思想

# 目录

- 无穷大
- 排序和去重
- 中位数
- 埃氏筛
- 二分查找
- 在线和离线
- 二维数组
- 字符数组
- 字符串
- 高精度运算
- 结构体
- 指针

# 数组的应用

- 输入n个数，按输入顺序逆序输出这n个数

```
5
1 2 3 4 5
5 4 3 2 1
-----
Process exited after 7.92 seconds with return value 0
请按任意键继续. . .
```

```
#include<bits/stdc++.h>
using namespace std;
int a[1010];
int main()
{
    int i, n;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

    for (i = n - 1; i >= 0; i--)
        printf("%d ", a[i]);
    return 0;
}
```

# 数组的应用

- 输入n个数，输出其中低于它们算术平均值的数

```
5
21 16 32 9 25
16 9
-----
Process exited after 19.54 seconds with return value 0
请按任意键继续. . .
```

```
#include<bits/stdc++.h>
using namespace std;
int a[1010];
int main()
{
    int i, n, sum = 0;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
        sum += a[i]; // 读入过程中同步统计和
    }

    for (i = 0; i < n; i++)
        if (a[i] < sum / n) printf("%d ", a[i]);
    return 0;
}
```



# 陶陶摘苹果

- 陶陶家的院子里有一棵苹果树，苹果成熟的时候，陶陶就会跑去摘苹果。
- 陶陶有个高30的板凳，当她不能直接用手摘到苹果时，就会踩到板凳上试试。
- 现在已知十个苹果到地面的高度，以及陶陶把手伸直的时候能够达到的最大高度，请帮陶陶算一下她能摘到其中多少个苹果。

Sample input	Sample output
100 200 150 140 129 134 167 198 200 111 //每个苹果的高度 110 //陶陶的身高	5

# 分析



- 把苹果高度逐个存起来，然后从头到尾扫一遍就可以了
- 如果先读入陶陶的身高后读入苹果的高度，那么甚至不需要数组存起来：直接每读入一个苹果的高度就判断一次即可

# 参考代码

```
#include<bits/stdc++.h>
using namespace std;
int a[20], h, ans;
int main()
{
    for (int i = 0; i < 10; i ++) scanf("%d", &a[i]); //存入数组
    scanf("%d", &h);
    for (int i = 0; i < 10; i ++)
        if (a[i] <= h + 30) ans ++; //从头到尾扫一遍统计答案
    printf("%d", ans);
    return 0;
}
```

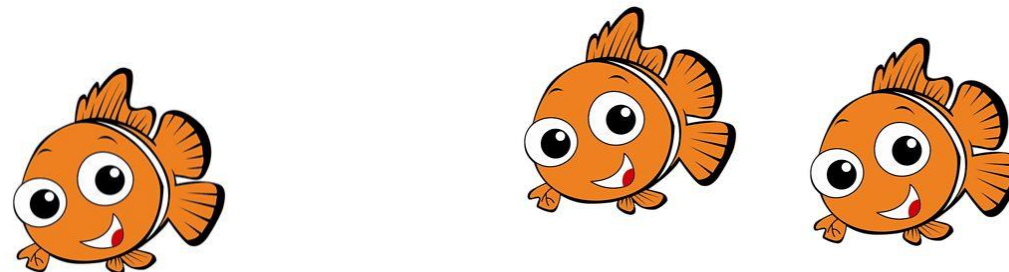


# 拓展

- 如果苹果个数未知（假设至多10万个苹果），你还会做吗？

```
#include<bits/stdc++.h>
using namespace std;
int a[100010], x, cnt, ans;    // cnt是读入数据个数
int main()
{
    while (scanf("%d", &x) == 1) a[++ cnt] = x;
    int h = a[cnt];    //数组最后一个元素是陶陶的身高
    for (int i = 1; i < cnt; i ++)    //第一个苹果存在a[1]中
        if (a[i] <= h + 30) ans ++;
    printf("%d", ans);
    return 0;
}
```

