实验舱
青少年编程
走近科学 走进名校

挑战信息学奥林匹克

C++程序设计（2）

一维数组应用

一维数组

- 数组输入

```
for ( int i = 1; i <= n; i++ )
{
    cin >> a[i];
}
```

```
for ( int i = 0; i < n; i++ )
{
    cin >> a[i];
}
```

- 枚举数组

```
for ( int i = 1; i <= n; i++ )
{
    ..... a[i] .....;
}
```

```
for ( int i = 0; i < n; i++ )
{
    ..... a[i] .....;
}
```

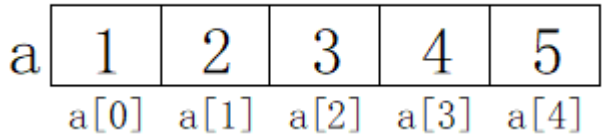
- 数组输出

```
for ( int i = 1; i <= n; i++ )
{
    cout << a[i] << ' ';
}
```

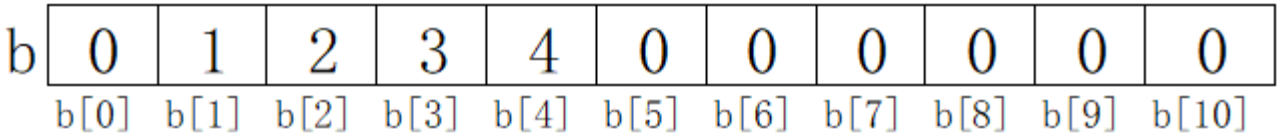
```
for ( int i = 0; i < n; i++ )
{
    cout << a[i] << ' ';
}
```

一维数组的初始化

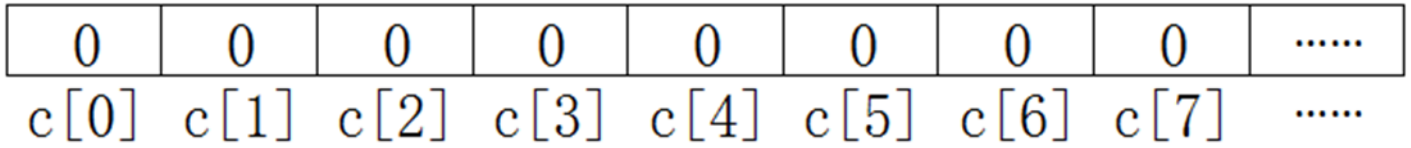
int a[5]={1,2,3,4,5};



int b[10]={0,1,2,3,4};



int c[100]={0};



观察与思考：输出会一样吗？

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int a[5];
    for ( int i = 0; i < 5; i++ )
    {
        cout << a[i] << ' ';
    }
    cout << endl;
    return 0;
}
```

```cpp
#include <bits/stdc++.h>
using namespace std;

int a[5];
int main()
{
    for ( int i = 0; i < 5; i++ )
    {
        cout << a[i] << ' ';
    }
    cout << endl;
    return 0;
}
```

一维数组清0方法

方法一：数组赋初值0

