1.数组

2.链表

3.数学方法求最后一个人

4.数学方法求第k个人

1.数组

[复制代码](javascript:void(0);)

1 #include <stdio.h>

2 #include <stdlib.h>

3

4 int main()

5 {

6 int next[1000],n,s,i,j,cur=1;

7 scanf("%d%d",&n,&s);

8 if (s==1)

9 {

10 for (i=1;i<=n;i++)

11 printf("%d ",i);

12 return 0;

13 }

14 for (i=1;i<n;i++)

15 next[i]=i+1;

16 next[n]=1;

17 for (i=1;i<=n;i++)

18 {

19 for (j=1;j<s-1;j++)

20 cur=next[cur];

21 printf("%d ",next[cur]);

22 next[cur]=next[next[cur]];

23 cur=next[cur];

24 }

25 return 0;

26 }

[复制代码](javascript:void(0);)

2.链表

[复制代码](javascript:void(0);)

1 ·#include <stdio.h>

2 #include <stdlib.h>

3 #include <malloc.h>

4

5 int main()

6 {

7 struct node

8 {

9 int num;

10 struct node \* next;

11 };

12 struct node \*line,\*temp,\*cur;

13 int n,m,i,j;

14 scanf("%d%d",&n,&m);

15 if (m==1)

16 {

17 for (i=1;i<=n;i++)

18 printf("%d ",i);

19 return 0;

20 }

21 line=(struct node \*) malloc (sizeof(struct node));

22 //line=new node;

23 line->num=1;

24 cur=line;

25 for (i=2;i<=n;i++)

26 {

27 temp=(struct node \*) malloc (sizeof(struct node));

28 //temp=new node;

29 temp->num=i;

30 line->next=temp;

31 line=temp;

32 }

33 line->next=cur;

34 for (i=1;i<=n;i++)

35 {

36 for (j=1;j<m-1;j++)

37 cur=cur->next;

38 printf("%d ",cur->next->num);

39 cur->next=cur->next->next;

40 cur=cur->next;

41 }

42 return 0;

43 }

[复制代码](javascript:void(0);)

3.数学方法求最后一个人

[复制代码](javascript:void(0);)

1 #include <stdio.h>

2 #include <stdlib.h>

3

4 int main()

5 {

6 int n,m,f[1000],i;

7 scanf("%d%d",&n,&m);

8 f[1]=0;

9 for (i=2;i<=n;i++)

10 f[i]=(f[i-1]+m)%i;

11 printf("%d\n",f[n]+1);

12 return 0;

13 }

[复制代码](javascript:void(0);)

4.数学方法求第k个人

[复制代码](javascript:void(0);)

1 #include <stdio.h>

2 #include <stdlib.h>

3

4 int main()

5 {

6 //n个人，每m个人除去一个人，求k轮除去的人的编号

7 //f[n][k]可以分为（第一轮报数）+（n-1个人报k-1轮）

8 //f[n][k]为第一轮报数的编号的基础上加上n-1个人在第k-1轮除去的人的编号

9 //f[n][k]=(f[n-1][k-1]+m) % n

10 //f[n-k+1][1]=m-1

11 int n,m,k,i,x;

12 scanf("%d%d%d",&n,&m,&k);

13 x=(m-1)%(n-k+1);

14 //编号认为从0~n-1;因为若为1~n，%n有可能出现0的情况，不太好

15 for (i=n-k+2;i<=n;i++)

16 x=(x+m)%i;

17 printf("%d\n",x+1);

18 return 0;

19 }

[复制代码](javascript:void(0);)

附上他人的数学方法证明

[**约瑟夫问题的数学方法**](http://applebolanshu-126-com.javaeye.com/blog/368401)

无论是用链表实现还是用数组实现都有一个共同点：要模拟整个游戏过程，不仅程序写起来比较烦，而且时间复杂度高达O(nm)，当n，m非常大(例如上百万，上千万)的时候，几乎是没有办法在短时间内出结果的。我们注意到原问题仅仅是要求出最后的胜利者的序号，而不是要读者模拟整个过程。因此如果要追求效率，就要打破常规，实施一点数学策略。  
  
为了讨论方便，先把问题稍微改变一下，并不影响原意：  
  
问题描述：n个人（编号0~(n-1))，从0开始报数，报到(m-1)的退出，剩下的人继续从0开始报数。求胜利者的编号。  
  
我们知道第一个人(编号一定是m%n-1) 出列之后，剩下的n-1个人组成了一个新的约瑟夫环（以编号为k=m%n的人开始）:  
   k   k+1   k+2   ... n-2, n-1, 0, 1, 2, ... k-2  
并且从k开始报0。  
  
现在我们把他们的编号做一下转换：  
k      --> 0  
k+1    --> 1  
k+2    --> 2  
...  
...  
k-2    --> n-2  
k-1    --> n-1  
  
变换后就完完全全成为了(n-1)个人报数的子问题，假如我们知道这个子问题的解：例如x是最终的胜利者，那么根据上面这个表把这个x变回去不刚好就是n个人情况的解吗？！！变回去的公式很简单，相信大家都可以推出来：x'=(x+k)%n  
  
如何知道(n-1)个人报数的问题的解？对，只要知道(n-2)个人的解就行了。(n-2)个人的解呢？当然是先求(n-3)的情况 ---- 这显然就是一个倒推问题！好了，思路出来了，下面写递推公式：  
  
令f[i]表示i个人玩游戏报m退出最后胜利者的编号，最后的结果自然是f[n]  
  
递推公式  
f[1]=0;  
f[i]=(f[i-1]+m)%i;   (i>1)  
  
有了这个公式，我们要做的就是从1-n顺序算出f[i]的数值，最后结果是f[n]。因为实际生活中编号总是从1开始，我们输出f[n]+1.