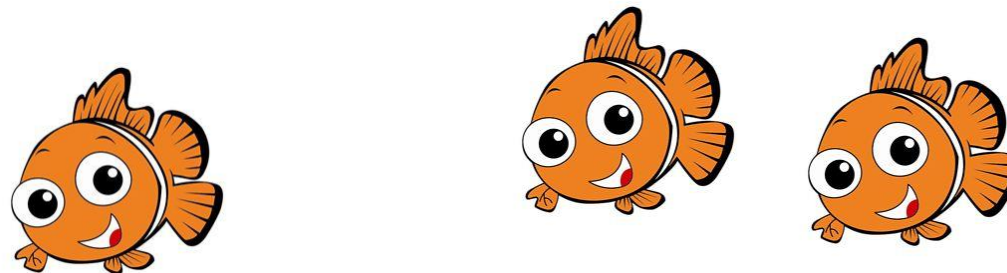
# 小鱼比可爱



- 小鱼最近参加了一个“比可爱”比赛，比的是每只小鱼的可爱程度。
- 参赛的鱼被从左到右排成一排，头都朝向左边，然后每只小鱼会得到一个整数数值，表示可爱程度，整数越大，表示越可爱。
- 由于所有的鱼头都朝向左边，所以每只小鱼只能看见它左边的鱼的可爱程度。请帮它们计算，在自己的眼力范围内有多少只鱼不如自己可爱。

Sample input		Sample output
6	//小鱼的数量（至多100）	0 0 0 3 1 2
4 3 0 5 1 2	//每只小鱼的可爱度	

# 分析



- 研究样例答案是怎样得出的：把每个数左边比它小的数的个数统计出来就可以了
- 唯一要注意的细节是：每只鱼的答案是独立的

# 参考代码

```
#include<bits/stdc++.h>
using namespace std;
int a[110], n;
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i ++){
        scanf("%d", &a[i]);
    }
    for (int i = 0; i < n; i ++){
        int ans = 0; //注意每只鱼统计答案前要置零
        for (int j = 0; j < i; j ++){ //这里的范围只到 i 即可
            if (a[j] < a[i]) ans ++;
        }
        printf("%d ", ans);
    }
    return 0;
}
```

但其实我们完全没有必要循环两遍：因为每只鱼的答案只与已经读入的数据有关



# 优化一下

虽然这题的数据规模，  
跑两遍和跑一遍几乎可以  
忽略不计，只是优化程序的一小步，  
但却是优化常数的思想启蒙



```
#include<bits/stdc++.h>
using namespace std;
int a[110], n;
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i ++)
    {
        scanf("%d", &a[i]); //可以边输入边统计
        int ans = 0;
        for (int j = 0; j < i; j ++)
            if (a[j] < a[i]) ans ++;
        printf("%d ", ans);
    }
    return 0;
}
```

# 标记思想

- 简单来说，就是给数组中符合条件的元素逐一打上标记（true/false等）
- 然后再扫一遍，检查哪些元素被标记了

# 校门外的树

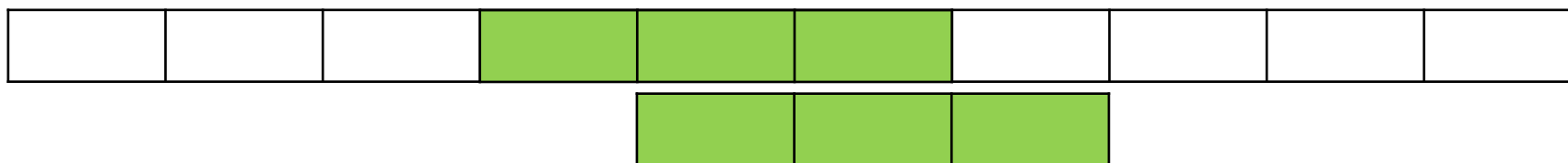
- 学校大门外长度为  $L$  的路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把路看成一个数轴，数轴上的每个整数点，即  $0、1、2、……、L$ ，都种有一棵树。
- 由于路上有一些区域要用来建地铁站，现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树都移走后，路上还剩有多少棵树。



Sample input	Sample output
500 3 //L和M，路的长度和区域数量 150 300 //每个区域的起止点，下同 100 200 470 471	298

# 分析

- 一个数组元素记录一棵树的状态。刚开始树都在，全标记为true
- 然后把指定范围内的数组元素逐个标记为false
- 最后统计仍然为true的元素个数即为答案
- 区间可能有重叠？
- 注意有个细节：刚开始有 $L+1$ 棵树





# 参考代码

```
#include<bits/stdc++.h>
using namespace std;
bool tree[10010]; //用 bool 型数组做标记
int L, M, ans;
int main()
{
    for (int i = 0; i < 10010; i ++) tree[i] = true; //初始化数组
    scanf("%d%d", &L, &M);
    while (M --)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        for (int i = a; i <= b; i ++) tree[i] = false;
    }
    for (int i = 0; i <= L; i ++)
        if (tree[i]) ans ++;
    printf("%d", ans);
    return 0;
}
```