```
int a[10001] = {0};
```

方法二：枚举数组逐个元素置0

```
for ( int i = 0; i < n; i++ )
{
        a[i] = 0;
}
```

方法三：使用函数

```
memset(a, 0 ,sizeof(a));
```

例1：数字统计

输入n个不大于1000的正整数，统计每个数出现的次数。

输入样例

18

3 2 5 6 8 1 3 5 8 9 8 9 8 7 10 4 1 4

输出样例

1 2
2 1
3 2
4 2
5 2
6 1
7 1
8 4
9 2
10 1

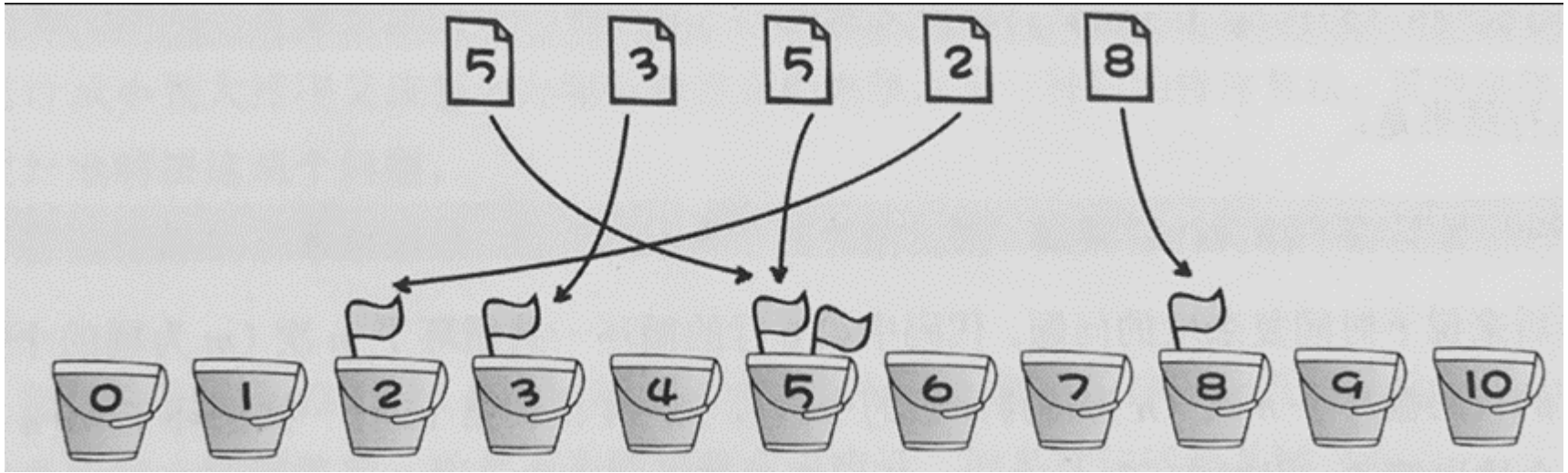
查找？

查找算法

```
cin >> n;
for (int i = 1; i <= n; i++) cin >> a[i];
for (int i = 0; i <= 1000; i++)
{
    sum = 0;
    for (int j = 1; j <= n; j++)
    {
        if (a[j] == i) sum++;
    }
    if (sum > 0) cout << i << ' ' << sum << endl;
}
```

这是一个效率低下的算法！

利用数组下标统计数字个数

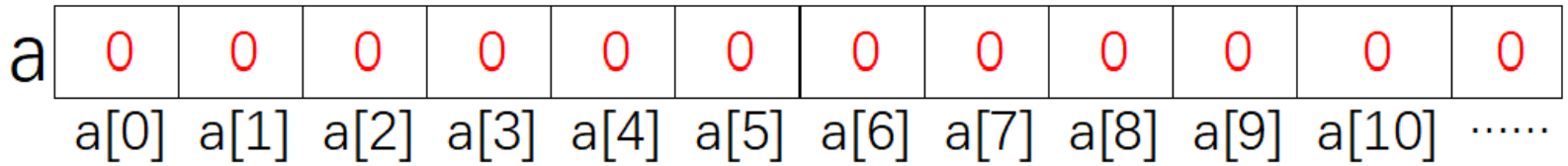


算法分析

输入

18

3 2 5 6 8 1 3 5 8 9 8 9 8 7 10 4 1 4

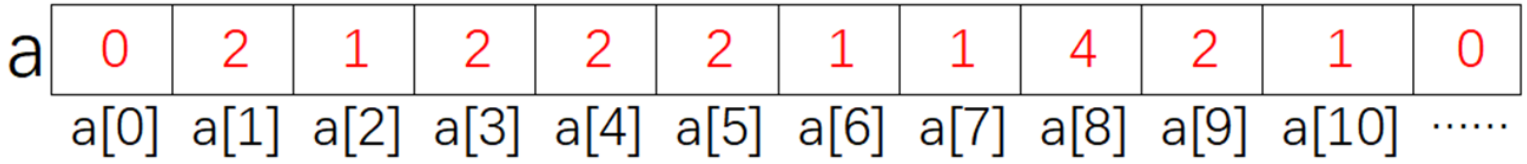


算法分析

输入

18

3 2 5 6 8 1 3 5 8 9 8 9 8 7 10 4 1 4



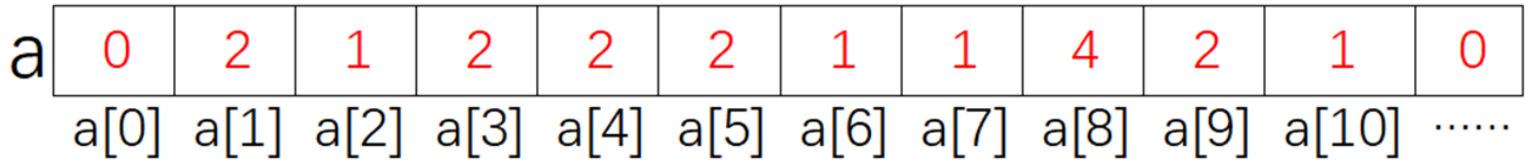