# 也可以不用bool数组

这里的  
if(tree[i]==0)  
可以简写为:  
if(!tree[i])



```
#include<bits/stdc++.h>
using namespace std;
int tree[10010], L, M, ans;
int main()
{
    scanf("%d%d", &L, &M);
    while (M --) //有 M个区域要处理
    {
        int a, b;
        scanf("%d%d", &a, &b); //逐个读入区域
        for (int i = a; i <= b; i ++) tree[i] = 1;
    }
    for (int i = 0; i <= L; i ++)
        if (tree[i] == 0) ans ++;
    printf("%d", ans);
    return 0;
}
```

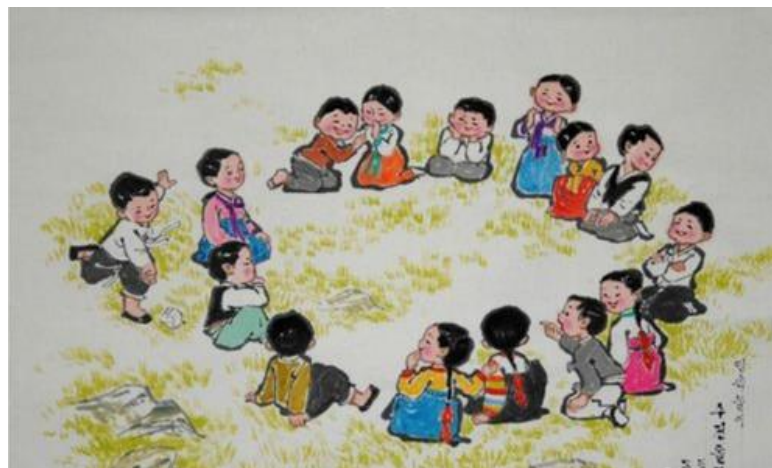
# 牛刀小试（课后加练）

- Luogu 1996      约瑟夫问题
- Luogu 1059      明明的随机数
- Luogu 1138      第k小整数
- Luogu 1862      输油管道问题

# 约瑟夫问题

- 编号为  $1 \sim n$  的  $n$  ( $n \leq 1000$ ) 个人围坐一圈，从第一个人开始报数，报到  $m$  ( $m \leq n$ ) 的人出局，重复这一过程直至剩下一个人即为获胜者。
- 输出所有人出列的先后顺序（最后一位即为获胜者）

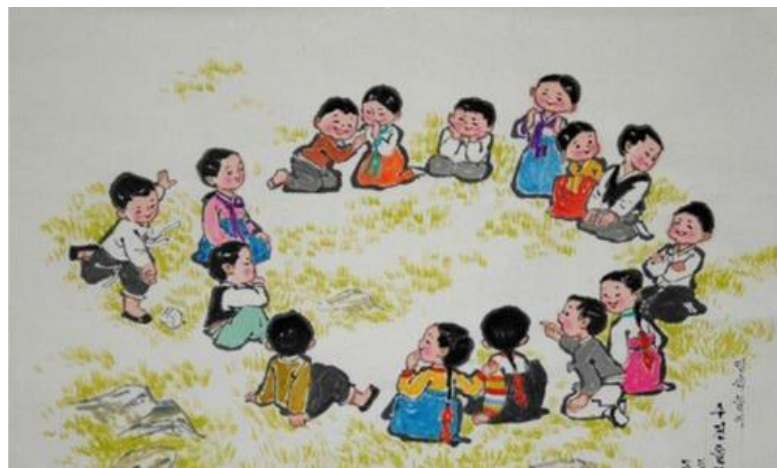
Sample input		Sample output
7	//n	3 6 2 7 5 1 4
3	//m	