```
for ( int i = 0; i < n; i++ )
{
    cin >> x;
    a[x]++;
}
```

算法分析

输入

18

3 2 5 6 8 1 3 5 8 9 8 9 8 7 10 4 1 4



输出样例

1 2

2 1

3 2

4 2

5 2

6 1

7 1

8 4

9 2

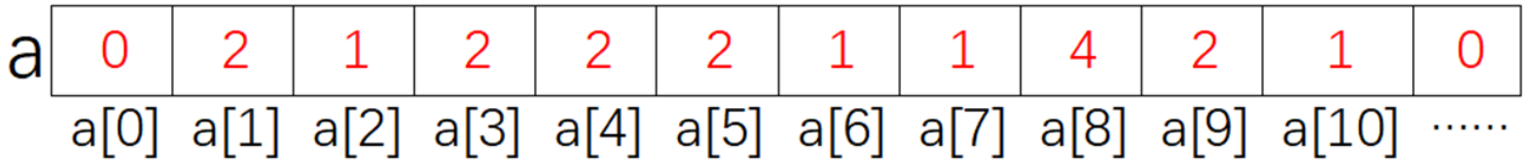
10 1

算法分析

输入

18

3 2 5 6 8 1 3 5 8 9 8 9 8 7 10 4 1 4




```
for ( int i = 0;i < mm; i++ )
{
    if ( a[i] > 0)
        cout << i << " " << a[i] << endl;
}
```

输出样例

1 2
2 1
3 2
4 2
5 2
6 1
7 1
8 4
9 2
10 1

算法设计

1. 定义数组a;
2. 输入数据个数n;
3. 逐个输入整数x，进行下标统计;
4. 将统计结果输出

```
const int mm = 1001;
int a[mm];

int main()
{
    int n, x;
    cin >> n;
    for ( int i = 0; i < n; i++ )
    {
        cin >> x;
        a[x]++;
    }
    for ( int i = 0;i < mm; i++ )
    {
        if ( a[i] > 0)
            cout << i << " " << a[i] << endl;
    }
    return 0;
}
```

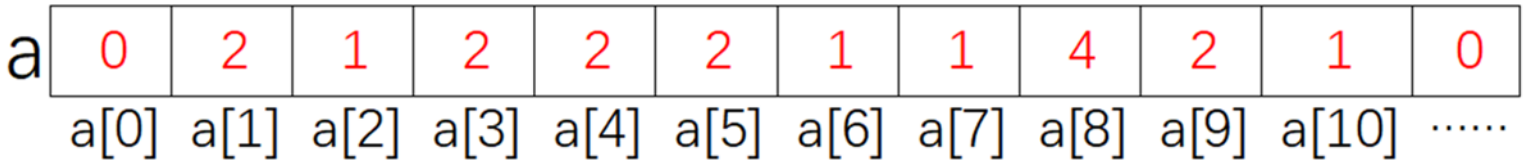
例2：去重

输入n个不大于1000的整数，按照从小到大顺序输出不重复的数字。

输入样例

18

3 5 5 6 8 1 3 5 8 9 8 9 8 9 10 4 1 4



输出样例

1

3

4

5

6

8

9

10

算法设计

1. 定义数组a;
2. 输入数据个数n;
3. 逐个输入整数x，进行下标统计;
4. 将统计结果输出

```
const int mm = 1001;
int a[mm];

int main()
{
    int n, x;
    cin >> n;
    for ( int i = 0; i < n; i++ )
    {
        cin >> x;
        a[x]++;
    }
    for ( int i = 0;i < mm; i++ )
    {
        if ( a[i] > 0)
            cout << i << " " << a[i] << endl;
    }
    return 0;
}
```

例3：排序

输入n个不大于1000的整数，将数字从小到大排列后输出。

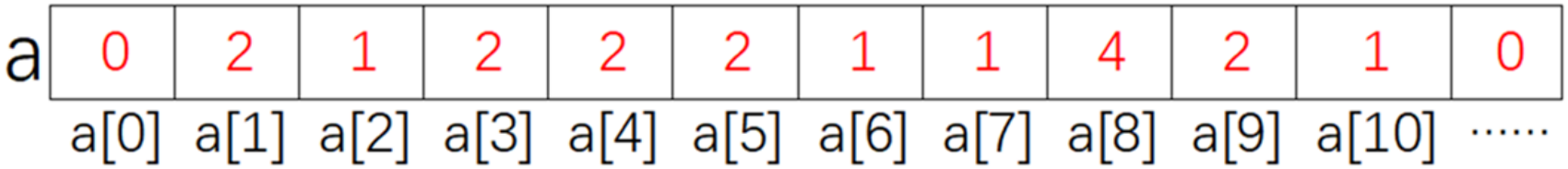
输入样例

18

3 2 5 6 8 1 3 5 8 9 8 9 8 7 10 4 1 4

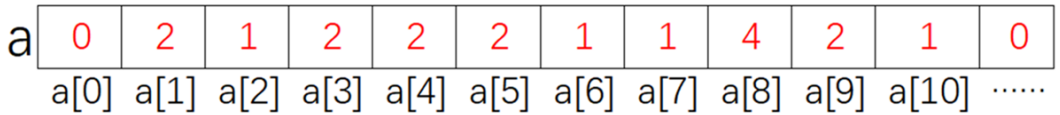
输出样例

1 1 2 3 3 4 4 5 5 6 7 8 8 8 8 9 9 10



算法设计

1. 定义数组a;
2. 输入数据个数n;
3. 逐个输入整数x，进行下标统计；（桶排）
4. 将统计结果输出




```
const int mm = 1001;
int a[mm];

int main()
{
    int n, x;
    cin >> n;
    for ( int i = 0; i < n; i++ )
    {
        cin >> x;
        a[x]++;
    }
    for ( int i = 0;i < mm; i++ )
    {
        if ( a[i] > 0)
            cout << i << " " << a[i] << endl;
    }
    return 0;
}
```

算法设计

1. 定义数组a；
2. 输入数据个数n；
3. 逐个输入整数x，进行下标统计；
   （桶排）
4. 将统计结果输出