# 分析

- 首先我们无法预知该游戏进行几轮后结束，所以判定游戏结束的标志是所有人出局，需要记录出局人数

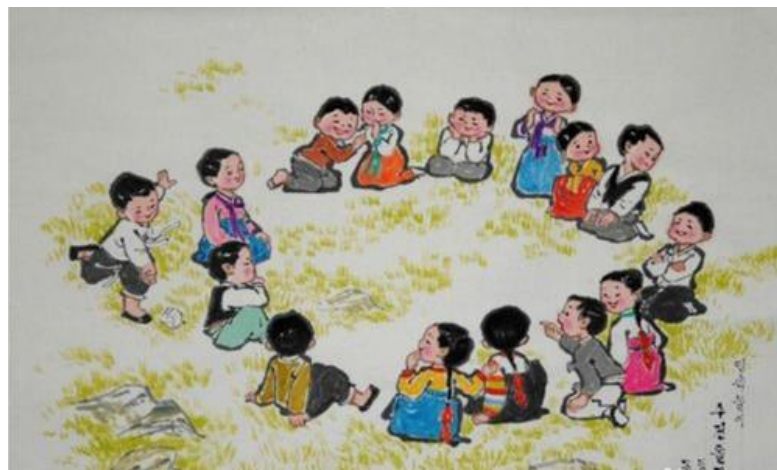
```
while (记录的出局人数 < 初始人数 n)
{
}
}
```



# 分析

- 循环内要做什么呢？
- 首先需要有一个大圈报数，而且要往复进行（因为是环形）

```
while (记录的出局人数 < 初始人数 n)
{
    从编号1开始大圈报数
    一旦发现报数到  $n+1$ ，就要重置为 1
}
```

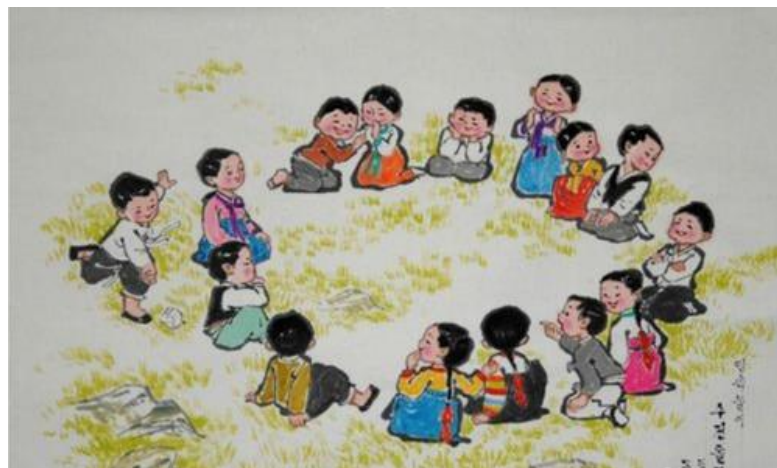


# 分析

- 接下来要小圈报数
- 小圈报数只有未出局的人才能参与，所以需要（用数组）记录每个人的状态

```
while (记录的出局人数 < 初始人数 n)
{
    从编号1开始循环报数
    一旦发现报到  $n+1$ ，就要重置为 1

    未出局的人，参与小圈报数直到 m
}
```



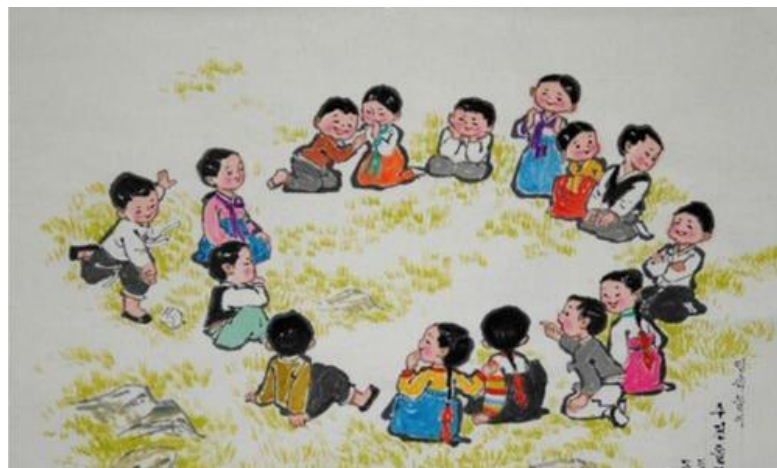
# 分析

- 接下来就要处理小圈报数到 $m$ 的人
  1. 输出该位置并修改对应位置的数组标记为“出局”
  2. 出局人数+1，并且小圈报数要清零（大圈报数不必）

```
while (记录的出局人数 < 初始人数 n)
{
    从编号1开始循环报数
    一旦发现报到  $n+1$ ，就要重置为 1

    未出局的人，参与小圈报数

    处理小圈报数到 $m$ 的那个人（出局）
}
```





# 参考代码

这里的  
if(!a[i]) 就是:  
if(a[i]==0)



```
#include<bits/stdc++.h>
using namespace std;
int a[1010], n, m, i, j, sum; // 数组初始值 0代表未出局
int main()
{
    scanf("%d%d", &n, &m);
    while (sum < n)
    {
        i ++; if (i == n + 1) i = 1; // 从编号 1 开始循环报数
        if (!a[i]) j ++; // 未出队的参与小圈报数
        if (j == m) // 处理出队的编号
        {
            printf("%d ", i);
            a[i] = 1; // 修改对应数组元素为 1, 代表已出局
            sum ++;
            j = 0;
        }
    }
    return 0;
}
```