```
const int mm = 1001;
int a[mm];
int main()
{
    int n, x;
    cin >> n;
    for ( int i = 0; i < n; i++ )
    {
        cin >> x;
        a[x]++;
    }
    for ( int i = 0; i < mm; i++ )
    {
        if ( a[i] > 0 )
        {
            for ( int j = 0; j < a[i]; j++ ) cout << i << " ";
        }
    }
    return 0;
}
```

利用下标统计数字算法小结

- 功能
  1. 排序（桶排序）
  2. 数字统计
  3. 去重
- 需要数组满足待排序的最大数字
- 如果数字大小超过10^6，则不考虑使用下标统计算法。

例4：数的频度

描述

小明学会了用程序生成随机数，就想知道生成的随机数的频度是多少（频度为相同数字出现的次数），他想到用频度差来表示n个整数的频度。所谓的频度差，就是数字出现最多的次数，和出现最少的次数之差。

输入

二行。

第一行，一个整数n（3≤n≤1000000)。

第二行，n个整数（大于等于1，小于等于1000）。

输出

一个整数，频度差。

输入样例

10

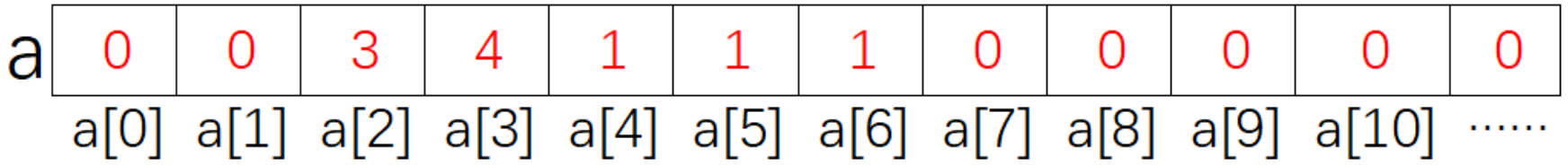
2 3 2 2 3 4 3 5 3 6

输出样例

3

问题分析

- 利用桶排序，统计各个数字出现的次数。
- 对桶排的结果查找最大值和最小值。
- 输出最大值和最小值的差值。



查找最大值和最小值

```
int mmax = 0;
int mmin = 10000;
for ( int i = 0; i <= 1000; i++ )
{
    if (a[i] > 0)
    {
        if ( a[i] > mmax ) mmax = a[i];
        if ( a[i] < mmin ) mmin = a[i];
    }
}
```

例5：开关灯问题

假设有N盏灯，从1到N按顺序依次编号，初始时全部处于开启状态；有M个人也从1到M依次编号。

第一个人（1号）将灯全部关闭，第二个人（2号）将编号为2的倍数的灯打开，第三个人（3号）将编号为3的倍数的灯做相反处理（即，将打开的灯关闭，将关闭的灯打开）。依照编号递增顺序，以后的人都和3号一样，将凡是自己编号倍数的灯做相反处理。

请问：当第M个人操作之后，哪几盏灯是关闭的，按从小到大输出其编号，其间用空格间隔。

例6：开关灯问题

【输入说明】

一行两个整数N (N不大于5000)和M (M不大于N)，以单个空格隔开。

【输出说明】

顺次从小到大输出关闭的灯的编号，其间用空格隔开。

【输入样例】

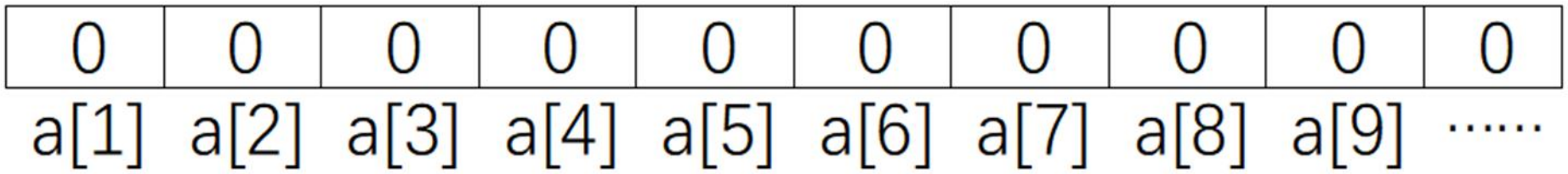
20 10

【输出样例】

1,4,9,11,12,13,14,15,17,18,19,20

算法分析：模拟开关灯过程

- 用一维数组模拟一组灯的开关状态
- 下标为灯的编号
- 值为0时，表示灯是关闭的，值为1时，表示灯是开启的。



算法分析：模拟开关灯过程

- 二重循环模拟开关过程

```
for (int i=2;i<=m;i++)
    for (int j=i;j<=n;j++)
        if (j%i==0) a[j]=!a[j];
```

程序代码

```
cin >> n >> m;
for ( int i = 2; i <= m; i++ )
    for ( int j = i; j <= n; j++ )
        if ( j % i == 0 ) a[j] =! a[j];

for ( int i = 1; i <= n; i++ ){
    if ( a[i] == 0 ) {
        if ( bj == 0){
            cout << i;
            bj = 1;
        }
        else cout << ',' << i;
    }
}
```

例6：校门外的树

某校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置；数轴上的每个整数点，即0，1，2，……，L，都种有一棵树。由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。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，马路上还有多少棵树。

校门外的树

【输入】

第一行有两个整数L（1 <= L <= 10000）和M（1 <= M <= 100），L代表马路的长度，M代表区域的数目，L和M之间用一个空格隔开。接下来的M行每行包含两个不同的整数，用一个空格隔开，表示一个区域的起始点和终止点的坐标。

【输出】

一行，这一行只包含一个整数，表示马路上剩余的树的数目。

【样例输入】

500 3

150 300

100 200

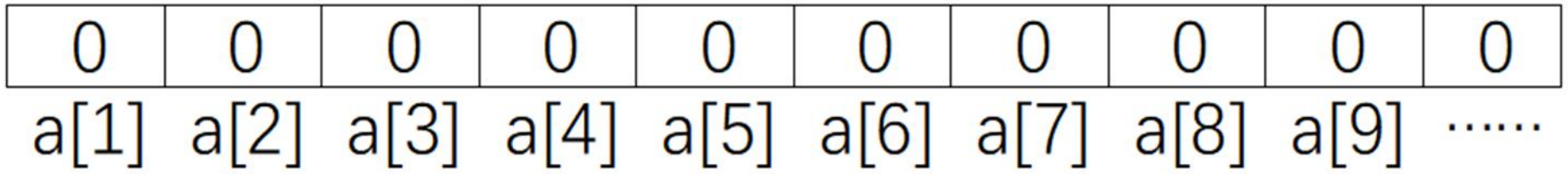
470 471

【样例输出】

298

问题分析

- 根据题目的数据范围，可以考虑用模拟方法。
- 用一个数组模拟马路种树的状态，初始状态为0，表示所有的树都在。
- 读入一个区间范围，就将该区间范围的数组元素值都标记为1。
- 顺序访问数组，统计数组元素值为0的个数。



程序代码