# 也可以用bool数组

并不推荐这样用memset, 详见后页



```
#include<bits/stdc++.h>
using namespace std;
bool a[1010];
int n, m, i, j, sum;
int main()
{
    memset(a, true, sizeof(a)); // 数组全体赋初值为 true, 代表未出局
    scanf("%d%d", &n, &m);
    while (sum < n)
    {
        i ++; if (i == n + 1) i = 1; // 从编号 1 开始循环报数
        if (!a[i]) j ++; // 未出队的参与小圈报数
        if (j == m) // 处理出队的编号
        {
            printf("%d ", i);
            a[i] = false; // 修改对应数组元素为 false, 代表已出局
            sum ++;
            j = 0;
        }
    }
    return 0;
}
```

# memset 函数

- 用于数组整体初始化
- 如果是对高维数组/结构体数组之类，方便程度更甚

```
int a[10010];  
memset(a, 0, sizeof(a));
```

```
int a[10010];  
for (int i = 0; i < 10010; i ++)  
    a[i] = 0;
```

# memset 函数

- 但要注意它是按字节对指定内存区域进行初始化的
- 所以memset一般只用来清零或者置-1，或者赋极大值，而不可贸然赋 1 之类

```
#include<bits/stdc++.h>
using namespace std;
int a[100010], b[100010], c[100010];
int main()
{
    memset(a, 0, sizeof(a));
    memset(b, -1, sizeof(b));
    memset(c, 0x3f, sizeof(c));
    return 0;
}
```

# 漫谈无穷大/极大值：INF



- 我们现在设置无穷大/极大值（常表示为：INF），一般是用2147483647
- 但是这个数有个问题：它处在溢出的边沿，稍有不慎就溢出，实在危险
- 那怎么办？因为我们以前说过：int 型上限可视作九位数，所以可以设置为999999999，它足够大了，又比2147483647安全
- 但一来看起来很不专业，二来非常容易数错，尬
- 所以又衍生出1234567890这种，但依然很LOW的样子

# 漫谈无穷大/极大值：INF



- 有一种做法是设置为十六进制数：0x7fffffff
- 除了“看起来”很专业这一点外，并没有比2147483647更高明

```
#include<bits/stdc++.h>
using namespace std;
const int INF = 0x7fffffff; // const为定义常量
int main()
{
    int a = INF;
    printf("%d", a);
    return 0;
}
```

```
2147483647
-----
Process exited after 0.1018 seconds w
ith return value 0
请按任意键继续. . .
```



# 漫谈无穷大/极大值：INF

- 终极做法是设置为十六进制数：0x3f3f3f3f
- 这个值1061109567，足够大，又安全
- 但看起来平淡无奇？

```
#include<bits/stdc++.h>
using namespace std;
const int INF = 0x3f3f3f3f; // const为定义常量
int main()
{
    int a = INF;
    printf("%d", a);
    return 0;
}
```

1061109567

-----  
Process exited after 0.1422 seconds w  
ith return value 0  
请按任意键继续. . .

# 0x3f3f3f3f的精妙之处



- 0x3f3f3f3f不仅仅是凭“看起来更专业”战胜1234567890之流的
- 首先，1234567890处理不了无穷大+无穷大的情况，但0x3f3f3f3f可以
- 其次，使用memset为数组整体赋极大值的时候，只能使用0x3f（之前介绍的0x7f也可以）




# memset 赋极大值



读音: [miào] 🔊

部首: 女 五笔: VITT

释义: 1.好; 美妙。 2.神奇; 巧妙; 奥妙。 3.姓。



```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int a[10];
    memset(a, 1234567890, sizeof(a));
    for (int i = 0; i < 10; i ++)
        printf("%d ", a[i]);
    return 0;
}
```

```
-757935406 -757935406 -757935406 -757
935406 -757935406 -757935406 -7579354
06 -757935406 -757935406 -757935406
```

```
-----
Process exited after 0.02245 seconds
with return value 0
请按任意键继续. . .
```

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int a[10];
    memset(a, 0x3f, sizeof(a));
    for (int i = 0; i < 10; i ++)
        printf("%d ", a[i]);
    return 0;
}
```

```
1061109567 1061109567 1061109567 1061
109567 1061109567 1061109567 10611095
67 1061109567 1061109567 1061109567
```

```
-----
Process exited after 0.1341 seconds w
ith return value 0
请按任意键继续. . .
```