```
int a[10001];
int main()
{
    int L,M,s,e;
    cin>>L>>M;
    for (int i=0;i<M;i++)
    {
        cin>>s>>e;
        for (int j=s;j<=e;j++) a[j]=1;   //标记移除的树
    }
    int t=0;
    for (int i=0;i<=L;i++)   //统计还有多少棵树
        if (a[i]==0) t++;
    cout <<t<<endl;
    return 0;
}
```

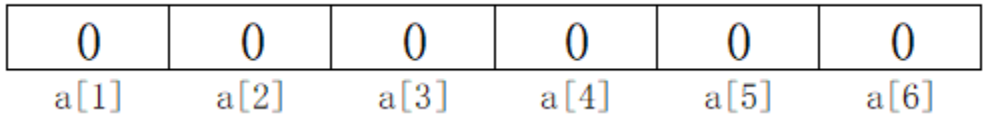
例7：约瑟夫问题

- N个人围成一圈，从第一个开始报数，第M个将出圈，最后剩下一个人，其余人都出圈。
- 例如N=6，M=5，出圈的顺序是：5，4，6，2，3，1。
- 编一个程序，输入N、M，输出出圈人的次序。

【分析】

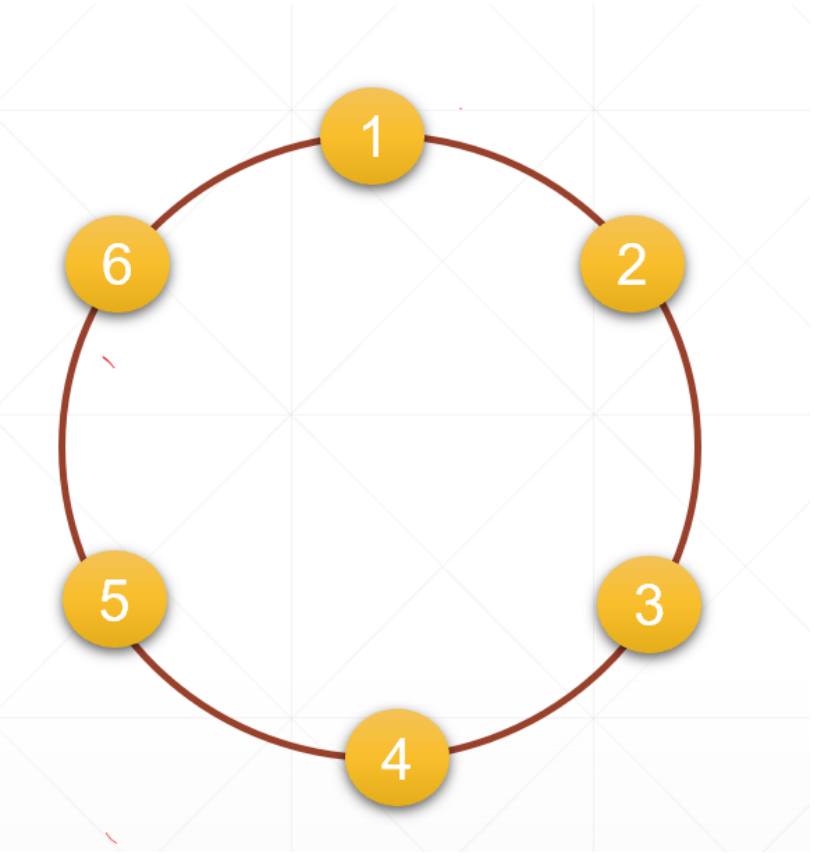
（1）用数组记录是否出圈的状态；

（2）模拟报数的过程。



问题分析

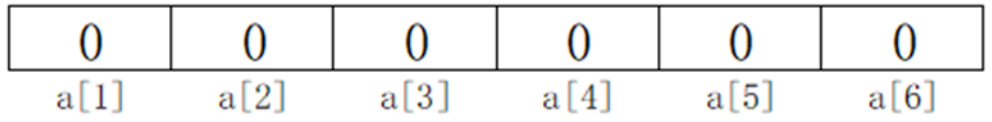
- 模拟报数过程
- 用数组构造一个约瑟夫圈
    1. 数组的下标是n个人的编号
    2. 数组元素的值用0和1表示是否出圈的状态



0	0	0	0	0	0
a[1]	a[2]	a[3]	a[4]	a[5]	a[6]

问题分析

- 模拟报数过程
- 用数组构造一个约瑟夫圈
  1. 数组的下标是n个人的编号
  2. 数组元素的值用0和1表示是否出圈的状态




```
while (f<n)
{
      t++;
      if (t>n) t=1;
      if (a[t]==0) s++;
      if (s==m)
      {
            cout <<t<<" ";
            a[t]=1;
            s=0;
            f++;
      }
}
```

程序代码

```
int a[110];
int n,m;
cin >>n>>m;
memset(a,0,sizeof(a));
int t=0;     //记录下标位置
int s=0;     //报数
int f=0;     //记录出圈人数
while (f<n)
{
    t++;
    if (t>n) t=1;
    if (a[t]==0) s++;
    if (s==m)
    {
        cout <<t<<" ";
        a[t]=1;
        s=0;
        f++;
    }
}
```